Applied and industrial algebra

Tim Hsu, San José State University

December 7, 2024

Contents

Introduction

Cash Rules Everything Around Me
C.R.E.A.M., get the money
Dollar dollar bill, y'all

— Wu-Tang Clan, *C.R.E.A.M.*

One of the tricky parts of writing a book in applied math is deciding what you mean by "applied". This book uses the following definition of applied and industrial math:

> *Applied math is math you can use to make money outside of academia (aka the "real world").*

That definition of "applied" lies behind all of the topics in this book, including:

- The Euclidean algorithm
- Error-correcting codes
- Finite fields
- The Discrete Fourier Transform and the Fast Fourier Transform

(And yes, in case you were wondering, cryptography also falls under the heading of algebra you can use to make money in the real world, but algebra in cryptography is the main subject of a lot of other books, so there's only a bit of it in here.)

In full disclosure, there's another motive behind this book and its choice of material. Namely, by the end of this book, you should have an idea of what the basic objects of study are in abstract algebra, and you should also know some practical reasons why you'd want to study them. Moreover, each topic in the book is designed to illustrate the following idea:

$$\text{Abstraction} \Rightarrow \text{Simplification} \Rightarrow \text{Generalization} \Rightarrow \text{Power}$$

And indeed, the not-so-secret hope is that you'll find that process so interesting that you'll go on to take a full theorem-proving course in abstract algebra. Even if you don't, however, if you can become not necessarily a *producer* of abstract algebra (i.e., someone who creates new definitions and theorems), but instead an *informed consumer* of abstract algebra (i.e., someone who understands and applies existing definitions and theorems), then this book will have succeeded.

On the other hand, an informed consumer of abstract algebra does need to be at least acquainted with the role of proof, so this book also serves as an introduction to proof if you need that. Other notable features of the book include:

- Precise, rigorous definitions of objects often appear long after the book has been using those object. For example, the idea of a ring first appears in Section 2.1, but the actual definition of ring is delayed until Chapter 4.
- A few sections cover proof strategies to be used in some frequently occurring situations, such as Section 2.1, where you'll be asked to prove facts about divisibility of integers, and Section 5.3, where you'll be asked to prove facts about spanning and linear independence.
- Going on the philosophy that the proofs you learn the most from are the ones you *do*, not the ones you just read, some of the more approachable proofs of fundamental results appear as problems to be solved by the reader. In particular, a problem labelled (say) "(*Proves Theorem x.y.z*)" has the goal of proving Theorem x.y.z — which means that the answer "This is true because of Theorem x.y.z" is circular logic to be avoided.

(outline of rest of book)
(contents of a typical semester)

Chapter 1

How to think about mathematics

Time to level up.

— Kamala Khan, *Ms. Marvel Vol. 1: No Normal*, G. Willow Wilson

1.1 HEY, YOU!

That's right, you, the reader. Who else would I be talking to?

Sorry about the theatrics, but I wanted to get your attention, because I have good news and bad news. The good news is that I meant what I said in the Introduction: this book will teach you algebra you can use to make money. The bad news is that the thing that makes you money isn't (rote) computation — it's **theory**, or at least conceptual understanding. Why is that?

- In the fanciest possible scenario, you could become someone who creates new theory to solve real-world algorithmic problems.
- In a more common scenario, you can get paid to apply someone else's theory in practice. To do that, you need to be able to understand what happens in abstract algebra at a conceptual level, and not just follow a recipe. Even if you're just using a canned solution someone else implemented, you can reach a whole other level (and pay grade) if you can understand what that canned solution does, instead of just treating it as a black box.
- Conversely, let's face it, no one will pay you to do computation that you can put into a recipe — that's what computers are for.

The fact that theory is the useful part means that you may need to approach this book (and course, if you're reading this book as part of a course) differently than you have approached other math books and courses before.

- You'll need to put more of an emphasis on **language** than you have in previous courses. Much of learning abstract math, especially algebra, is more about learning a particular and precise language than about doing particular computations, so try to stay attuned to the language used as you go through this book. (It helps to have friends to talk with about the material.)

- The first step in mastering the language of algebra is to pay careful attention to the **definitions** you encounter. To start with, you need to memorize definitions very precisely; unlike everyday language, small deviations in wording can completely change the meaning of a mathematical definition. It may help you to think of mathematical definitions not as saying what an idea *means*, but more as *code* (in the sense of computer code!) that programs what an idea *is*.

- In particular, you need not just to memorize, but familiarize yourself with the key **examples** of the book, in the same way that a chef must be familiar with both various knives and cooking methods and also various kinds of spices and ingredients. For example, some of the most important examples in this book include $\mathbf{Z}$, $\mathbf{Z}/(p) = \mathbf{F}_p$, $\mathbf{F}_p^n$, $\mathbf{F}_q$, $F[x]$, $F[x]/(p(x))$, the cyclic group $\langle a \rangle$, and the Nth root of unity ω. Now, there's no reason you should recognize any of those symbols right now! But keep an eye out for them as they appear, and find out everything about them that you can when they do.

- Finally, a small but important point: You'll probably have to pay more attention to cases (upper vs. lower) and fonts (standard vs. boldface) than you may have in the past. For example, the letters f, F, and $\mathbf{F}$ are all different, so as you go through this book, try to absorb the different connotations of each case of font. For example, when we start to consider fields (an idea you have no reason to know about for now), F will usually refer to an unspecified field in the abstract, whereas $\mathbf{F}$ will usually be used in the name of a particular concrete example of a field, like $\mathbf{F}_9$.

In fact, you'll probably have to change the way you read mathematics.

- At this in point in your mathematical career, you may be used to reading math by the following common procedure. (1) You try to do the homework; (2) if you run into problems, you look for an example to imitate; (3) if you don't understand the example, only then do you actually read the main ideas. That may have worked for you so far, but it'll be a very inefficient (and probably ineffective) way to read this book, because in this book, the ability to do the problems comes from understanding the main ideas. Try (3)–(2)–(1) instead, i.e., start with the ideas and work your way to the problems.

- You may need to acccept slow progress in reading. That is, your expectations should be sert so that if you read only a few pages in an hour, instead of thinking "Wow, only a few pages in an hour," you think "Wow, I got through a few pages, and it only took an hour!"

- Every once in a while in this book, you'll encounter an interruption like:

Ask Yourself 1.1.1. *(slightly annoying open-ended question, blah blah)*

If you actually spend a little time thinking about these "Ask Yourself" questions, in the end, it will make reading the book easier, as you'll be in the right frame of mind for what comes next.

- It also happens occasionally that an important idea in a section of the book can really only be understood by doing the problems at the end of that section; this will be indicated by a "see Problem x.y.z" in the text. When that happens, make sure you *do the problem*, if your instructor hasn't already assigned them for you to do or done them in class.

To be clear, it may turn out that the theory in this book will end up being no big deal for you, and reading it won't be much of a challenge. However, if and when things get tough, come back to this section for help on how to proceed.

1.2 Problems vs. exercises

The renowned math educator Paul Zeitz, in his book *The Art and Craft of Problem Solving* [**?**], draws the following distinction between exercises and problems.

> An *exercise* is a question that tests the student's mastery of a narrowly focused technique, usually one that was recently "covered." Exercises may be hard or easy, but they are never puzzling, for it is always immediately clear how to proceed.... A *problem* is a question that cannot be answered immediately. Problems are often open-ended, paradoxical, and sometimes unsolvable, and require investigation before one can come close to a solution.

At this point in your mathematical career, you have almost certainly spent most of your time, maybe all of your time, doing exercises. This book has its share of exercises, but it also definitely has its share of problems, requiring substantial thought, review of fundamental ideas, or at the very least, some experimentation, to solve. After all, to return to our mission statement: What do you think will earn you money in the future, the ability to do exercises, or the ability to solve problems?

When you come up against a true problem, i.e., something you don't know how to solve right away, try the following ideas.

- **Read the problem.** Get a general idea of what the problem is about, and what the goal of the problem is.

- **Review the definitions.** Make sure that you know the definition of all of the mathematical terms occurring in the problem. Especially in algebra, knowing the detailed, precise definition of mathematical terms can be half the battle.

- **Experiment.** Try special cases of the problem: Small cases, random cases. Look for examples to which the problem applies: Is there a typical example, or even a general example?

- **Keep at it.** I can't always guarantee that you'll solve a problem if you keep trying, but I *can* guarantee that you *won't* solve it if you give up.

1.3 Sets, theorems, and proofs

To help make sure everyone starts in the same place, I'm now going to take you through a crash course in the basics of mathematical theory. To be clear, if you've never seen this stuff before, there's no reason that it should all make sense to you immediately; I just want to introduce you to the fundamental ideas you'll need now, and you'll grow to understand them better as you put them into practice in the rest of this book.

1.3.1 Set and set-builder notation

A *set* S is a bunch of objects, and those objects are called the *elements* of S. A finite set can be described by listing its elements inside $\{\ \}$. For example, the elements of the set $S = \{2, 3, 5, 7, 11\}$ are the numbers 2, 3, 5, 7, and 11. We also write $2 \in S$, $3 \in S$, and so on, to mean that 2 is an element of S, 3 is an element of S, and so on.

It's often convenient to describe a set S not by listing the elements of S, but by giving a precise condition for being an element of S. This *set-builder notation* looks something like:

$$S = \{x \mid \text{(defining condition on } x)\}\,. \tag{1.3.1}$$

To break (1.3.1) down, you read the initial $\{$ as "the set of all", the middle $\mid$ as "such that", and the final $\}$ as a sort of period to the sentence. In other words, (1.3.1) says "S is the set of all x such that x satisfies the condition (defining condition)."

To give a concrete example, the set of all even numbers is defined as

$$E = \{n \in \mathbf{Z} \mid n = 2k \text{ for some } k \in \mathbf{Z}\}\,. \tag{1.3.2}$$

As you'll see, $\mathbf{Z}$ denotes the set of all integers (positive, zero, and negative), so (1.3.2) says that E is defined to be the set of all integers n such that n is equal to $2k$ for some integer k. And indeed, that's what you've been told an even integer is your whole life: a whole number that's twice some other whole number.

The following principle describes how to work with a set described in set-builder notation.

The Defining Condition Principle: If a set S is given by a defining condition, then saying that x is an element of S is the same thing as saying that x satisfies the defining condition of S.

For example, to say that

$$m \in \{n \in \mathbf{Z} \mid n = 2k \text{ for some } k \in \mathbf{Z}\} \tag{1.3.3}$$

means that $m = 2r$ for some integer r. This illustrates that you shouldn't get too attached to the particular letter occurring in a "for some" part of a set-builder definition. For example, if you have two even numbers m and n, you shouldn't say that they're both equal to $2k$, because that makes it look like $m = n$; instead, you should say something like $m = 2r$ and $n = 2k$ for some integers r and k.

You will also encounter the following slightly fancier version of set-builder notation:

$$S = \{\textit{foo} \mid (\text{defining condition } \textit{bar})\}\,. \tag{1.3.4}$$

Translated, this says: S is the set of all things that look like *foo* and satisfy condition *bar*.

Again, to give a concrete example, the set $\mathbf{R}^2$ of points in the plane is defined by

$$\mathbf{R}^2 = \left\{ \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \,\middle|\, x_1, x_2 \in \mathbf{R} \right\}. \tag{1.3.5}$$

Putting (1.3.5) in practice, to say that $\mathbf{x} \in \mathbf{R}^2$ means that $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, where $x_1, x_2 \in \mathbf{R}$. In other words, $\mathbf{R}^2$ is the set of all 2×1 column vectors with entries in $\mathbf{R}$.

1.3.2 Definitions vs. theorems

The theoretical structure of mathematics can be broken down into *definitions* and *theorems*, and it's crucial for you to understand the difference between them. The idea is that definitions describe the objects we choose to study, and theorems are logical consequences that we subsequently deduce about those objects.

Much of the power of theoretical mathematics lies in the fact that, if you choose your definitions well, then:

1. The definitions will be natural and simple enough that no reasonable person will disagree with them.

2. Nevertheless, you can deduce interesting theorems about them.

The point is to obtain mathematical conclusions that are based on only a small set of reasonable assumptions, but can nevertheless be applied to lots of different situations.

Now, if you don't have much experience thinking about definition-theorem mathematics, one natural tendency is to lump definitions and theorems together as a list of facts that are all "true." However, to understand mathematical theory it's important that you understand which facts are true by definition (i.e., because we said so), and which facts are true by theorem (i.e., because we deduced them). Make sure to pay attention to the distinction between definitions and theorems as you read this book.

1.3.3 If-then statements and theorems

Most mathematical theorems are *if-then statements*, so it's important for you to know a bit about if-then statements. (This is especially true for the occasional proofs you'll have to think about in this book, but it's even true if you just want to apply what we discuss here.)

An if-then statement has the form “If p, then q,” for some logical statements p and q. The important things to remember about the statement “If p, then q” are:

- Any statement (truthfully) implies a true statement; that is, “If p, then q” is true whenever q is true.
- A false statement (truthfully) implies any statement; that is, “If p, then q” is true whenever q is true.

In other words “If p, then q” is false exactly when p is true and q is false. In particular, if you want to explain why the possible theorem “If p, then q” is false, you need to come up with a situation where p is true and q is false.

Here’s a typical example of a theorem stated in if-then form.

Theorem 1.3.1. *If an integer n is divisible by* 6*, then n is even.*

Sometimes if-then statements are expressed using “for every” or even just “every”. For example, Theorem 1.3.1 is equivalent to:

Theorem 1.3.2. *Every number divisible by* 6 *is even.*

1.3.4 Direct proofs

A *proof* is a logical explanation of why a theorem is true. When you are called to do a proof in this book, the method you should use is what is often called *direct proof*, though I personally prefer the name *if-then method*. The idea of the if-then method is:

The if-then method: To prove the statement “If p, then q,” **assume** that p is true and use logic to **conclude** that q is true.

In other words, a proof of an if-then statement is a logical explanation why certain assumptions lead to certain conclusions.

For example, the proof of Theorem 1.3.1 should look something like:

Assume: n is divisible by 6.
(blah blah logical deduction blah blah)
Conclude: n is even.

To go one important step further, if we incorporate the definition of divisibility you’ll see in Chapter 2 (see Definition 2.1.4) at the beginning and the end of the proof, we get:

Assume: n is divisible by 6.
That means that $n = 6q$ for some integer q.
(blah blah logical deduction blah blah)
Therefore, $n = 2k$ for some integer k.
Conclude: n is even.

1.3.5 Closure proofs

The most common example of an if-then proof that you'll have to do in this book is a *closure proof.* Closure under an operation is itself an important general idea in algebra, but for concreteness, for now, let's just consider closure under addition. (If you've seen some linear algebra, you should have seen this idea before.)

Definition 1.3.3. To say that a set of numbers S is *closed under addition* means: "If $x, y \in S$, then $x + y \in S$."

Closure under multiplication is similar, replacing $x + y$ with $x \cdot y$, and so on.

Combining the definition of closure under addition (Definition 1.3.3) and the if-then method, we see that the proof of the following theorem:

Theorem 1.3.4. *The set S is closed under addition.*

Looks like this:

Proof. Assume that $x \in S$ and $y \in S$.
(blah blah logic whatever)
So $x + y \in S$. □

To give a bit more detail, when you prove that a set S is closed under an operation, quite often S is defined using set-builder notation (Section 1.3.1). To give a concrete example, suppose you want to prove the following theorem.

Theorem 1.3.5. *The set of even numbers E is closed under addition.*

To prove Theorem 1.3.5, you first have to remember that the even numbers have the following set-builder definition:

$$E = \{n \in \mathbf{Z} \mid n = 2k \text{ for some } k \in \mathbf{Z}\}. \tag{1.3.6}$$

Combining the closure proof structure with the Defining Condition Principle (Section 1.3.1), we get the following proof structure.

Proof. Assume that $x \in E$ and $y \in E$.
By the definition of E, we know that $x = 2k$ for some $k \in \mathbf{Z}$ and $y = 2m$ for some $m \in \mathbf{Z}$.
(blah blah logic whatever)
Therefore, $x + y = 2r$ for some $r \in \mathbf{Z}$, which means that $x + y \in E$. □

You may be wondering why we have to use different variable names on the right-hand sides of $x = 2k$, $y = 2m$, and $x + y = 2r$, even though the definition of E only mentions "$= 2k$ for some $k \in \mathbf{Z}$". The reason is that the condition that defines x being an even number is really just a formal way of saying that x is twice some integer, and the value of that integer will be different for different even numbers. (If you have some programming experience, another way of understanding why we need different variable names here is that the k appearing in the definition of E is a local variable only existing inside the definition, so if you want to refer to "k" globally, you have to give a different name for each case in which it appears.)

In any case, as you'll see, it's definitely worth learning the structure of a closure proof, as you'll use it in several different places in this book.

1.3.6 Set containment and equality

Several times in this book, you'll be asked to prove that two sets are equal. Here's one example of the kind of statement you might have to prove using a *set equality* proof. (This example is both somewhat artificial and a bit overly complicated, to illustrate a number of things at once.)

Theorem 1.3.6. *Let*

$$\begin{aligned} A &= \{n \in \mathbf{Z} \mid n = 6k \textit{ for some } k \in \mathbf{Z}\}, \\ B &= \{n \in \mathbf{Z} \mid n = 10\ell \textit{ for some } \ell \in \mathbf{Z}\}, \\ C &= \{n \in \mathbf{Z} \mid n = 30m \textit{ for some } m \in \mathbf{Z}\}. \end{aligned} \tag{1.3.7}$$

Then $A \cap B = C$.

If you haven't seen much about sets before, or if it's been a while, there's a lot to unpack in Theorem 1.3.6. Here are the definitions you need to understand the statement.

Definition 1.3.7. Let A and B be sets. We define

$$\begin{aligned} A \cap B &= \{x \mid x \in A \text{ and } x \in B\}, \\ A \cup B &= \{x \mid x \in A \text{ or } x \in B\}. \end{aligned} \tag{1.3.8}$$

That is, the *intersection* $A \cap B$ is defined by AND, and the *union* $A \cup B$ is defined by OR.

Definition 1.3.8. To say that a set A is *contained* in another set B, or alternately, that A is a *subset* of B, written $A \subseteq B$, means that every element of A is an element of B. In other words, $A \subseteq B$ means that if x is an element of A, then x is also an element of B. To say that two sets A and B are *equal* means that $A \subseteq B$ and $B \subseteq A$.

The main virtue of Definition 1.3.8 is that it turns set containment into an if-then statement, and set equality into two if-then statements. For example, here's the how the structure of the proof of Theorem 1.3.6 goes on first pass.

Part of the proof of Theorem 1.3.6. First, we show that $A \cap B \subseteq C$. Assume that $x \in A \cap B$.
(using the definitions of A, B, and C, blah blah)
Therefore, $x \in C$.
On the other hand, we also show that $C \subseteq A \cap B$. Assume $x \in C$.
(using the definitions of A, B, and C again, yadda yadda)
Therefore, $x \in A \cap B$. □

If we add in the definition of intersection and the definitions of A, B, and C, we get:

More of the proof of Theorem 1.3.6. First, we show that $A \cap B \subseteq C$. Assume that $x \in A \cap B$. Therefore, $x = 6k$ and $x = 4\ell$ for some $k, \ell \in \mathbf{Z}$.

(apply some number theory)

It follows that $x = 12m$ for some $m \in \mathbf{Z}$, and therefore, that $x \in C$.

On the other hand, we also show that $C \subseteq A \cap B$. Assume $x \in C$. Therefore, $x = 12m$ for some $m \in \mathbf{Z}$.

(using the definitions of A, B, and C again — this is the less difficult direction)

It follows that $x = 6k$ and $x = 4\ell$ for some $k, \ell \in \mathbf{Z}$, and therefore, that $x \in A \cap B$. □

If you've seen some of the fundamental facts about prime numbers before somewhere, try finishing the proof yourself.

Remark 1.3.9. Sometimes students try to do set equality proofs by trying to combine set-builder definitions like those in (1.3.7) directly, without considering individual elements one at a time. That approach doesn't save much time or space, is harder to read than the "one element at a time" approach, and tends to produce more mistakes. So please, stick with one element at a time — you'll thank me later. (Or at least anyone who has to read your proofs will thank me.)

1.3.7 Induction proofs

Let me say right away that you will only rarely be asked to *do* any induction proofs in this book, so in some sense, you don't really need to understand induction to use this book. On the other hand, we'll actually use induction a few times in some key proofs, so we'll give a brief explanation of induction here to avoid giving the impression that there's something mysterious or impenetrable going on with those proofs.

In a nutshell, *induction* is the following axiom, which we assume as part of our background mathematical language.

Axiom 1.3.10 (Induction)**.** Suppose $T(n)$ is a sequence of theorems (mathematical statements) indexed by a variable n, and suppose that:

- (Base case) We can prove that $T(0)$ is true.
- (Induction step) For $n \geq 0$, we can prove that if $T(n)$ is true, then $T(n+1)$ is true.

Then the axiom of induction says that we consider $T(n)$ to be proven for all $n \geq 0$.

Now, once you really understand induction, one of your first reactions might be that there's no reason for induction to be an extra axiom. For if we know that $T(0)$ is true and that $T(n)$ always implies $T(n+1)$, we can then reason that

$$T(0) \Rightarrow T(1) \Rightarrow T(2) \Rightarrow \cdots \Rightarrow T(n) \Rightarrow T(n+1) \Rightarrow \dots, \tag{1.3.9}$$

And so $T(k)$ holds for all nonnegative integers k. Seems obvious when you look at it that way, right? What's all this business about needing an extra axiom?

The catch is that (1.3.9) is actually an infinite proof, if we are to prove $T(k)$ for *all* k, and the rules of math don't allow for infinite proofs, in general. You may therefore find it helpful to think of induction as one particular kind of infinite proof that we allow.

Finally, one way in which induction is often applied, even if you're not that interested in proof, is when we use what is know as an *inductive definition.* To give a concrete example, we can define $n!$ (a.k.a. "n *factorial*") for all integers $n \geq 0$ by declaring that

$$\begin{aligned} 0! &= 1, \\ (n+1)! &= (n+1)n! \quad \text{for } n \geq 0. \end{aligned} \tag{1.3.10}$$

Induction is used here to show that since:

- $0!$ is defined; and
- If $n!$ is defined, then $(n+1)!$ is defined;

we therefore can be assured that $k!$ is defined for all $k \geq 0$.

Note that one immediately useful thing about inductive definitions, and sometimes even proofs by induction, is that you can often think of them as *recursive* algorithms. For example, applying (1.3.10) repeatedly tells us that

$$5! = 5(4!) = 5 \cdot 4(3!) = 5 \cdot 4 \cdot 3(2!) = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 \cdot 1. \tag{1.3.11}$$

This example displays two important features of recursive algorithms. On the one hand, recursive algorithms can be written with very few lines of code; in fact, (1.3.10) is essentially complete code for the factorial function. On the other hand, the actual details of the computation in (1.3.11) show that recursive algorithms can require much more space than the usual iterative algorithms. For a discussion of how to make recursive algorithms efficient, see (??).

1.3.8 Functions, one-to-one, and onto

definition of function

domain and codomain notation

codomain versus range

In Definitions 1.3.11–1.3.13, let X and Y be sets, and let $f : X \to Y$ be a function.

Definition 1.3.11. To say that f is *one-to-one*, or *injective*, means that for $x_1, x_2 \in X$, if $f(x_1) = f(x_2)$, then $x_1 = x_2$.

Definition 1.3.12. To say that f is *onto*, or *surjective*, means that for every $y \in Y$, there exists some $x \in X$ such that $f(x) = y$.

Definition 1.3.13. To say that f is *bijective* means that f is one-to-one and onto.

1.3.9 Invertibility

definition of inverse

next should be better: inverse if and only if bijective.

Theorem 1.3.14. *Let X and Y be sets, and let $f : X \to Y$ be a bijective function. Then we can define a function $f^{-1} : Y \to X$ by saying that for all $y \in Y$,*

$$f^{-1}(y) = \text{the unique } x \in X \text{ such that } f(x) = y. \tag{1.3.12}$$

Furthermore, for all $x \in X$, $f^{-1}(f(x)) = x$, and for all $y \in Y$, $f(f^{-1}(y)) = y$.

We call the above function f^{-1} the *inverse* of f. Note that the last statement of the theorem means that f and f^{-1} reverse each other's effects, or undo each other.

Proof. We first have to show that f^{-1} is well-defined (i.e., defined unambiguously), and the key point is whether the right-hand side of (1.3.12) makes sense. However, on the one hand, since f is onto, there is some $x \in X$ such that $f(x) = y$; and on the other hand, if $f(x_1) = y$ and $f(x_2) = y$ for $x_1, x_2 \in X$, then since f is one-to-one, $x_1 = x_2$, which means that the $x \in X$ such that $f(x) = y$ is unique. It follows that f^{-1} is well-defined.

As for the final statement, on the one hand, if $x \in X$ and $y = f(x)$, then

$$f^{-1}(f(x)) = f^{-1}(y) = x, \tag{1.3.13}$$

since x is the (unique) element of X such that $f(x) = y$. On the other hand, if $y \in Y$ and $f^{-1}(y) = x$, then

$$f(f^{-1}(y)) = f(x) = y, \tag{1.3.14}$$

where the last equality holds by definition of f^{-1}. The theorem follows. □

1.4 Number systems

Every theory course, and especially every algebra course, needs to choose a starting point, and ours is, roughly speaking, that we'll take as given everything you learned up through high school algebra, at least in terms of "facts." (A more advanced abstract algebra course would actually assume *less*, as one important part of such a course is to reconstruct what you learned up through high school on a firmer logical foundation.)

For example, we'll assume you're familiar with the *natural numbers* and the *integers:*

$$\mathbf{N} = \{1, 2, 3, \dots\}, \qquad \mathbf{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}. \tag{1.4.1}$$

Note that we use the convention of starting $\mathbf{N}$ with 1 instead of starting with 0.[*]

You should also be familar with the *rational numbers:*

$$\mathbf{Q} = \left\{ \frac{k}{n} \,\middle|\, k, n \in \mathbf{Z},\ n \neq 0 \right\}. \tag{1.4.2}$$

[*]Apologies to fans of starting with 0 — you gotta choose one of the two conventions, right?

In other words, the rationals **Q** are precisely all numbers formed as the legitimate (thus the condition $n \neq 0$) ratio of two integers. If you're in a fussy mood, you might observe that (1.4.2) is not a definition *per se*, just a list of the numbers within the real numbers that happen to be rational — to which we reply, buckle up, friend, you're going to find this book to be pretty un-fussy by your standards.

Speaking of the real numbers, in this book, we will all agree to pretend that we know what the *real numbers* **R** are. Which is obvious unless you think about it: You've used them all your life, you actually used deep properties about them when you took calculus, they form the number line, they're all possible numbers that can be expressed as a decimal, finite or infinite... sure, that all sounds fine! So we'll just all agree to avoid unpleasant questions like "What is the actual, precise definition of the real numbers?".[†]

Finally, we'll assume that at some point you have seem the *complex numbers* **C**, or at least that you have some vague memory of $i = \sqrt{-1}$. We'll have much more to discuss about complex numbers in Chapter 9.

Again, if anything in this chapter seems unfamilar, or didn't make sense on first reading, don't worry too much — there will be plenty of chances to get to know this "background" material later by using it in other contexts. The important point is that you have now at least seen the material once.

In any case, that's enough throat-clearning. Let's get started! As they tell you just before you go off an island adventure in *Animal Crossing*:

> *Go catch some bees and chop some trees!*
>
> — Wilbur (the Dodo Airlines pilot), *Animal Crossing New Horizons*

[†]This question is the heart of introductory analysis, one of the more difficult classes in the standard undergraduate math curriculum.

Chapter 2

Faster: The Euclidean Algorithm

We might call [Euclid's algorithm] the granddaddy of all algorithms, because it is the oldest nontrivial algorithm that has survived to the present day.

— *The Art of Computer Programming, vol. 2*, Donald E. Knuth

2.1 Divisibility

Usually we'll start each chapter on applications with a motivating problem from the real world. However, we're still warming up here, so instead, let's start with what might look like an easy question.

Ask Yourself 2.1.1. What are all of the divisors of 12?

Ask Yourself 2.1.2. No, seriously, did you really think about Ask Yourself 2.1.1? Maybe even write down an answer (at least in your head), and then come back to reading this book.

Most probably, you answered 1, 2, 3, 4, 6, 12, the correct grade-school answer. However, here are some answers to Ask Yourself 2.1.1 that you might not have thought about.

- First, since $12 = (-4)(-3)$, if you allow the use of negative numbers, -4 divides 12.
- Remember $i = \sqrt{-1}$? If we allow the use of i, then since $12 = (1-i)(6+6i)$, $1-i$ divides 12.
- Furthermore, since $12 = (17)\left(\frac{12}{17}\right)$, if you allow the use of rational numbers, 17 divides 12.
- Most outrageously, since $12 = (\pi)\left(\frac{12}{\pi}\right)$, π divides 12; in fact, if you allow the use of real numbers, *any* nonzero real number divides 12.

In short, Ask Yourself 2.1.1 is maybe not quite as straightforward as it might look, because the question doesn't specify which numbers we're allowed to use. In fact, to get an unambiguous answer to many of the questions we consider in this book, we first need to specify which numbers we're allowed to use in our answer. We therefore come to the following idea.

Not a Definition 2.1.3. Suppose R is some system of numbers like $\mathbf{Z}$, $\mathbf{Q}$, $\mathbf{R}$, or $\mathbf{C}$. To say that we are working in the *ring* R means that we are allowed to use numbers in R, and only numbers in R, in our computations, explanations, and so on. Phrases like "over the ring R" have a similar meaning.

Note that Not a Definition 2.1.3 is too vague to be a real mathematical definition; for example, what does "system of numbers" mean, exactly? However, I promised to avoid abstract nonsense until it's absolutely necessary, so we'll stick with this non-definition until Chapter 4. For now, the most important thing is to absorb the idea of a ring as the set of all allowable numbers in a given situation. It may also be helpful to think of the word "ring" in the sense of a boxing ring or wrestling ring: That is, when we work in the ring R, R denotes the arena in which the game is played.[*]

Anyway, now that you have at least a vague idea of what a ring is, here's a precise definition of divisibility over the integers.

Definition 2.1.4. To say that an integer d *divides* an integer n in $\mathbf{Z}$, or alternately, that d is a *divisor* of n, means that $n = qd$ for some $q \in \mathbf{Z}$ (i.e., some integer q). When the context is clear, we omit "in $\mathbf{Z}$" and just say that a divides n.

In other words, Definition 2.1.4 establishes the convention that for now, when we talk about divisors, we work in the ring $\mathbf{Z}$. Be warned, soon enough we'll look at divisibility in other contexts, and use other definitions! But again, for now, we'll stick with the integers.

Example 2.1.5. Returning to Ask Yourself 2.1.1, under Definition 2.1.4, the divisors of 12, in order from least to greatest, are: -12, -6, -4, -3, -2, -1, 1, 2, 3, 4, 6, and 12.

I'm sure you find the answer in Example 2.1.5 a bit unsatisfying, and you may even be asking yourself (or your instructor): "Can't we just list the positive parts, or at least ± 1, ± 2, …, ± 12?" And you're right; more generally, if d divides n, then $-d$ must also divide n, and similarly, if d divides n, then d must divide $-n$ (Problem 2.1.1). In other words, a number and its negative have exactly the same divisibility properties. We therefore have the following term to formalize that idea.

Definition 2.1.6. To say that integers a and b are *associates* means that $a = \pm b$; equivalently, we say that a and b are the same *up to associates.*

You might object that Definition 2.1.6 is just a very complicated way to say "plus or minus". And so far, you'd be right! But later on, we'll see that a suitably generalized definition of associate is useful when considering divisibility of polynomials, for example.

[*]This is a bit of a mathematical dad joke here; for the actual reason behind the name "ring", see Remark 4.2.5.

Now, I've promised in both the Introduction and in Chapter 1 that proofs are not the focus of this book. However, there's no better way to understand how to use a definition than to use that definition in a proof, so the problems for this section all ask you to do relatively short proofs using the definition of divisibility. See Section 1.3, especially Section 1.3.4, for more about how to do proofs like these.

Problems

2.1.1. Suppose d and n are integers.

(a) Prove that if d divides n, then $-d$ divides n. (In other words, assume that d divides n, and use logic to conclude that $-d$ divides n; see Section 1.3.4 for details of how that works.)

(b) Prove that if d divides n, then d divides $-n$.

2.1.2. Suppose d, a, and b are integers. Prove that if d divides a and d divides b, then d divides $a + b$.

2.1.3. Suppose d, a, and b are integers. Prove that if d divides a, then d divides ab.

2.1.4. Suppose d, a, and b are integers. Prove that if d divides a and a divides b, then d divides b.

2.1.5. What are all of the divisors of 0? Explain.

2.2 Greatest common divisors

To recap, the point of Section 2.1 is that we now have the language we need to state the main problem of this chapter precisely. (Though to give a real-life perspective, sometimes stating the problem precisely is half the battle.)

Definition 2.2.1. For integers d, a, and b, to say that d is a *common divisor* of a and b means that d divides a and d divides b.

Since Defintion 2.2.1 marks our first compound definition, let's pause to stop and think about that for a while. (As you go on in this book, or even when you study more math later, try to build the habit of doing a similar self-reflection whenever you encounter a definition that refers back to another definition.)

Ask Yourself 2.2.2. What do you get when you combine Definitions 2.1.4 and 2.2.1? What does the definition of common divisor look like if you have to start from scratch, instead of being able to use Definitions 2.1.4? In other words, if you assume that d is a common divisor of a and b, what do you know about d, a, and b?

Definition 2.2.3. For integers a and b, at least one of which is not 0, the *greatest common divisor*, or *GCD*, of a and b is exactly what it sounds like: the greatest integer d that is a common divisor of a and b. We denote the greatest common divisor of a and b by the symbol $\gcd(a, b)$.

Ask Yourself 2.2.4. When you encounter a new definition, always try small, weird, and exceptional cases. What is $\gcd(6, 10)$? Suppose a is a nonzero integer. What is $\gcd(a, 1)$? What is $\gcd(a, 0)$? How about $\gcd(a, a)$? Why don't we give a definition for $\gcd(0, 0)$? (See Problem 2.1.5.) Can $\gcd(a, b)$ ever be negative? Zero? (See Problem 2.2.1.)

Now, Definition 2.2.3 is actually unambiguous in the precise mathematical language we used, but in terms of ordinary language, there's an interesting ambiguity related to the following question:

Ask Yourself 2.2.5. Silly question: What's bigger, -10000 or 3?

The thing is, Ask Yourself 2.2.5 is not entirely silly: If we account for signs, as we do in Definition 2.2.3, then $-10000 \leq 3$, and 3 is bigger, but in ordinary language, we might well say that -10000 is bigger because $|-10000| > |3|$. It therefore falls to us to decide which meaning of bigger we're using in Definition 2.2.3.

So, because it turns out to be more consistent with what we'll do later:

In Definition 2.2.3, we use "greatest" to mean "largest absolute value".

This seemingly picky point actually has noticeable consqueneces! Most notably, it's equally correct to say that $\gcd(6, 10) = 2$ or $\gcd(6, 10) = -2$; more generally, "the" GCD of two integers is only determined up to associates (Definition 2.1.6), i.e., up to $\pm$. In fact, we'll see momentarily that this ambiguity comes up naturally, in that the most efficient algorithm we'll discuss for finding $\gcd(a, b)$ will sometimes produce a negative number as an answer.

Remark 2.2.6. I know, I promised that this book would focus on things you can use to make money, so what's with all the fussiness about definitions? The catch is, as an applied mathematician, you make money not just by applying the algorithm, or even coding the algorithm, but also by understanding why it's OK that your GCD algorithm gives an answer of -2 instead of 2.

In any case, at last, we come to the motivating problem of this chapter.

Motivating Problem 2.2.7. Given nonzero integers a and b how can we efficiently compute $\gcd(a, b)$?

One of your first questions about Motivating Problem 2.2.7 might be, how can you make money off of something you might have learned how to do in grade school? Well, the key word in Motivating Problem 2.2.7 is *efficiently*, and to dig into that idea, we start with the following question.

Ask Yourself 2.2.8. What is $\gcd(24, 105)$? More importantly, how did you figure that out, or in other words, what *algorithm* did you use?

The reason to consider Ask Yourself 2.2.8 is that Motivating Problem 2.2.7 isn't very interesting if you just want *some* algorithm for computing $\gcd(a, b)$, and you don't care how fast it is. For example, one method is:

Naive Algorithm 2.2.9. Let a and b be positive integers.

1. Make an ordered list of positive divisors of a.

2. Check which of those divisors of a also divides b, starting from the largest divisor and going downwards.

The first common divisor found in step 2 will be $\gcd(a, b)$.

No problem! Unless, of course, you actually want to compute something practical: Imagine using Naive Algorithm 2.2.9 to compute $\gcd(1723729, 8675309)$, let alone the 300-digit computations you can actually use to make money. The problem with Algorithm 2.2.9 is that the amount of time required grows too rapidly as a function of the size of a and b. More precisely, here's an upper bound (and maybe even a reasonable estimate) for the amount of time required. Suppose $a, b \leq n$.

1. In Step 1 of Algorithm 2.2.9, one way to find all positive divisors of a is to consider all d from 1 to a and divide a by d with remainder. This could take up to n divisions.

2. Then for Step 2, we do the same thing, except letting d go down the list of divisors of a and dividing d into b. There are no more than n divisors of a, so again we have no more than n divisions.

Therefore, in total, Naive Algorithm 2.2.9 takes at most $2n$ steps. It turns out not to be super-hard to improve that estimate using only elementary facts about divisors (see Problem 2.2.3). The real challenge comes in getting an *exponential* speedup, or more precisely:

Motivating Problem 2.2.10. Suppose a, b are positive integers $\leq n$. Can we find an algorithm for computing $\gcd(a, b)$ that takes fewer than $C \log n$ steps, for some constant C?

Now, you've taken enough math courses in your life to know that I probably wouldn't ask a leading question like Motivating Problem 2.2.10 without a good answer, and that's coming. For now, the thing to realize about the appearance of $\log n$ in Motivating Problem 2.2.10 is that you should think of $\log n$ as being roughly equivalent to the number of digits of n — a *much* smaller rate of growth. See Section 2.6 for a more detailed and precise discussion.

Problems

2.2.1. Let a and b be nonzero integers. Explain why $\gcd(a, b)$ must be positive.

2.2.2. Let a be a fixed positive integer.

(a) Write out the (positive) factors of $a = 24$ in increasing order, and do the same for $a = 45$ and $a = 36$. What symmetry do you see around the halfway point of each list?

(b) Suppose d and q are positive integers, $a = dq$, and $d \leq q$ (i.e., d is the smaller of the two factors of a). Explain how you can be sure (i.e, prove) that $d \leq \sqrt{a}$. (Suggestion: What would happen if that were false?)

(c) Keeping part (a) in mind, explain how you can group the positive divisors of a into pairs (d_1, d_2) with $d_1 \leq d_2$ such that every divisor of a appears in exactly one such pair. Explain why it follows that a has no more than $2\sqrt{a}$ positive divisors.

2.2.3. The goal of this problem is for you to create a Modified Naive Algorithm (a refinement of 2.2.9) that requires fewer steps to compute $\gcd(a, b)$ than indicated by the analysis given above.

Suppose N is the size of the largest possible integer that can be accepted as input. Use the results of Problem 2.2.2 to figure out a Modified Naive Algorithm such that if $a, b \leq N$, then the number of divisions required to compute $\gcd(a, b)$ is no more than $C\sqrt{N}$, where C is some constant.

2.2.4. Let $N > 1$ be an integer. Recall from your K–12 education that N is *prime* when its only positive divisors are 1 and N. Use the results of Problem 2.2.2 to figure out a Naive Algorithm that requires no more than $C\sqrt{N}$ divisions to determine whether N is prime.[†]

2.2.5. Let $N > 1$ be an integer. The goal of this problem is for you to create a Naive Factorization Algorithm that gives the factorization of N into primes.

(a) Suppose $N = qd$, where q and d are positive integers, neither equal to 1. Explain why both q and d must be $\leq N/2$.

(b) Suppose N is *not* prime, and suppose d is the smallest possible divisor of N such that $d > 1$. Explain why d must be prime.

(c) Use the results of Problem 2.2.2 to figure out a Naive Factorization Algorithm such that if $T(N)$ is the number of divisions required to run the algorithm on the input N, then

$$T(N) \leq C\sqrt{N} + T(N/2). \tag{2.2.1}$$

(d) Use (2.2.1) and induction on N to prove that $T(N) \leq C \log_2(N)\sqrt{N}$.

[†]Professional-grade primality tests are much faster; for example, the AKS deterministic primality test requires no more than $C(\log N)^{12}$ operations, an *exponential* speedup compared to the naive algorithm described here, and algorithms that use randomness but work with very high probability are even faster.

2.3 Division with remainder

The first building block of the Euclidean Algorithm for computing $\gcd(a, b)$ is a careful consideration of something you've known since grade school, namely, division with remainder. The details are important enough that we'll express them as a theorem.

Theorem 2.3.1 (Division Theorem)**.** *Let a and d be positive integers. There exist unique nonnegative integers q and r such that*

$$a = dq + r, \qquad \textit{with } 0 \leq r < d. \tag{2.3.1}$$

The word "unique" in the statement of the Division Theorem 2.3.1 means that there is only one choice of nonnegative integers q and r that makes (2.3.1) true. Note that "$0 \leq r < d$," or in other words, the fact that the remainder r is *strictly* less than the divisor d, is an important part of (2.3.1); in fact, if we change that condition even slightly, the theorem fails (Problem 2.3.1).

The Division Theorem is important enough that we'll prove it twice, as each proof generalizes in a different way, giving two different new algorithms. (Again, I promise you, the theory is really the moneymaking stuff!)

Traditional proof. Start off with some initial guess for q with $a - qd \geq 0$ ($q = 0$ works). If $r < d$, then we've found q, r that makes (2.3.1) true; otherwise, increase q by 1, which is still OK, because the new remainder will be

$$a - (q + 1)d = a - qd - d = r - d \geq 0. \tag{2.3.2}$$

We can't go on increasing q forever, since there is some q such that $qd > a$, and we always preserve $r \geq 0$, so we'll eventually get (2.3.1) to be true.

For a proof of uniqueness, see Problem 2.3.3. □

Note that this "traditional proof" is really an induction argument (Section 1.3.7); if you're familiar with induction, you might want to try rewriting the proof that way (Problem 2.3.2). Practically speaking, the proof is also a non-practical algorithm for doing long division (keep guessing a bigger q), illustrating the general principle that many induction arguments are secretly recursive algorithms.

Our nontraditional proof uses the *floor* function, which you may have seen in calculus as an example of a discontinuous function.

Definition 2.3.2. For a real number x, $\lfloor x \rfloor$, or the *floor* of x, is the greatest integer less than or equal to x.

For example, $\lfloor 2.7 \rfloor = 2$, $\lfloor 13 \rfloor = 13$, and $\lfloor -5.3 \rfloor = -6$.

Nontraditional proof. Let $q = \left\lfloor \dfrac{a}{d} \right\rfloor$. By the definition of floor (Definition 2.3.2), we know that

$$q \leq \frac{a}{d} < q + 1, \tag{2.3.3}$$

so multiplying by d and subtracting qd, we get

$$0 \leq a - qd < d. \tag{2.3.4}$$

Letting $r = a - qd$ yields (2.3.1), and uniqueness again follows from Problem 2.3.3. □

Remark 2.3.3. If the nontraditional proof made more sense to you, you may be wondering: Why would anyone bother with the induction-ish nonsense of the traditional proof? For logical sticklers, the answer is that the fact that the floor function is well-defined (has an unambiguous meaning) *also* relies on induction, or rather, its logical equivlent, the Well-Ordering Principle. See, for example, Ross (specific ref to be added).

We'll generalize our traditional proof later (Section 3.3), but right now, we'll use our nontraditional proof to obtain the following generalization of the Division Theorem.

Theorem 2.3.4 (Signed Division Theorem)**.** *Let a and d be nonzero integers. There exist integers q and r such that*

$$a = dq + r, \qquad \text{with } |r| \leq \frac{|d|}{2}. \tag{2.3.5}$$

The idea behind the Signed Division Theorem is to replace the floor function with the *rounding function.* There are actually many standard ways to define rounding off to the nearest integer that differ in how they deal with half-integers, especially negative ones, so we'll just assume that we have some particular definition of $\lfloor x \rceil$ such that $\lfloor x \rceil$ is an integer and

$$x - 0.5 \leq \lfloor x \rceil \leq x + 0.5, \tag{2.3.6}$$

much as you'd expect.

Proof. Assume $d > 0$ and let $q = \left\lfloor \frac{a}{d} \right\rceil$ (i.e., $\frac{a}{d}$ rounded to the nearest integer). By (2.3.6),

$$\frac{a}{d} - 0.5 \leq q \leq \frac{a}{d} + 0.5, \tag{2.3.7}$$

so multiplying by d and subtracting a, we get

$$-\frac{d}{2} \leq qd - a \leq \frac{d}{2}. \tag{2.3.8}$$

Letting $r = a - qd$ yields $|r| \leq \frac{d}{2}$, as desired. When $d < 0$, the inequalities in (2.3.8) flip, but the result in terms of absolute values is the same, and the theorem follows in general. □

Since you've known how to do regular long division since grade school, but you probably haven't thought much about how to do division with smallest (signed) remainders, here's one method for dividing n by some positive d and getting a remainder no larger than $\frac{d}{2}$.

1. Do regular long division of n by d, to get $n = qd + r$ as usual.

2. If $r \leq \frac{d}{2}$ already, good, you're done. Otherwise, if $r > \frac{d}{2}$, add 1 to the quotient q, to get

$$n = (q+1)d + r'. \tag{2.3.9}$$

Then since $\frac{d}{2} < r < d$, the new remainder $r' = r - d$ will be between $-\frac{d}{2}$ and 0, so we get $|r'| \leq \frac{d}{2}$, as desired.

Problems

2.3.1. Give an example of positive integers a and d where there are two different pairs of nonnegative integers q_1, r_1 and q_2, r_2 such that $a = dq_i + r_i$ for both $i = 1$ and $i = 2$.

2.3.2. (If you're familiar with induction.) Rewrite the "traditional proof" of the Division Theorem as an induction argumennt.

2.3.3. (*Proves Theorem 2.3.1*) Suppose a and d are positive integers. The goal of this problem is to prove the uniqueness part of Theorem 2.3.1 by showing that if we seemingly have two possibilities q_1, r_1 and q_2, r_2 for the quotient and remainder in Theorem 2.3.1, then in reality, those two possibilities actually have to be the same, i.e., $q_1 = q_2$ and $r_1 = r_2$. (Not the most obvious way to prove there's only one answer, but I hope it makes sense once you hear the idea.)

(a) Suppose r_1 and r_2 are integers such that $0 \leq r_1 \leq r_2 < d$. What is the largest possible value that $r_2 - r_1$ could have? (Suggestion: Try drawing a picture of r_1 and r_2 on the number line.)

(b) Now suppose that q_i and r_i ($i = 1, 2$) are integers such that

$$a = q_1 d + r_1, \qquad a = q_2 d + r_2, \tag{2.3.10}$$

with $0 \leq r_1, r_2 < d$. Prove that $r_1 = r_2$ and $q_1 = q_2$. (Suggestion: By switching q_1, r_1 with q_2, r_2 if necessary, you can assume that $r_1 \leq r_2$.)

2.3.4. In the Signed Division Theorem 2.3.4, the choice of quotient q and remainder r aren't unique, but they almost are.

(a) Find integers a and d such that there are at least two possible choices of quotient q and remainder r that satisfy the condition (2.3.5).

(b) When do you get non-unique quotient and remainder in the Signed Division Algorithm? Find a condition on a and d that describes exactly when this happens.

(c) Prove your assertion in part (b). That is, state a theorem like "The Signed Division Algorithm has a unique quotient and remainder unless..." and prove your theorem.

2.4 The Euclidean Algorithm

At last, after much throat-clearning, we come to the *Euclidean Algorithm* for computing $\gcd(a, b)$.

Algorithm 2.4.1 (The Euclidean Algorithm)**.** Suppose a and b are positive integers and $a > b$.

1. *Initialize.* Let $r_{-1} = a$ and $r_0 = b$.

2. *Main loop.* For $i = 1, 2, \ldots$, apply the Division Theorem to divide r_{i-2} by r_{i-1} with quotient q_i and remainder r_i, or in other words,

$$r_{i-2} = q_i r_{i-1} + r_i \qquad \text{with } 0 \le r_i < r_{i-1}. \tag{2.4.1}$$

   Stop, after N divisions, as soon as you get a remainder $r_N = 0$.

3. *Claim.* The last nonzero remainder r_{N-1} is exactly $\gcd(a, b)$.

As we will often do in this book, we end the above algorithms with a “claim” as to the correctness of our answer. We use the word “claim” to indicate that it may not be clear to you why that should be the correct answer. (Indeed, it probably shouldn’t be clear to you, if you’re being skeptical and honest.) Therefore, to ensure that the algorithm really does work, we’ll prove it as a theorem. Again, rest assured, this is *not* the sort of thing this book will ask you to produce — the proof is not at all obvious! But I do hope that as an informed consumer of abstract algebra, after some work, you’ll be able at least to *understand* the proof.

First, though, it may help to lay out the algorithm visually and do an example or two. If we write out the iterations of the Euclidean Algorithm 2.4.1 in order, we get something like:

$$\begin{aligned}
r_{-1} &= q_1 r_0 + r_1 & &(0 \le r_1 < r_0)\\
r_0 &= q_2 r_1 + r_2 & &(0 \le r_2 < r_1)\\
r_1 &= q_3 r_2 + r_3 & &(0 \le r_3 < r_2)\\
&\vdots\\
r_{N-3} &= q_{N-1} r_{N-2} + r_{N-1} & &(0 \le r_{N-1} < r_{N-2})\\
r_{N-2} &= q_N r_{N-1}
\end{aligned} \tag{2.4.2}$$

Notice how each particular remainder r_n moves down and to the left at each stage, something that is useful to remember when doing the algorithm by hand.

Example 2.4.2. To give a moderate-sized example, applying the Euclidean Algorithm to gcd(441, 192), we get:

$$\begin{aligned} 441 &= 2(192) + 57 \\ 192 &= 3(57) + 21 \\ 57 &= 2(21) + 15 \\ 21 &= 1(15) + 6 \\ 15 &= 2(6) + 3 \\ 6 &= 2(3) \end{aligned} \tag{2.4.3}$$

Since 3 is the last nonzero remainder, $\gcd(441, 192) = 3$, and the algorithm finishes in 6 steps.

Ask Yourself 2.4.3. Make up your own examples by randomly choosing 3-digit positive integers a and b and applying the Euclidean Algorithm. (One thing this exercise shows is that it's not easy to come up with an example where the algorithm takes more than a few steps!)

Now that you've had the chance to get better acquainted with the Euclidean Algorithm, it's time to prove that it works as advertised. In particular, we need to explain how we can be sure the algorithm actually stops!

Theorem 2.4.4. *The Euclidean Algorithm 2.4.1 terminates after finitely many steps, and the result is actually equal to* $\gcd(a, b)$.

Proof. Keeping the notation of the Euclidean Algorithm 2.4.1, since each r_n is a nonnegative integer and $r_n < r_{n-1}$, we see that the the number of steps in the Euclidean Algorithm is bounded above by $r_0 = b$, so it will eventually stop. (We'll get a much better speed estimate in Section 2.6.)

For the correctness of the answer, let d be a common divisor of a and b. Starting from the top of (2.4.2), since d divides both $a = r_{-1}$ and $b = r_0$, and

$$r_1 = r_{-1} - q_1 r_0, \tag{2.4.4}$$

we see that d also divides r_1, by Problems 2.1.2 and 2.1.3. Next, since d divides both r_0 and r_1, d must divide r_2. Continuing all the way down (2.4.2), we see that d divides r_{N-1}. In particular, $d \leq r_{N-1}$, which means that r_{N-1} is greater than or equal to any common divisor of a and b.

On the other hand, let $c = r_{N-1}$. Starting from the bottom of (2.4.2), we first see that c divides r_{N-2}. (By the definition of divisibility!) Moving up a line, since c divides r_{N-2} and $r_{N-1} = c$, we see that c also divides r_{N-3}, again by Problems 2.1.2 and 2.1.3. Continuing up to the top, we eventually see that c divides both $r_0 = b$ and $r_{-1} = a$, or in other words, c is a common divisor of a and b. Therefore, since c is also greater than or equal to any common divisor of a and b, $c = \gcd(a, b)$. □

Not a thing that makes money directly, but I can't resist pointing out one non-obvious fact that follows immediately from the proof of Theorem 2.4.4.

Corollary 2.4.5. *For positive integers a and b, any common divisor of a and b is also a divisor of* $\gcd(a, b)$*.* □

Now, experience shows that the Euclidean Algorithm is fast, and we'll quantify that statement later in Section 2.6. However, you should always look to make your algorithnm faster — witness the following variation.

Algorithm 2.4.6 (The Signed Euclidean Algorithm)**.** Suppose a and b are nonzero integers and $|a| > |b|$.

1. *Initialize.* Let $r_{-1} = a$ and $r_0 = b$.
2. *Main loop.* For $i = 1, 2, \ldots$, apply the Signed Division Theorem to divide r_{i-2} by r_{i-1} with quotient q_i and remainder r_i, or in other words,
$$r_{i-2} = q_i r_{i-1} + r_i \qquad \text{with } 0 \le |r_i| \le \frac{|r_{i-1}|}{2}. \tag{2.4.5}$$
Stop, after N divisions, as soon as you get a remainder $r_N = 0$.
3. *Claim.* The last nonzero remainder r_{N-1} is exactly $\gcd(a, b)$.

In other words, the Signed Euclidean Algorithm is exactly the same as the Euclidean Algorithm, except we use smallest possible remainders instead of standard nonnegative remainders. (For hand calculations, see also the discsussion at the end of Section 2.3.)

Example 2.4.7. Applying the Signed Euclidean Algorithm to $\gcd(441, 192)$, we get:

$$\begin{aligned} 441 &= 2(192) + 57 \\ 192 &= 3(57) + 21 \\ 57 &= 3(21) + (-6) \\ 21 &= (-3)(-6) + 3 \\ -6 &= (-2)(3) \end{aligned} \tag{2.4.6}$$

Compare Example 2.4.2.

Though Example 2.4.7 is not much faster than Example 2.4.2, Kronecker showed (in the 1800s!) that the Signed Euclidean Algorithm is always at least as fast as the standard one [**?**], so hey, why not, right? The worst-case time required is also slightly easier to analyze for the Signed Euclidean Algorithm; see Section 2.6. In any case, we note for the record that thing does, indeed, work as advertised.

Theorem 2.4.8. *The Signed Euclidean Algorithm 2.4.6 terminates after finitely many steps, and the result is actually equal to* $\gcd(a, b)$*.*

Proof. The proof is almost the same as the proof of Theorem 2.4.4, so it's left to you; see Problem 2.4.3. □

Problems

2.4.1. Use the Euclidean Algorithm to compute the following gcd's.

(a) $\gcd(135, 85)$
(b) $\gcd(1047, 470)$
(c) $\gcd(1615, 1410)$
(d) $\gcd(1502, 586)$
(e) $\gcd(23009, 19670)$
(f) $\gcd(50739, 16301)$

2.4.2. Same as Problem 2.4.1, but use the Signed Euclidean Algorithm 2.4.6.

2.4.3. Modify the proof of Theorem 2.4.4 to obtain a proof of Theorem 2.4.8. Does anything need to be changed at all? Where, if anywhere, does the original proof rely on the fact that all of the numbers involved are nonnegative? Look for places where you have to use absolute values to compare sizes.

2.5 Bezout's identity and Euclean Rewriting

We pause here for what might appear right now to be a digression, but turns out to be a very useful calculation. (If you want to look ahead for some motivation, see Section 3.1.)

Theorem 2.5.1 (Bezout's Identity). *Let a and b be nonzero integers. The equation*

$$ax + by = \gcd(a, b) \tag{2.5.1}$$

has a solution $x, y \in \mathbf{Z}$.

Proof. We prove Bezout's identity by providing an algorithm to compute one such solution $x, y \in \mathbf{Z}$. Retaining the notation of the Euclidean Algorithm 2.4.1, we see that if we define an *integer linear combination* of a and b to be a number of the form $ax + by$ for some $x, y \in \mathbf{Z}$, then our goal is to show that $r_{N-1} = \gcd(a, b)$ is an integer linear combination of a and b.

To start, note that $r_{-1} = a$ and $r_0 = b$ are each integer linear combinations of a and b, and note that we can rewrite the first equation of the Euclidean Algorithm (see (2.4.2)) as

$$r_1 = r_{-1} - q_1 r_0. \tag{2.5.2}$$

Substituting $r_{-1} = a$ and $r_0 = b$, and combining coefficients on the right-hand side of (2.5.2), we see that r_1 is also an integer linear combination of a and b. By the same reasoning, since r_0 and r_1 are integer linear combinations of a and b, and the second equation of (2.4.2) can be rewritten as

$$r_2 = r_0 - q_2 r_1, \tag{2.5.3}$$

we see that r_2 is an integer linear combination of a and b. Continuing similarly down the equations in (2.4.2), we eventually get that $r_{N-1} = \gcd(a, b)$ is an integer linear combination of a and b, and the theorem follows. □

We call the algorithm in the proof of Bezout's identity *Euclidean Rewriting*. (Other authors often use the name *Extended Euclidean Algorithm* to describe an equivalent algorithm.) To write out that algorithm explicitly:

Algorithm 2.5.2 (Euclidean Rewriting)**.** Suppose a and b are positive integers. To solve the equation $ax + by = \gcd(a, b)$ for x and y:

1. Perform the Euclidean Algorithm (Algorithm 2.4.1) to calculate $\gcd(a, b)$. We use the notation of Algorithm 2.4.1 in the rest of what follows.
2. Rewrite each step of the Euclidean Algorithm in the form
$$r_i = r_{i-2} - q_i r_{i-1}. \tag{2.5.4}$$
3. For i going from 1 to $N - 1$, since r_{i-2} and r_{i-1} have already been expressed as integer linear combinations of a and b, use (2.5.4) to rewrite r_i as an integer linear combination of a and b. (Note that in the first step, we use the fact that $r_{-1} = a$ and $r_0 = b$.)

Example 2.5.3. We use Euclidean Rewriting to find an integer solution to $34x + 25y = \gcd(34, 25)$. First off, the Euclidean Algorithm gives:

$$\begin{aligned} 34 &= 1(25) + 9 \\ 25 &= 2(9) + 7 \\ 9 &= 1(7) + 2 \\ 7 &= 3(2) + 1 \\ 2 &= 2(1), \end{aligned} \tag{2.5.5}$$

so $\gcd(34, 25) = 1$.

Next, we rearrange (2.5.5) to solve for the remainder in each step except the last, where there is no remainder:

$$\begin{aligned} 9 &= 34 - 1(25) \\ 7 &= 25 - 2(9) \\ 2 &= 9 - 1(7) \\ 1 &= 7 - 3(2). \end{aligned} \tag{2.5.6}$$

Rewriting (2.5.6) from the top, starting with $34 = a$ and $25 = b$, we then get:

$$\begin{aligned} 9 &= 34 - 1(25) = a - b \\ 7 &= 25 - 2(9) = b - 2(a - b) = 3b - 2a \\ 2 &= 9 - 1(7) = (a - b) - 1(3b - 2a) = 3a - 4b \\ 1 &= 7 - 3(2) = (3b - 2a) - 3(3a - 4b) = 15b - 11a. \end{aligned} \tag{2.5.7}$$

So our final answer is $x = -11$, $y = 15$; or in other words, $-11(34) + 15(25) = 1$. (Check this!)

Finally, the following consequence of Bezout's Identity will also be useful later.

Corollary 2.5.4. *Let a and b be nonzero integers. For $c \in \mathbf{Z}$, the equation*

$$ax + by = c \tag{2.5.8}$$

has a solution $x, y \in \mathbf{Z}$ if and only if $\gcd(a, b)$ divides c.

Proof. Let $d = \gcd(a, b)$. On the one hand, if $ax + by = c$, since d divides both a and b, it follows by Problems 2.1.2 and 2.1.3 that d also divides c. On the other hand, suppose d divides c. Then since $c = dq$ for some $q \in \mathbf{Z}$, and $ax + by = d$ for some $x, y \in \mathbf{Z}$ (Bezout's Identity), we have that $a(xq) + b(yq) = c$. □

In any case, to recap, the key takeaway from this section is:

> To find integer solutions x, y to $ax + by = c$, use the Euclidean Algorithm and Euclidean Rewriting.

Problems

2.5.1. For the following pairs of integers a, b, use Euclidean Rewriting to solve the equation $ax + by = \gcd(a, b)$.

(a) $a = 161$, $b = 70$.

(b) $a = 78$, $b = 53$.

(c) $a = 58$, $b = 51$.

(d) $a = 169$, $b = 125$.

(e) $a = 79$, $b = 56$.

(f) $a = 510$, $b = 208$.

2.6 A crash course in complexity

At this point, you may be (rightfully) saying, "Well, this is all well and good, with the history and the algorithm and whatever, but you promised me that I could make some money here, pal!" And in this section, we'll finally get to that, but we have to introduce one last big idea.

Definition 2.6.1. The *complexity* of an algorithm is the (estimated) time or space that an algorithm needs to finish, given an input of size n. Often, this description comes in the form of a *worst-case time estimate* $T(n)$, that is, a function $T(n)$ such that, given an input of size n, the algorithm is guaranteed to finish within $T(n)$ steps (though possibly sooner).

Complexity is important to us because many, or maybe most, of the problems we consider can be solved by easier methods that are too slow to work in practice. (Remember the grade-school GCD algorithms from Ask Yourself 2.2.8?) What makes the algorithms we discuss money-making is the fact that they are faster, and often much faster, than easier methods. We can sometimes measure this kind of speed-up by doing computer experiments, but it often helps to have some conceptual way to discuss the speed of an algorithm. To be precise, the following terminology can be used to give both a quantitative and a qualitative description of "worst-case time" function $T(n)$.

Definition 2.6.2. Let $T(n)$ and $f(n)$ be real-valued functions with domain the natural numbers $\mathbf{N}$ (in fancy function notation, $T : \mathbf{N} \to \mathbf{R}$ and $f : \mathbf{N} \to \mathbf{R}$). To say that $T(n) = O(f(n))$ means that there exists some constant C such that $T(n) \leq Cf(n)$ for all $n \in \mathbf{N}$. The notation $O(f(n))$ is known as *big O notation.*

To make big O notation useful, we need to understand some standard "comparison functions" $f(n)$ that we can use to describe our worst-case time functions $T(n)$. More specifically, the following idea helps in sorting out the relative sizes of comparison functions.

Definition 2.6.3. Suppose $f(n)$ and $g(n)$ are real-valued functions on n. To say that $f(n) << g(n)$ means that

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0. \tag{2.6.1}$$

We also say that $g(n)$ *dominates* $f(n)$ *asymptotically.*

If you don't remember exactly what $\lim\limits_{n\to\infty}$ means, the point of (2.6.1) is really just that as n gets very large, $g(n)$ is far bigger than $f(n)$. With that mind, we come to the following theorem from calculus[‡], which we state without proof.

Theorem 2.6.4 (The Asymptotics Theorem). *For fixed constants $C > 0$, $0 < p < q$, and $1 < a < b$, we have that*

$$C << \log n << n^p << n^q << a^n << b^n << n!. \tag{2.6.2}$$

In particular, constants are dominated by logs are dominated by powers are dominated by exponentials are dominated by factorials.

In other words, in terms of growth as n approaches ∞, logarithmic growth ($\log n$) is much smaller than polynomial growth (n^p) is much smaller than exponential growth (a^n) is much smaller than combinatorial growth ($n!$). An algorithm with worst-case time $T(n) = O(\log n)$ is called a *logarithmic-time* algorithm, and *polynomial-time*, *exponential-time*, and *combinatorial-time* algorithms are defined similarly.

To give an idea of which times are useful in practice, exponential-time algorithms are generally regarded as hopeless (in terms of "Will this finish before we all die?") and combinatorial-time algorithms are *really* hopeless ("Will this finish before the universe

[‡]Or really, analysis, the theory of calculus, which explains why you may not have seen this before.

ends?"). Polynomial-time algorithms are usually classified as computationally tractable, but even $O(n^3)$ algorithms can be impractical for some problems, given the size of the n that you often need to make money.

When your input is a number n, the best kind of result you can hope for is a log-time algorithm, because $\log n$ is roughly the number of decimal digits in n. In other words, an $O(\log n)$ algorithm solves the problem in a time proportional to the amount of time it takes to type in the input! It's also worth knowing that since (remember the change of base formula for logs?)

$$\log n = (\log 2)(\log_2 n) = (\log e)(\ln n), \tag{2.6.3}$$

any log function differs from $\log n$ only by a constant multiple, which means that $O(\log_a n)$ means the same thing no matter what the base a is.

We next come to one more fact from calculus[§] that is helpful in studying complexity, and that we again state without proof.

Theorem 2.6.5 (The Addition Principle)**.** *If $f(n) << g(n)$, then $f(n) + g(n) = O(g(n))$.* □

In other words, the slowest part of an algorithm dominates its runtime, so you can ignore the faster parts.

With the above facts in hand, we can now show that the Euclidean Algorithm runs in $O(\log n)$ time, where n is the smaller of a and b; in other words, this ancient algorithm, literally thousands of years old, is as fast as you could ask a numerical algorithm to be! We'll start with the Signed Euclidean Algorithm, where the idea is a little clearer.

Theorem 2.6.6. *Let a, b, and n be nonzero integers with $|a| \geq |b|$ and $|b| \leq n$. Using the Signed Euclidean Algorithm to compute $\gcd(a, b)$ finishes in $O(\log n)$ time, or more precisely, requires $O(\log n)$ division-with-remainder steps to finish.*

Note that you should be careful with units in complexity estimates; for example, our time estimate treats division-with-remainder as a step taking a constant amount of time, which would not be the case if we were using some kind of division algorithm that depended on the size of the integers in question.

Proof. In the notation of the Signed Euclidean Algorithm 2.4.6, we see that for $1 \leq i \leq N-1$, $|r_i| \leq \dfrac{|r_{i-1}|}{2}$. Therefore,

$$|r_{N-1}| \leq \frac{|r_{N-2}|}{2} \leq \frac{|r_{N-3}|}{2^2} \leq \cdots \leq \frac{|r_0|}{2^{N-1}} = \frac{|b|}{2^{N-1}} \tag{2.6.4}$$

Now, since r_{N-1} is a nonzero integer, $|r_{N-1}| \geq 1$. Therefore,

$$n \geq |b| \geq 2^{N-1} |r_{N-1}| \geq 2^{N-1}, \tag{2.6.5}$$

[§]Again, really analysis.

so taking $\log_2$ of both sides gives

$$N - 1 \leq \log_2 n, \tag{2.6.6}$$

or $N \leq \log_2 n + 1$. By the Addition Principle, N (the number of steps required to finish) is $O(\log n)$. □

A slightly more complicated argument (Problem 2.6.5) gives the analogous result for the standard Euclidean Algorithm.

Theorem 2.6.7. *Let a, b, and n be positive integers with $a \geq b$ and $b \leq n$. Using the Euclidean Algorithm to compute* $\gcd(a, b)$ *requires $O(\log n)$ division-with-remainder steps to finish.*

Proof. Problem 2.6.5. □

Remark 2.6.8. The fact that the Euclidean Algorithm and the faster Signed Euclidean Algorithm are both $O(\log n)$ algorithms makes an important point: Just because two algorithms have the same "big-O" complexity doesn't mean they have same speed. You can hide a lot of improvements/problems in the constant in front!

In a similar vein, the problems that go with this section primarily focus on doing similar kinds of big-O estimates for several other algoritms.

Speaking of which, it's time for some good news and some bad news. The good news is that we'll close out this chapter by talking about some very specific and practical skills you can pick up here that you can use to make money! The bad news is:

Real-life Application 2.6.9. One of the most important ways that you, as a human being (as opposed to an algorithm or an AI) and a math student, can make money, is to understand *why* things work, and explain that reasoning to others. As of this writing, understanding and explanation is not a task that (for example) AI will be able to do reliably and correctly anytime soon, so there we go – job security! However, the catch is that to be able to understand and explain for a living, you have to practice understanding and explaining, and not just calculations. Therefore, the problems in this section require to practice just that; that is, when you do big-O estimates in the problems, make sure you *explain* the processes behind your big-O estimates, instead of just writing down context-free calculations.

On a less meta note, we should also mention some ways you can use the Euclidean Algorithm to make money.

Real-life Application 2.6.10. One source of real money-making power in the Euclidean Algorithm lies in the following idea:

Roughly speaking, if you can reduce any numerical problem to the Euclidean Algorithm, then you're done.

One prominent recent example of this idea comes from the problem of factoring a number $n = pq$, where p and q are large (hundreds of digits) prime numbers. As you may know, most standard encryption depends on this problem being computationally intractable, so if you can solve it, you can pretty much undo (say) Internet security as we know it (at the time of this writing). *Shor's algorithm* [**?**], which relies on a (still hypothetical, as of this writing) quantum computer, factors $n = pq$ in two steps:

1. (The hard step) With high probability, produce a number $m < n$ such that $\gcd(n, m) > 1$.
2. (The easy step) Use the Euclidean Algorithm to compute $\gcd(m, n)$, which must be either p or q.

Obviously, the (still mostly hypothetical) quantum computer part is the hard step, and where the new ideas come in. Still, Shor's algorithm wouldn't work if it weren't for the "easy" step — solved thousands of years ago!

In fact, much of the rest of this book relies on using the Euclidean Algorithm and its variants; see, for example, Chapters 3 and 7. However, to explain how and why the Euclidean Algorithm is useful requires quite a bit more theory, so we'll hold off on that until later.

Problems

In the following problems, make sure you justify/explain each big-O estimate.

2.6.1. Suppose a and b are two n-digit numbers.

(a) Give a big-O estimate for the amount of time it takes the standard grade-school algorithm to add a and b, taking the operation of adding two single-digit numbers as your fundamental unit of time.

(b) Same, but for multiplying a and b, taking the operation of either adding or multiplying two single-digit numbers as your unit of time. To simplify your estimate, ignore carrying. (Carrying actually turns out to be one of the trickiest aspects of multiplication! See (??) for a discussion.)

2.6.2. A classic children's song starts: "99 bottles of beer on the wall, 99 bottles of beer/take one down, pass it around, 98 bottles of beer on the wall," and continues until there are no more bottles of beer. Give a big-O estimate for the amount of time to takes to finish the song, starting from n bottles of beer on the wall.

2.6.3. The traditional Christmas carol "The 12 Days of Christmas" has the following structure: On day 1, the singer gets one gift of type 1 (a partridge in a pear tree) from their true love; on day 2, the singer gets two gifts of type 2 and one gift of type 1 (two turtledoves and a partridge in a pear tree); and so on. Suppose this song can be extended to any arbitrary number of days.

(a) Give a big-O estimate of the time it takes to sing verse n, occurring on day n, as a function of n. (Assume that it takes the same length of time to sing about each gift, e.g., the eight maids a-milking takes the same length of time as the two turtledoves, which in turn takes the same amount of time as whatever is gifted on day 2739.)

(b) Give a big-O estimate of the time it takes to sign the *entire* song, starting with verse 1 on day 1, and going all the way to verse n on day n.

(c) Switching gears, give a big-O estimate of the *number* of gifts the singer receives on day n.

(d) Finally, give a big-O estimate of the *total number* of gifts the singer receives over the entire song, going from day 1 through day n.

2.6.4. This problem requires familiarity with basic matrix operations. Use arithmetic operations as your unit of time (i.e., the addition or multiplication of two numbers).

(a) Give a big-O estimate of the amount of time required to take the dot product of two vectors of length n.

(b) Give a big-O estimate of the amount of time required to take the product of an $n \times n$ matrix and an $n \times 1$ column vector.

(c) Give a big-O estimate of the amount of time required to take the product of two $n \times n$ matrices.

2.6.5. This problem shows that the standard Euclidean Algorithm is an $O(\log n)$ algorithm.

(a) In the notation of the Euclidean Algorithm 2.4.1, prove that for any $n \geq 1$, $r_n \leq \frac{r_{n-2}}{2}$. (Suggestion: Consider the cases $r_{n-1} \geq \frac{r_{n-2}}{2}$ and $r_{n-1} < \frac{r_{n-2}}{2}$.)

(b) Imitate the proof of Theorem 2.6.6 to prove that the Euclidean Algorithm requires $O(\log n)$ division-with-remainder steps to finish.

2.6.6. The goal of this problem is to estimate the complexity of multiplying two polynomials with real coefficients (i.e., polynomials in the sense of high school). We take our fundamental unit of time to be arithmetic operations in our *real coefficients*, i.e., one unit of time is one multiplication of two real numbers or one addition of two real numbers. We don't keep track of multiplications of powers of x because if you think about how a polynomial would actually be stored in memory, each power of x would be represented by a different location in an array, and so "multiplying powers of x" is really just choosing a location where a real number will be stored.

(a) To multiply two polynomials of degree 3, we do:

$$\begin{array}{rrrrrrrr}
 & & & & a_3x^3 + & a_2x^2 + & a_1x + & a_0 \\
 & & & \times & b_3x^3 + & b_2x^2 + & b_1x + & b_0 \\
\cline{5-8}
 & & & & a_3b_0x^3 + & a_2b_0x^2 + & a_1b_0x + & a_0b_0 \\
 & & & a_3b_1x^4 + & a_2b_1x^3 + & a_1b_1x^2 + & a_0b_1x & \\
 & & a_3b_2x^5 + & a_2b_2x^4 + & a_1b_2x^3 + & a_0b_2x^2 & & \\
 & a_3b_3x^6 + & a_2b_3x^5 + & a_1b_3x^4 + & a_0b_3x^3 & & & \\
\cline{2-8}
 & c_6x^6 + & c_5x^5 + & c_4x^4 + & c_3x^3 + & c_2x^2 + & c_1x + & c_0
\end{array}$$

How many (real number) multiplications does this calculation require, and how many (real number) additions?

(b) Generalize part (a) to give a big-O estimate of the total number of (real number) multiplications and additions required to multiply two polynomials of degree n.

2.6.7. For each n, consider the following procedure on an $n \times n$ chessboard: Put 1 grain of rice on the first square, 2 grains of rice on the second square, 3 grains on the third square, 4 grains on the 4th square, and so on, for each of the n^2 squares on the board.

(a) For a 3×3 chessboard, how many total grains of rice end up on the board? Express your answer as a sum or product.

(b) Given a big-O estimate of the total number of grains of rice that end up on an $n \times n$ board. Express your answer in the form $O(n^k)$ for some constant k.

In Problems 2.6.8–2.6.9, we consider the problem of sorting a list $a_1, \dots, a_n$ of n distinct numbers into ascending order, i.e., so that $a_1 < a_@ < \cdots < a_n$. Note that sorting any list of items that have a set order (e.g., words in alphabetical order) works pretty much the same way; it's just that sorting numbers is slightly easier to describe.

2.6.8. In this problem, we take the operation of `compare-and-swap` as our unit of computational time, where one compare-and-swap applied to two numbers in a list determines which number is bigger and possibly swaps their positions in the list.

Consider the following two algorithms. First, the `largest-last` algorithm takes a list $a_1, \dots, a_k$ of k numbers and moves them around until the largest number is in the last (kth) position, as follows:

1. If $a_1 < a_2$, do nothing; if $a_! > a_2$, swap a_1 and a_2 so that the (new) $a_1 < a_2$.
2. Same, but for a_2 and a_3, a_3 and a_4, all the way up to a_{k-1} and a_k.

Second, the `bubble-sort` algorithm takes a list $a_1, \dots, a_n$ of n numbers and sorts them, as follows:

1. Apply `largest-last` to the full list $a_1, \dots, a_n$, putting the largest number in the nth position.

2. Apply `largest-last` to the (new) sublist $a_1, \ldots, a_{n-1}$, putting the second largest number in the $(n-1)$st position.
3. Same, but for $a_1, \ldots, a_{n-2}$, $a_1, \ldots, a_{n-3}$, and so on, down to a_1, a_2.

You may take it as given the `largest-last` and `bubble-sort` both work as advertised; the goal of this problem is to find their time complexity.

(a) Try `bubble-sort` on the list $3, 5, 1, 4, 2$. How many `compare-and-swap` operations does that take?

(b) Given a big-O upper bound for the time, measured in `compare-and-swap` operations, it takes to run `bubble-sort` on a list of length n. Explain your estimate(s).

2.6.9. In this problem, we take the operation of `compare-and-move` as our unit of computational time, where one `compare-and-move` applied to two numbers in two lists determines which number is smaller and adds the smaller number to the end of a third list.

Consider the following two algorithms. First, the `merge-lists` algorithm takes two *sorted* lists $a_1, \ldots, a_k$ and $b_1, \ldots, b_\ell$ and creates a new third list consisting of the elements of $\{a_1, \ldots, a_k, b_1, \ldots, b_\ell\}$ in sorted order, as follows.

1. Start with an empty list (list with nothing in it) as the new list.
2. Compare a_1 and b_1, and move the smaller of a_1 and b_1 to the new list, deleting that smaller element from its original list and making that original list shorter. (This is the `compare-and-move` operation.)
3. Repeat step 2 until one of the original lists runs out of elements, at which point the other list can be appended to the new list.

Second, the `merge-sort` algorithm is a recursive algorithm that takes a list $a_1, \ldots, a_n$ of $n = 2^k$ numbers (k a nonnegative integer) and sorts them, as follows:

1. If $n = 1$ (i.e., we have a list of length 1), return the given list. Otherwise:
2. Apply `merge-sort` to the first half $a_1, \ldots, a_{n/2}$ of the list, returning a sorted list $a_1, \ldots, a_{n/2}$.
3. Apply `merge-sort` to the second half $a_{(n/2)+1}, \ldots, a_n$ of the list, returning a sorted list $a_{(n/2)+1}, \ldots, a_n$.
4. Apply `merge-lists` to $a_1, \ldots, a_{n/2}$ and $a_{(n/2)+1}, \ldots, a_n$, returning a sorted list $a_1, \ldots, a_n$.

You may take it as given that `merge-lists` and `merge-sort` both work as advertised; the goal of this problem is to find their time complexity.

(a) Try `merge-sort` on the list $8, 4, 6, 2, 7, 3, 5, 1$. How many compare-and-move operations does that take?

(b) Explain why applying `merge-lists` to two sorted lists $a_1, \ldots, a_k$ and $b_1, \ldots, b_\ell$ requires at most $k + \ell$ compare-and-move operations.

(c) Now let $T(n)$ be the maximum amount of time it takes to sort a list of length $n = 2^k$. Explain why

$$T(n) \leq 2T(n/2) + n. \tag{2.6.7}$$

(d) Use induction to prove that $T(n) \leq n \log_2(n) = k2^k$.

2.6.10. In this problem, we take the operation of dividing d into a, with remainder, as our unit of computational time.

Consider the following recursive algorithm, `prime-factors`, that takes a positive integer N as input and returns a list of its prime factors:

1. Starting with $d = 2, 3, \ldots$, divide d into N to see if d is a divisor of N.
2. The first time we find a divisor d of N, apply `prime-factors` to N/d, and append the factor d to the resulting output.
3. Otherwise, if we don't find a divisor d for $d \leq \sqrt{N}$, then N is prime, and we return the list containing exactly N.

(a) Try `prime-factors` on $N = 60$.

(b) Explain why `prime-factors` works as advertised. In particular, why is the first divisor you find necessarily prime? Why do we only need to test $d \leq \sqrt{N}$? (Suggestion: See Problems 2.2.2 and 2.2.4.)

(c) Let $T(N)$ be the maximum amount of time it takes to run `prime-factors` on a number of size at most N. Explain why

$$T(N) \leq T(N/2) + \sqrt{N}. \tag{2.6.8}$$

(d) Use (strong) induction on $N \geq 2$ to prove that $T(N) \leq (2 + \sqrt{2})\sqrt{N}$. In other words, `prime-factors` is an $O(\sqrt{N})$ algorithm.

2.6.11. Later: prove special cases of the Master Theorem: $T(n) \leq aT(n/b) + n^d$.

Chapter 3

More: The Polynomial Euclidean Algorithm

Give a small boy a hammer, and he will find that everything he encounters needs pounding.

— Abraham Kaplan, *The Conduct of Inquiry: Methodology for Behavioral Science*

If I had a hammer
I'd hammer in the morning
I'd hammer in the evening
All over this land

— Pete Seeger, *If I Had a Hammer*

As an algebraist, I have to come down on Mr. Seeger's side on the hammer question. Some of the best algebra comes from taking a tool or solution (the proverbial "hammer") and thinking, "What else can we do with this?" In this chapter, we take the Euclidean Algorithm developed in the previous chapter and generalize it to an even more useful setting, that of *polynomials with coefficients in a field.* In doing so, we illustrate yet again the key moneymaking process of applied and industrial algebra:

$$\text{Abstraction} \Rightarrow \text{Simplification} \Rightarrow \text{Generalization} \Rightarrow \text{Power}$$

(outline of chapter)

3.1 The integers mod m

In the previous chapter, we discussed what it means to work in different rings, like the integers $\mathbf{Z}$, the rationals $\mathbf{Q}$, or the real numbers $\mathbf{R}$ (Section 2.1). (Again, for now, we think of a ring as an arena in which we do battle, i.e., solve problems.) However, all of those rings are pretty much systems of numbers as you knew them in K–12, just given a fancier name.

In contrast, we now come to a system of numbers that you probably didn't see in K–12, though you may well have seen them elsewhere: The *integers (mod m)*.

So exactly how do we define a brand new system of numbers? In other words, how do we define a new ring to work in?

How to define a ring

To define a ring R:

- *Choose a set:* First, choose a set R of objects that will be the "numbers" of your ring.
- *Define addition:* Next, define how to add two elements of R. Note that when you define a ring, you're free to define $a + b$ however you please; you don't necessarily have to choose a definition of $a + b$ that resembles or is somehow related to ordinary addition of numbers (though that's what happens with most rings).
- *Define multiplication:* Finally, define how to multiply two elements of R. Again, your definition of multiplication need not be related to ordinary multiplication.

As we'll see in Chapter 4, for a given set R and a definition of addition and multiplication to form a ring, we also need to know that certain algebraic *axioms* hold. However, in this chapter, we'll only be looking at one example of a ring and we won't need the general theory, so we'll put off discussing those axioms until we actually look at rings in general.

Returning to $\mathbf{Z}/(m)$, one good place to start thinking about $\mathbf{Z}/(m)$ is with the following question.

Ask Yourself 3.1.1. Suppose we start with the ring of integers $\mathbf{Z}$ and impose the requirement that $13 = 0$. (For the moment, ignore the question of whether you can do that consistently.) If you do that, what other numbers are required to be equal to 7?

For example, if $13 = 0$, then it must be the case that $20 = 7 + 13 = 7$, and similarly, 7 must also be equal to $33, 46, 59, \ldots$. Going in the other direction, $7 = 7 - 13 = -6$, and similarly, 7 must also be equal to $-19, -32, -45, \ldots$. Putting those two chains of reasoning together, we see that if $13 = 0$, then

$$\cdots = -45 = -32 = -19 = -6 = 7 = 20 = 33 = 46 = 59 = \ldots. \tag{3.1.1}$$

Note that one thing that the numbers $20, 33, 46, 59, \ldots$ all have in common is that when you divide each of them by 13, we get a remainder of 7 (try it!), and the same fact holds true for $-19 = -32 = -45 = \ldots$, though the signs are more confusing. But there's nothing special about 7; the fact is that if we set $13 = 0$, then each integer is classified by its remainder upon division by 13, and so every integer is equal to one of the numbers $0, \dots, 12$.

That basic idea is why we turn next to the following definition.

Definition 3.1.2. Let m be a positive integer. For any integer k, k *reduced (mod m)* is the remainder you get when you divide k by m. In other words, if, for some $q, r \in \mathbf{Z}$,

$$k = qm + r \qquad \text{with } 0 \leq r < m, \tag{3.1.2}$$

then k reduced (mod m) is equal to r.

Note that the uniqueness part of the Division Theorem 2.3.1 is what makes Definition 3.1.2 unambigous. In any case, if we want to declare that $m = 0$, then (3.1.2) means that every number k is equal to its remainder when you divide by m, which is why in the following defintion, $\mathbf{Z}/(m)$ only contains the numbers $0, \dots, m-1$.

Definition 3.1.3. Let m be a positive integer. We define the ring $\mathbf{Z}/(m)$, or the *integers (mod m)*, as follows.

- The underlying set of $\mathbf{Z}/(m)$ is $\{0, \dots, m-1\}$.
- For $a, b \in \mathbf{Z}/(m)$, we define $a + b$ to be the ordinary integer sum of a and b, reduced mod m.
- Similarly, for $a, b \in \mathbf{Z}/(m)$, we define the product ab to be the ordinary integer product of a and b, reduced mod m.

When we work in $\mathbf{Z}/(m)$, we refer to m as the *modulus* of our ring.

Definition 3.1.3 gives an unambigous definition of the algebraic object we need. However, it will be very helpful, and even conceptually useful, to have more flexibility in how we can write down the elements of $\mathbf{Z}/(m)$, so we introduce the following idea.

Definition 3.1.4. Let m be a positive integer. To say that two integers a and b are *congruent (mod m)*, or equivalently, that a is equal to b (mod m), means that $b - a$ is divisible by m. Put another way, a is equal to b (mod m) exactly when

$$a = qm + b \tag{3.1.3}$$

for some integer q.

So, for example, the reduction of k (mod m) (Definition 3.1.2) is congruent to k (mod m). More generally, we have the following *Congruent Substitution Principle.*

Congruent Substitution Principle: If we are working in the ring $\mathbf{Z}/(m)$, we can always replace any integer a with any integer b congruent to a (mod m). In fact, we think of a and b as two different names for the *same element* of $\mathbf{Z}/(m)$.

Even more generally, to expand on our original goal, we have that:

The $m = 0$ Principle: Arithmetic in $\mathbf{Z}/(m)$ is like regular arithmetic, except that we declare that $m = 0$, and accept all of the relations that follow as a consequence (such as the Congruent Substitution Principle).

Here are some examples of consequences of the Congruent Substitution Principle (or the $m = 0$ Principle).

Example 3.1.5. In $\mathbf{Z}/(11)$, $3+8=0$, so we can think of 8 as being equal to -3 in $\mathbf{Z}/(11)$. Note that $(-3)^2 = 9$, which is consistent with $8^2 = 64$ reducing to 9 (mod 11), since $64 = 5(11)+9$. More generally, if m is odd, and we're working on something multiplicative, it can be helpful to list the elemnts of $\mathbf{Z}/(m)$ using negatives. For example, the elements of $\mathbf{Z}/(11)$ can be written as:

$$\begin{aligned}\{0,1,2,3,4,5,6,7,8,9,10\} &= \{0,1,2,3,4,5,-5,-4,-3,-2,-1\}\\ &= \{-5,-4,-3,-2,-1,0,1,2,3,4,5\}.\end{aligned} \tag{3.1.4}$$

Example 3.1.6. Confession: While it's useful and important for you to understand $\mathbf{Z}/(m)$ for any m, in this book, the examples of $\mathbf{Z}/(m)$ we will use most often are $\mathbf{Z}/(2)$ and $\mathbf{Z}/(3)$ (and really, $\mathbf{Z}/(2)$ far more than any other $\mathbf{Z}/(m)$). For $\mathbf{Z}/(2)$, the elements are $\{0,1\}$, and addition and multiplication are:

$+$	0	1
0	0	1
1	1	0

$*$	0	1
0	0	0
1	0	1

(3.1.5)

Note also that $+1 = -1$ in $\mathbf{Z}/(2)$, or in other words, it's impossible to make sign errors mod 2.

For $\mathbf{Z}/(3)$, instead of the more commonly used $\{0,1,2\}$, I often personally prefer to use the names $\{0,1,-1\}$. In those terms, addition and multiplication become:

$+$	0	1	-1
0	0	1	-1
1	1	-1	0
-1	-1	0	1

$*$	0	1	-1
0	0	0	0
1	0	1	-1
-1	0	-1	1

(3.1.6)

Example 3.1.7. In $\mathbf{Z}/(13)$, $2(7) = 1$, so we can think of 7 as being equal to $\frac{1}{2}$ in $\mathbf{Z}/(13)$. Note that this substitution is consistent with the arithmetic of fractions. For example, still working in $\mathbf{Z}/(13)$, we have:

$$\frac{1}{8} = \left(\frac{1}{2}\right)^3 = 7^3 = 343 = 5, \tag{3.1.7}$$

which is indeed consistent with $8(5) = 40 = 1$.

I also confess that one reason for the above manipulations is that I want you to wonder:

Ask Yourself 3.1.8. On the one hand, will the Congruence Substitution Principle, or more generally, the $m = 0$ Principle, always work consistently in $\mathbf{Z}/(m)$, instead of resulting in some kind of contradiction like $0 = 1$? On the other hand, if we just stick with the more clearly consistent operations of Definition 3.1.3, can we still rely on ordinary properties of arithmetic like the associativity of multiplication, i.e., $(ab)c = a(bc)$?

Fear not, the answer to both questions in Ask Yourself 3.1.8 is yes. However, we'll need several more layers of abstraction to explain that fact efficiently, so we'll wait to do that until Chapter 7. Until then, several problems below give you a chance to play around in the ring $\mathbf{Z}/(p)$ (p a prime) with two interesting ideas. First, we have the idea of a *primitive element.*

Definition 3.1.9. Let p be an odd prime. To say that $a \in \mathbf{Z}/(p)$, $a \neq 0$, is a *primitive element of* $\mathbf{Z}/(p)$, or a *primitive element (mod p)*, means that if we compute the powers a, a^2, a^3, etc., we eventually encounter every nonzero element $1, \dots, p-1$ of $\mathbf{Z}/(p)$.

Example 3.1.10. Consider the ring $\mathbf{Z}/(7)$ (i.e., take $p = 7$). On the one hand, if we compute powers of 3 in $\mathbf{Z}/(7)$, we get:

$$\begin{array}{lll} 3^1 = 3 & 3^2 = 9 = 2 & 3^3 = 3(2) = 6 \\ 3^4 = 3(6) = 18 = 4 & 3^5 = 3(4) = 12 = 5 & 3^6 = 3(5) = 15 = 1. \end{array} \tag{3.1.8}$$

We see that each of the numbers $1, \dots, 6$ is a power of 3, so 3 is a primitive element of $\mathbf{Z}/(7)$. *Important computational note:* As we compute the powers of 3 (mod 7), we reduce the final answer mod 7 each time, and then multiply that reduced answer by 3 to get the next power of 3 (mod 7). That practice produces a mild savings of effort in this case, but is critical when, say, we compute the powers of 7919 mod 8675309 or the powers of a 150-digit prime number mod a 300-digit prime number.

Getting back to primitivity, in contrast, if we compute powers of 2 in $\mathbf{Z}/(7)$, we get:

$$\begin{array}{lll} 2^1 = 2 & 2^2 = 4 & 2^3 = 8 = 1 \\ 2^4 = 2(1) = 2 & 2^5 = 2(2) = 4 & 2^6 = 2(4) = 8 = 1, \end{array} \tag{3.1.9}$$

and so on. We see that if we keep cycling through all powers of 2, we'll only ever get 2, 4, and 1 as answers, so 2 is not primitive (mod 7).

Believe it or not, using only the terms we've defined in this section, we can now state a question to which, as of this writing in 2020, no one on earth knows the answer.

Unsolved Problem 3.1.11. Is 2 primitive mod p for infinitely many primes p?

Incredibly, there is actually a concrete conjectured answer for how often 2 is primitive mod p: *Artin's conjecture* posits, for example, that 2 is primitive mod p for roughly 37.4% of all primes p, or more generally, that the same holds replacing 2 with any other prime not equal to 1 mod 4. For more on Artin's conjecture, see (??); see Problems 3.1.3 and 3.1.3 for some experiments along these lines.

Another interesting idea you're now ready to experiment with is the idea of a *quadratic residue.*

Definition 3.1.12. Let p be an odd prime. To say that $a \in \mathbf{Z}/(p)$, $a \neq 0$, is a *quadratic residue (mod p)* means that a is a square (mod p), or in other words, the equation $x^2 = a$ has a solution $x \in \mathbf{Z}/(p)$. Nonzero elements of $\mathbf{Z}/(p)$ that are not quadratic residues are called *(quadratic) nonresidues (mod p).* (Note that by definition, 0 is neither a quadratic residue nor a nonresidue.)

It turns out that the percentage of numbers mod p that are quadratic residues is quite predictable (see Problem 3.1.6), but the order in which residues and nonresidues occur as we go from 1 to $p-1$ seems to behave quite randomly. As a result, quadratic residues have a surprising application, in that they can be used to create reproducible psuedorandom number generators; see (??).

Problems

3.1.1. Let m be a positive integer, and let k be an arbitrary integer. Prove that k is congruent (mod m) to exactly one integer r between 0 and $n-1$. (Suggestion: Division Theorem.)

3.1.2. You may remember from high school algebra that a quadratic equation has at most two solutions. This problem explores the fact that this is not generally the case in $\mathbf{Z}/(m)$. (Note that solutions that are the same (mod m) are considered to be the same solution; for example, in $\mathbf{Z}/(2)$, $x^2 = 1$ has exactly one solution, namely, $x = 1 = 3 = 5 = 7 = \dots$.)

(a) By trial and error, find a positive integer m such that the equation $x^2 = 1$ has more than two solutions in $\mathbf{Z}/(m)$.

(b) Can you find a $\mathbf{Z}/(m)$ where $x^2 = 1$ has 8 solutions? 16?

3.1.3. Is 2 primitive mod 11? Mod 13, 17, 19, 23, 29? (Remember to reduce mod p each time you compute a new power of 2; see Example 3.1.10.)

3.1.4. Is 3 primitive mod 11? Mod 13, 17, 19, 23, 29? (Remember to reduce mod p each time you compute a new power of 3; see Example 3.1.10.)

3.1.5. For $p = 7, 11, 13, 17, 19, 23$, list all quadratic residues mod p (Definition 3.1.12). (Suggestion: Square everything mod p.)

3.1.6. This problem relies on Problem 3.1.5.

(a) How many residues are there mod 7? Mod 11? Mod 13, 17, 19, 23? Look for a pattern in the data from Problem 3.1.5.

(b) Proving the pattern you discovered always holds is not easy, but partial progress is a more tractable problem: For p an odd prime, prove that no more than half of the nonzero elements of $\mathbf{Z}/(p)$ are residues. (Suggestion: Look for patterns in your work from Problem 3.1.5; see also Example 3.1.5.)

3.1.7. Use your data from Problem 3.1.5 to separate the odd primes p (including 3 and 5) into two catgories: Yes (-1 is a QR mod p) and No (-1 isn't a QR mod p). Do you see any patterns in the values of Yes/No primes mod 4? Mod 6? Mod 8? Mod 12?

3.1.8. Use your data from Problem 3.1.5 to separate the odd primes p (including 3 and 5) into two catgories: Yes (2 is a QR mod p) and No (2 isn't a QR mod p). Do you see any patterns in the values of Yes/No primes mod 4? Mod 6? Mod 8? Mod 12?

3.1.9. For each a such that $1 \leq a \leq 6$, find the smallest integer $n > 0$ such that $a^n = 1$ in $\mathbf{Z}/(7)$. Do you see any patterns relating to the number 7?

3.1.10. For each a such that $1 \leq a \leq 10$, find the smallest integer $n > 0$ such that $a^n = 1$ in $\mathbf{Z}/(11)$. Do you see any patterns relating to the number 11?

3.1.11. For each a such that $1 \leq a \leq 12$, find the smallest integer $n > 0$ such that $a^n = 1$ in $\mathbf{Z}/(13)$. Do you see any patterns relating to the number 13?

3.1.12. For this problem, you will need a list of all prime numbers up to 200, which shouldn't be hard to find online or in some other reference.

(a) Reduce each of those prime numbers (mod 6). What patterns do you see?

(b) Reduce each of those prime numbers (mod 9). What patterns do you see?

(c) Reduce each of those prime numbers (mod 10). What patterns do you see?

(d) What do you think will happen in general, if you look at the distribution of prime numbers (mod m)? Make a conjecture.

3.2 Modular linear equations and fields

The following question is interesting for both theoretical and practical (money-making) reasons.

Question 3.2.1. For which $a, b \in \mathbf{Z}/(m)$ can we solve the equation $ax = b$ in $\mathbf{Z}/(m)$ (i.e., for some $x \in \mathbf{Z}/(m)$)?

Fortunately, the Congruent Substitution Principle allows us to reduce Question 3.2.1 to the material in Section 2.5, since:

$$\begin{aligned} & ax = b \text{ in } \mathbf{Z}/(m) \\ \Leftrightarrow\ & ax = qm + b \text{ in } \mathbf{Z}, \text{ for some } q \in \mathbf{Z} \\ \Leftrightarrow\ & ax + my = b \text{ in } \mathbf{Z}, \text{ for some } y \in \mathbf{Z}, \end{aligned} \tag{3.2.1}$$

where the second $\Leftrightarrow$ comes from taking $y = -q$.

Therefore, in $\mathbf{Z}/(m)$, Bezout's Identity 2.5.1 and Corollary 2.5.4 become:

Corollary 3.2.2. *For $a, b \in \mathbf{Z}/(m)$, $ax = b$ has a solution $x \in \mathbf{Z}/(m)$ exactly when $\gcd(a, m)$ divides b (in $\mathbf{Z}$). Furthermore, Euclidean Rewriting gives an explicit algorithm for solving $ax = b$.*

Example 3.2.3. To give a concrete example of what Corollary 3.2.2 implies, we find one solution to the equation $50x = 2$ in $\mathbf{Z}/(68)$. The point of (3.2.1) is that solving $50x = 2$ in $\mathbf{Z}/(68)$ is equivalent to solving

$$50x + 68y = 2 \tag{3.2.2}$$

in $\mathbf{Z}$, and then ignoring y.

To solve (3.2.2), we first apply the Euclidean Algorithm to find $\gcd(m, a)$, where $m = 68$ (the modulus) and $a = 50$:

$$\begin{aligned} 68 &= 1(50) + 18 \\ 50 &= 2(18) + 14 \\ 18 &= 1(14) + 4 \\ 14 &= 3(4) + 2 \\ 4 &= 2(2). \end{aligned} \tag{3.2.3}$$

Applying Euclidean Reduction (Algorithm 2.5.2) with $68 = m$, $50 = a$, we get

$$\begin{aligned} 18 &= 68 - 1(50) = m - a \\ 14 &= 50 - 2(18) = a - 2(m - a) = 3a - 2m \\ 4 &= 18 - 1(14) = (m - a) - (3a - 2m) = 3m - 4a \\ 2 &= 14 - 3(4) = (3a - 2m) - 3(3m - 4a) = 15a - 11m. \end{aligned} \tag{3.2.4}$$

Therefore, $(-11)(68) + 15(50) = 2$, which mean that $15(50) = 2$ in $\mathbf{Z}/(68)$ — remember, in $\mathbf{Z}/(68)$, $68 = 0$. In fact, we can cut down on the bookkeeping involved here by using the fact that $m = 0$ at each step along the way to throw out all of the occurences of m:

$$\begin{aligned} 18 &= 68 - 1(50) = -a \pmod{68} \\ 14 &= 50 - 2(18) = a - 2(-a) = 3a \pmod{68} \\ 4 &= 18 - 1(14) = (-a) - (3a) = -4a \pmod{68} \\ 2 &= 14 - 3(4) = (3a) - 3(-4a) = 15a \pmod{68}. \end{aligned} \tag{3.2.5}$$

Less writing, and if you practice, more reliable.

(By the way, if you're curious, the reason we keep saying "one solution" is that there's another solution, namely, $x = 49 = -19$, as you can check. See Problem 3.2.2 for more about multiple solutions.)

As a special case of Corollary 3.2.2, we have:

Corollary 3.2.4. *If p is prime, and $a \neq 0$ in $\mathbf{Z}/(p)$ (i.e., a is not congruent to $0 \pmod{p}$), then $ax = 1$ for some $x \in \mathbf{Z}/(p)$.*

More generally, we have:

Corollary 3.2.5. *Let m be a positive integer, and let $a \in \mathbf{Z}/(m)$. We have that $ax = 1$ for some $x \in \mathbf{Z}/(m)$ if and only if* $\gcd(a, m) = 1$.

Corallary 3.2.4 is more important than it might appear to be at the moment, because that result means that $\mathbf{Z}/(p)$ with p prime is a kind of ring that will be more useful to us than $\mathbf{Z}/(m)$ is for m not prime. This calls, of course, for some definitions.

Definition 3.2.6. Let R be a ring. (Again, we haven't really defined ring yet, but think $R = \mathbf{Z}$, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{C}$, or $\mathbf{Z}/(m)$.) For $a \in R$, a *multiplicative inverse of* a, or if the context is clear, simply an *inverse of* a, is some $b \in R$ such that $ab = 1$. Since an element can have only one inverse (Problem 3.2.3), we use a^{-1} to denote *the* inverse of a.

To say that a is a *unit* in R means that a has a multiplicative inverse in R.

Note that the definitions of inverse and unit in a ring R depend highly on the requirement that inverses also be contained in R. For example, 2 is *not* a unit of the ring of integers $\mathbf{Z}$ because $\frac{1}{2}$ is not an integer; but 2 *is* a unit of the ring of rational numbers.

Definition 3.2.7. A *field* is a ring R in which every nonzero element is a unit (and $1 \neq 0$). In other words, to say that a nonzero ring R is a field means that for every $a \neq 0$ in R, there exists some $b \in R$ such that $ab = 1$.

Familiar rings that also happen to be fields include the rationals $\mathbf{Q}$, the reals $\mathbf{R}$, and the complex numbers $\mathbf{C}$. Crucially, we also have the following interpretation of Corollary 3.2.4 when m is prime.

Corollary 3.2.8. *The ring $\mathbf{Z}/(p)$ is a field.* $\square$

To spell it out a bit more, Corollary 3.2.4 says that when p is prime and $a \neq 0$ in $\mathbf{Z}/(p)$, we can always use Euclidean reduction to solve the equation $ax = 1$ in $\mathbf{Z}/(p)$. We indicate the importance of the fact that $\mathbf{Z}/(p)$ is a field by calling it by the following alternate names.

Definition 3.2.9. We use $\mathbf{F}_p$ to refer to $\mathbf{Z}/(p)$, or the *field of order p*. (The uniqueness implied by the "the" here will be justified later.) This field is also sometimes known as the *Galois field of order p*, or $GF(p)$ for short. Alas, as we'll see, the term *order* is overused in algebra, but here order refers to the fact that there are p elements in $\mathbf{Z}/(p)$.

We close this section by introducing some notation for inverses that will be useful when we work with fields.

Definition 3.2.10. Let R be a ring, let b be a unit of R, and let a be an element of R. We define the *fraction* $\dfrac{a}{b}$ to be ab^{-1}.

For now, we can think of fractions as convenient abbreviations, giving a partial retroactive justification for the fractions appearing in Example 3.1.7. We will later show that, as defined, fractions have the same basic properties in general as fractions in the real numbers have; see Section 4.2.

Problems

3.2.1. Use the Euclidean Algorithm either to find one solution for the following linear equations in $\mathbf{Z}/(m)$, for the indicated m, or to show that no solution exists.

(a) Solve $12x = 7$ in $\mathbf{Z}/(35)$.

(b) Solve $24x = 7$ in $\mathbf{Z}/(81)$.

(c) Solve $77x = 5$ in $\mathbf{Z}/(85)$.

(d) Solve $77x = 5$ in $\mathbf{Z}/(110)$.

(e) Solve $314x = 5$ in $\mathbf{Z}/(451)$.

(f) Solve $226x = 12$ in $\mathbf{Z}/(538)$.

3.2.2. Suppose m is a fixed positive integer, a and b are nonzero elements of $\mathbf{Z}/(m)$, and x is a solution to $ax = b$ in $\mathbf{Z}/(m)$. Let $d = \dfrac{n}{\gcd(a,n)}$. Prove that for any integer k, $x + kd$ is also a solution to $ax = b$ in $\mathbf{Z}/(m)$. (Conversely, one can prove that this is the complete list of solutions to $ax = b$ in $\mathbf{Z}/(m)$.)

3.2.3. Let R be a ring, let a be a nonzero element of R, and suppose that b and c are each inverses of a. Prove that $b = c$. You may assume that multiplication is commutative in R.

3.2.4. Use the Euclidean Algorithm to find the inverse of each of the following elements of $\mathbf{F}_p$ for the indicated primes p.

(a) Find the inverse of 9 in $\mathbf{F}_{19}$.

(b) Find the inverse of 27 in $\mathbf{F}_{31}$.

(c) Find the inverse of 34 in $\mathbf{F}_{71}$.

(d) Find the inverse of 17 in $\mathbf{F}_{101}$.

(e) Find the inverse of 118 in $\mathbf{F}_{257}$.

3.2.5. For each of the following rings $\mathbf{Z}/(m)$:

- Make a list of the set $U(m)$ of all units in $\mathbf{Z}/(m)$. (See Corollary 3.2.5.)
- Make a "multiplication table" for $U(m)$. That is, make a table whose rows and columns are labelled with the elements of $U(m)$, and in the entry corresponding to the row labelled a and the column labelled b, write in the element ab (reduced mod m).

(a) $\mathbf{Z}/(20)$.

(b) $\mathbf{Z}/(24)$.

(c) $\mathbf{Z}/(7)$.

(d) $\mathbf{Z}/(11)$.

(e) $\mathbf{Z}/(15)$.

(f) $\mathbf{Z}/(18)$.

3.2.6. Let R be a ring, and let a and b be units in R. Prove that ab is also a unit. (Suggestion: Think $\dfrac{1}{ab}$.)

3.3 Polynomials with coefficients in a ring

Now, we pull a move that is entirely characteristic of how algebra (and really, theoretical math in general) works: We describe how to take one piece of abstract nonsense and build a new piece of abstract nonsense. To be specific, we now describe how to take an arbitrary ring R (think $R = \mathbf{Z}$, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{C}$, $\mathbf{Z}/(m)$) and create a new ring $R[x]$. Most of the time we are interested in the case where R is a field ($R = \mathbf{Q}$, $\mathbf{R}$, $\mathbf{C}$, $\mathbf{Z}/(p) = \mathbf{F}_p$), but it's mildly useful to describe the general case, and takes no extra effort.

Before we get to the formal definition of polynomials, however, we need to get one thing straight.

Polynomials are not (just) functions — they are abstract objects that are elements of a ring. In fact, we will most often use polynomials as if they were numbers in some very strange system of numbers.

So with that out of the way:

Definition 3.3.1. Let R be a ring. (Again, we haven't defined what a ring is yet, but it's enough to think of R as being one of the examples $\mathbf{Z}$, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{C}$, $\mathbf{Z}/(m)$ that we have discussed so far.) We define the ring $R[x]$, the *ring of polynomials with coefficients in R*, as follows.

- The underlying set for $R[x]$ is the set of all expressions of the form

$$\sum_{i=1}^{n} a_i x^i = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_2x^2 + a_1x + a_0, \tag{3.3.1}$$

  where each a_i is an element of the ring R. An expression of the form (3.3.1) is called a *polynomial with coefficients in R*. Note that we can "pad out" a polynomial like 3.3.1 by adding more terms of the form $0x^k$ for $k > n$, and we declare that this does not change the value of the polynomial. More generally, we declare that adding or removing any finite number of such zero terms does not change the value of a polynomial, but otherwise, two polynomials are equal exactly when their corresponding coefficients in each degree are equal. Similarly, we define the *zero polynomial* to be the polynomial whose coefficients (written out explicitly or not) are all zero.

- Briefly, addition and multiplication of polynomials with coefficients in R are each defined to work like addition and multiplication of polynomials with real coefficients, except that all coefficient arithmetic is performed in the ring R. That is, we define the sum of two polynomials by:

$$\begin{array}{rrrr} & a_nx^n+\cdots+ & a_1x+ & a_0 \\ + & b_nx^n+\cdots+ & b_1x+ & b_0 \\ \hline & (a_n+b_n)x^n+\cdots+ & (a_1+b_1)x+ & (a_0+b_0) \end{array} \tag{3.3.2}$$

  where all additions use the addition operation from R.

  Similarly, we define the product of two polynomials by:

$$\begin{array}{rrrrrrrrr} & & & & a_nx^n & + \quad \cdots & + \; a_2x^2+ & a_1x+ & a_0 \\ & & & & b_kx^k & + \quad \cdots & + \; b_2x^2+ & b_1x+ & b_0 \\ \cline{5-9} & & & & a_nb_0x^n & + \quad \cdots & +a_2b_0x^2 & +a_1b_0x & +a_0b_0 \\ & & & a_nb_1x^{n+1} \; + & \cdots & +a_2b_1x^3 & +a_1b_1x^2 & +a_0b_1x & \\ & a_nb_2x^{n+2} \; + & \cdots & +a_2b_2x^4 & +a_1b_2x^3 & +a_0b_2x^2 & & & \\ \iddots & \iddots & \iddots & \iddots & \iddots & & & & \\ \hline a_nb_kx^{n+k}+ & & \cdots & & & & + \; c_2x^2+ & c_1x+ & c_0 \end{array}$$

  where the coefficients $c_0, c_1, c_2, \ldots$ of the product are defined by

$$\begin{aligned} c_0 &= a_0b_0 \\ c_1 &= a_1b_0 + a_0b_1 \\ c_2 &= a_2b_0 + a_1b_1 + a_2b_0 \\ &\vdots \\ c_m &= \sum_{i+j=m} a_ib_j = a_mb_0 + \cdots + a_0b_m \\ &\vdots \\ c_{n+k} &= a_nb_k \end{aligned} \tag{3.3.3}$$

  and again, all coefficient operations are done in the ring R.

When we work in the ring $R[x]$, we call R the *coefficient ring* of $R[x]$. In those terms, the above just says that polynomials in $R[x]$ work exactly the same as the polynomials you've seen since high school, except that all of the coefficient arithmetic is done in the coefficient ring R instead of in the real numbers.

Convention 3.3.2. When working with polynomials with coefficients in $\mathbf{Z}/(m)$, we write each coefficient as one of $0, \ldots, m-1$. For example, when $m = 2$, we only use the coefficients 0 and 1. Exception: For coefficients in $\mathbf{Z}/(3)$, we use the coefficients $-1, 0, 1$; see Example 3.1.6.

Remark 3.3.3. As we saw in several places in Section 3.1, the names $0, \dots, m-1$ are not always the best ones to use for elements of $\mathbf{Z}/(m)$. However, we'll stick with Convention 3.3.2 for two reasons:

1. If m is very large, as sometimes happens in practice, you would want to declare a standard fixed set of names when programming arithmetic in $\mathbf{Z}/(m)$.

2. More immediately, using Convention 3.3.2 will force you to reduce mod m constantly, which will help you get a feel for the idea.

Example 3.3.4. To help you start to learn to multiply polynomials with coefficients in a ring other than ordinary real numbers or complex numbers, consider $p(x) = 5x^3 + 10x^2 + 2x + 7$ and $q(x) = 4x^2 + 3x + 11$ in the ring $\mathbf{F}_{13}[x]$ of polynmials with coefficients in $\mathbf{F}_{13} = \mathbf{Z}/(13)$. We calculate the product $p(x)q(x)$ as follows.

$$\begin{array}{rrrrrr}
 & 5x^3 + & 10x^2 + & 2x + & 7 & \\
 & & 4x^2 + & 3x + & 11 & \\
\hline
 & 3x^3 + & 6x^2 + & 9x + & 12 & \\
2x^4 + & 4x^3 + & 6x^2 + & 8x & & \\
7x^5 + \quad x^4 + & 8x^3 + & 2x^2 & & & \\
\hline
7x^5 + 3x^4 + & 2x^3 + & x^2 + & 4x + & 12 &
\end{array} \tag{3.3.4}$$

Note that we reduce each coefficient (mod 13) at each stage. Mini-exercise: Check for yourself that this calculation is done correctly.

In the rest of this section, we consider some concepts related to the *degree* of a polynomial. We begin, as usual, with some definitions.

Definition 3.3.5. Let $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$, and assume that $f(x)$ is not 0 (i.e., not every coefficient of $f(x)$ is equal to 0). The *degree* of $f(x)$, or $\deg f(x)$, is defined to be the largest k such that $a_k \neq 0$. (Note that if $a_n = 0$, then $\deg f(x)$ will be strictly less than n; see the discussion about "padding out" a polynomial in Definition 3.3.1.) If $\deg f(x) = k$, then a_k is called the *leading coefficient* of $f(x)$, and $a_k x^k$ is called the *leading term* of $f(x)$. To say that a polynomial $f(x)$ is *monic* means that the leading coefficient of $f(x)$ is 1.

We also define the degree of the zero polynomial to be $\deg 0 = -\infty$, a choice that will make more sense momentarily.

Our next goal is to prove that the degree of the product $p(x)q(x)$ is the sum of the degrees of $p(x)$ and $q(x)$, as you may remember from high school algebra (Theorem 3.3.8). This statement is not hard to prove, except for the slight hitch that it isn't true, as the following example shows.

Example 3.3.6. In the ring $(\mathbf{Z}/(6))[x]$, we have

$$(2x^5)(3x^7) = 6x^{12} = 0x^{12} = 0, \tag{3.3.5}$$

so we have polynomials of degree 5 and 7 whose product has degree $-\infty$. Note that $6x^{12} = 0x^{12}$ because $6 = 0$ in the coefficient ring $\mathbf{Z}/(6)$.

Therefore, to make Theorem 3.3.8 actually work, we need to make an additional assumption about the coefficient ring R.

Definition 3.3.7. To say that a ring R has the *zero factor property* means that if $a, b \in R$ and $ab = 0$, then either $a = 0$ or $b = 0$. Equivalently, having the zero factor property means that the product of two nonzero elements of R is still nonzero.

For example, all of the familiar rings $\mathbf{Z}$, $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{C}$ have the zero factor property; in fact, we will show later that every field has the zero factor property (Theorem 4.2.12). As for our other favorite example $\mathbf{Z}/(m)$, if m is not prime (i.e., $m = ab$ for integers $a, b > 1$), then $\mathbf{Z}/(m)$ does not have the zero property (Problem 3.3.2); and if p is prime, then $\mathbf{F}_p = \mathbf{Z}/(p)$ is a field (Corollary 3.2.4) and therefore has the zero factor property.

Theorem 3.3.8. *Suppose R is a ring with the Zero Factor Property and $f(x), g(x) \in R[x]$. Then*

$$\deg(f(x)g(x)) = \deg(f(x)) + \deg(g(x)), \tag{3.3.6}$$

where the sum in (3.3.6) is defined for $-\infty$ by declaring that $-\infty$ plus anything is $-\infty$.

Proof. If either $f(x) = 0$ or $g(x) = 0$, then under the above interpretation, (3.3.6) becomes $-\infty = -\infty$, so we may assume that $f(x)$ and $g(x)$ are both nonzero. In that case, let $a_n x^n$ and $b_k x^k$ be the leading terms of $f(x)$ and $g(x)$, respectively. Then by the definition of polynomial multiplication (Definition 3.3.1), the leading term of $f(x)g(x)$ will be $a_n b_k x^{n+k}$, since $a_n b_k \neq 0$ (by the zero factor property in the coefficient ring R). Therefore, $\deg(f(x)g(x)) = n + k = \deg(f(x)) + \deg(g(x))$, and the theorem follows. □

Theorem 3.3.8 has a number of useful consequences, such as the following corollaries, whose proofs are left to you.

Corollary 3.3.9. *Let R be a ring with the zero factor property, and suppose $f(x), g(x), h(x)$ are polynomials in $R[x]$ such that $f(x) = g(x)h(x)$. Then one of $g(x)$ and $h(x)$ must have degree at most $\dfrac{\deg(f(x))}{2}$.*

Proof. Problem 3.3.3. □

Corollary 3.3.10. *Let R be a ring with the zero factor property, and suppose $u(x)$ is a unit (Definition 3.2.6) in $R[x]$. Then $u(x)$ must be a nonzero constant polynomial $u = u(x)$, and in fact, u is actually a unit in R.*

Proof. Problem 3.3.4. □

Problems

3.3.1. Calculate the following polynomial products in the indicated polynomial rings. (Remember, to say that a calculation takes place in the ring $R[x]$ means that the coefficient calculations take place in the ring R.) Put your final answer in a form where all coefficients are as small as possible (either smallest positive or smallest absolute value, up to you), and also make sure that each of the coefficients that appears in the intermediate steps of your work is similarly reduced. (I.e., reduce coefficients mod p as you go along, instead of only reducing at the end.)

(a) $(9x^3 + 10x^2 + 3x + 12)(2x^2 + 14x + 3)$ in $\mathbf{F}_{19}[x]$.

(b) $(9x^3 + 10x^2 + 3x + 10)(2x^2 + 12x + 3)$ in $\mathbf{F}_{17}[x]$.

(c) $(5x^3 + 2x^2 + x + 1)(3x^3 + 2x + 4)$ in $\mathbf{F}_7[x]$.

(d) $(4x^3 + 2x^2 + x + 1)(3x^3 + 2x + 4)$ in $\mathbf{F}_5[x]$.

(e) $(2x^4 + 3x^3 + x^2 + x + 1)(x^2 + 2x + 1)$ in $\mathbf{F}_3[x]$.

(f) $(x^7 + x^5 + x^4 + x^3 + x + 1)(x^4 + x^3 + x^2 + 1)$ in $\mathbf{F}_2[x]$.

(g) $(x^9 + x^6 + x^5 + x^3 + x^2 + x + 1)(x^5 + x^4 + 1)$ in $\mathbf{F}_2[x]$.

3.3.2. This problem considers the zero factor property (Definition 3.3.7).

(a) Prove (by giving an example) that $\mathbf{Z}/(6)$ does not have the zero factor property.

(b) Prove that if m is *composite* (i.e., $m = ab$ for some integers $a, b > 1$), then $\mathbf{Z}/(m)$ does not have the zero factor property.

3.3.3. (*Proves Corollary 3.3.9*) Let R be a ring with the zero factor property, and suppose $f(x), g(x), h(x)$ are polynomials in $R[x]$ such that $f(x) = g(x)h(x)$. By symmetry, we may assume $\deg(g(x)) \leq \deg(h(x))$ (else swap $g(x)$ and $h(x)$). Prove that $\deg(g(x)) \leq \frac{1}{2}\deg(f(x))$.

3.3.4. (*Proves Corollary 3.3.10*) Let R be a ring with the zero factor property, and suppose $u(x)v(x) = 1$ in $R[x]$ (i.e., $u(x)$ and $v(x)$ are units in $R[x]$).

(a) Prove that $\deg u(x) = \deg v(x) = 0$. Why does that mean that $u(x)$ and $v(x)$ are actually constant polynomials, and therefore, elements of R?

(b) Prove that $u(x)$ and $v(x)$ must actually be units in R.

3.3.5. The goal of this problem is to count polynomials of given types in $\mathbf{F}_p[x]$.

(a) How many polynomials of degree ≤ 4 are there in $\mathbf{F}_p[x]$? Describe how you would list all of them.

(b) How many *monic* polynomials of degree 5 (exactly) are there in $\mathbf{F}_p[x]$? Describe how you would list all of them.

(c) How many monic polynomials of degree 6 (exactly), *with nonzero constant terms*, are there in $\mathbf{F}_p[x]$? Describe how you would list all of them.

3.4 Polynomial division with remainder

The first building block of the Euclidean Algorithm for computing $\gcd(a, b)$ is a careful consideration of something you've known since grade school, namely, division with remainder. So let's start by reviewing division with remainder; specifically, let's see what happens when we divide 68931 (the *dividend*) by 212 (the *divisor*), with remainder.

$$
\begin{array}{r|r}
 & 3\,2\,5 \\ \cline{2-2}
212 & 6\,8\,9\,3\,1 \\
 & 6\,3\,6\,0\,0 \\ \cline{2-2}
 & 5\,3\,3\,1 \\
 & 4\,2\,4\,0 \\ \cline{2-2}
 & 1\,0\,9\,1 \\
 & 1\,0\,6\,0 \\ \cline{2-2}
 & 3\,1
\end{array}
\tag{3.4.1}
$$

Well OK, you knew how to do that already, but the point is, I want you think about *why* you take the steps you do you in (3.4.1).

Ask Yourself 3.4.1. In (3.4.1), why do you pick 3, 2, and 5 as the appropriate multiples of 212? That is, what general rule tells you how to pick the digits in the quotient? Similarly, what general rule tells us that we stop when we get 31?

After some thought, you might remember that the digits of the quotient are chosen to wipe out as much of the biggest digit of what's left of the dividend ("212 goes into 689 3 times," etc.), and we stop because the remainder 31 is strictly less than the divisor 212.

Moving forward in your mathematical history, you may recall that an analogous process works for polynomials with real coefficients. For example, here's how to divide the dividend $6x^4 + 8x^3 + 9x^2 + 3x + 1$ by the divisor $2x^2 + x + 2$ in the ring $\mathbf{R}[x]$ (i.e., the ring used in high school Algebra II).

$$
\begin{array}{r|r}
 & 3x^2 + \frac{5}{2}x + \frac{1}{4} \\ \cline{2-2}
2x^2 + x + 2 & 6x^4 + 8x^3 + 9x^2 + 3x + 1 \\
 & 6x^4 + 3x^3 + 6x^2 \qquad\qquad \\ \cline{2-2}
 & 5x^3 + 3x^2 + 3x + 1 \\
 & 5x^3 + \frac{5}{2}x^2 + 5x \qquad \\ \cline{2-2}
 & \frac{1}{2}x^2 - 2x + 1 \\
 & \frac{1}{2}x^2 + \frac{1}{4}x + \frac{1}{2} \\ \cline{2-2}
 & -\frac{9}{4}x + \frac{1}{2}
\end{array}
\tag{3.4.2}
$$

Again, try to explain why you do what you do there.

Ask Yourself 3.4.2. In (3.4.1), why do you pick $3x^2$, $\frac{5}{2}x$, and $\frac{1}{4}$ as the appropriate multiples of $2x^2 + x + 2$? That is, what general rule tells you how to pick the terms in the quotient? Similarly, what general rule tells us that we stop when we get $-\frac{9}{4}x + \frac{1}{2}$?

Ask Yourself 3.4.2 (and you really did ask yourself that question, right?) deserves an answer at length, along with some other observations.

- The first observation is that long division of polynomials is actually simpler than long division of integers in one important sense: The different "places" (the x^4 place, the x^3 place, etc.) never interact, in that there is no borrowing or carrying. For example, you never get a situation like the one in the last step of (3.4.1), where you have to multiply by 5 to wipe out the 1 (which we now think of as the "digit" 10) left over from a previous step.

- On the down side, the "digits" in each step are now arbitrary real coefficients, so there are negative signs, which are mildly annoying, and denominators, which are deadly. That is, the denominators make it very difficult to do long division with 100% accuracy, whether you're computing by hand, where small errors in denominators can produce disastrous eventual effects; or by machine, where you either have round-off error and its attendant problems, or you have to use arbitrary-precision rational numbers, which are difficult to manage and store.

- The first main observation is, however, that we pick each term in the quotient in order to wipe out the leading term remaining in the dividend. For example, in (3.4.2), we choose $3x^2$ in the quotient because $\dfrac{6x^4}{2x^2} = 3x^2$, we choose $\frac{5}{2}x$ because $\dfrac{5x^3}{2x^2} = \frac{5}{2}x$, and we choose $\frac{1}{4}$ because $\dfrac{\frac{1}{2}x^2}{2x^2} = \frac{1}{4}$.

- The other main observation is that we stop with the remainder $-\frac{9}{4}x + \frac{1}{2}$ because it's at that point that we get the remainder to have a strictly smaller degree than the degree of the divisor.

In fact, those final two observations (wipe out the biggest term remaining, stop when the degree of the reminder is smaller than the degree of the divisor) form the heart of the main point of this section, the Division Theorem for polynomials (Theorem 3.4.4). Before we get to that result, however, let's consider one more example: We again divide the dividend $6x^4 + 8x^3 + 9x^2 + 3x + 1$ by the divisor $2x^2 + x + 2$, but this time in the ring $\mathbf{F}_{13}[x]$ (i.e., polynomials with coefficents that are integers mod 13).

$$
\begin{array}{r|rrrrr}
\multicolumn{1}{r}{} & & & 3x^2 + & 9x + & 10 \\ \cline{2-6}
2x^2+x+2 & 6x^4 + & 8x^3 + & 9x^2 + & 3x + & 1 \\
\multicolumn{1}{r}{} & 6x^4 + & 3x^3 + & 6x^2 & & \\ \cline{2-6}
\multicolumn{1}{r}{} & & 5x^3 + & 3x^2 + & 3x + & 1 \\
\multicolumn{1}{r}{} & & 5x^3 + & 9x^2 + & 5x & \\ \cline{3-6}
\multicolumn{1}{r}{} & & & 7x^2 + & 11x + & 1 \\
\multicolumn{1}{r}{} & & & 7x^2 + & 10x + & 7 \\ \cline{4-6}
\multicolumn{1}{r}{} & & & & x + & 7
\end{array} \tag{3.4.3}
$$

Note that in (3.4.3), the remainder rule works as before (stop when the degree of the remainder is less than the degree of the divisor). Furthermore, it does seem that at each stage, we somehow manage to wipe out the leading term remaining in the dividend. But if you're honest, you should really be asking yourself the following.

Ask Yourself 3.4.3. In (3.4.3), how did we know that the $9x$ and 10 in the quotient would wipe out first the $5x^3$ and then the $7x^2$ in the dividend at the appropriate stages of the calculation? Also, check that the subtractions at each stage are actually correct. Where and how do we use the Congruent Substitution Principle?

The answer to Ask Yourself 3.4.3 is that mod 13 (since we are working in $\mathbf{F}_{13}[x]$), the calculation in (3.4.3) is actually *exactly the same* as the calculation in (3.4.2). However, in the coefficient ring $\mathbf{F}_{13}$, we can represent all of the fractions and negative numbers appearing in (3.4.2) as nonnegative integers (following Convention 3.3.2)! For example, in the second step of (3.4.2), we obtained the term $\frac{5}{2}x$ in the quotient. In the modular equivalent (3.4.3), we note that since $2(7) = 14 = 1$ in $\mathbf{F}_{13}$, 7 is the (multiplicative) inverse of 2 in $\mathbf{F}_{13}$ (Definition 3.2.6), or in other words, $7 = \frac{1}{2}$ in $\mathbf{F}_{13}$. Therefore, $\frac{5}{2} = 5(7) = 35 = 9$ in $\mathbf{F}_{13}$, and we obtain the term $9x$ in the quotient. You should work out for yourself how and why $\frac{1}{4} = 10$, $-\frac{9}{4} = 1$, and so on.

In any case, as promised earlier, we now come to the main point of this section. (Compare the integer Division Theorem 2.3.1 and Signed Division Theorem 2.3.4; the Division Theorem for polynomials can be seen as parallel to both of those previous theorems.)

Theorem 3.4.4 (The Division Theorem for Polynomials)**.** *Let F be a field, and let $a(x)$ and $d(x)$ be polynomials in $F[x]$ with $d(x) \neq 0$. There exist unique $q(x), r(x) \in F[x]$ such that*

$$a(x) = d(x)q(x) + r(x), \qquad \textit{with } \deg(r(x)) < \deg(d(x)). \tag{3.4.4}$$

Note that the degree condition in (3.4.4) includes the possibility that $r(x) = 0$, and therefore, that $\deg(r) = -\infty$.

Proof. The proof of this theorem can be thought of as either a generalization of the "traditional proof" of the Division Theorem 2.3.1 or as a formal version of the polynomial long division procedure discussed above. (More formally, you can also turn this proof into an inducation proof; see Problem 3.4.4.)

Tackling the existence of $q(x)$ and $r(x)$ first, if $\deg(a(x)) < \deg(d(x))$, then we can take $q(x) = 0$ and $r(x) = 0$, so suppose the leading terms of $a(x)$ and $d(x)$ are $a_n x^n$ and $b_k x^k$, respectively, with $k \leq n$ and $b_k \neq 0$. Because F is a field, we can form the fraction $\dfrac{a_n}{b_k}$, let $q_1(x) = \left(\dfrac{a_n}{b_k} x^{n-k}\right)$, and let

$$a_1(x) = a(x) - q_1(x)d(x) = (a_n x^n + \dots) - \left(\frac{a_n}{b_k} x^{n-k}\right)(b_k x^k + \dots) = a_{n-1}x^{n-1} + \dots \tag{3.4.5}$$

Note that since the term $q_1(x)$ is precisely chosen to cancel out the $a_n x^n$ term in $a(x)$, we have that $\deg(a_1(x)) \leq n - 1$. Proceeding similarly to cancel out the leading term of $a_1(x)$, we obtain $a_2(x)$ of degree at most $n-2$, and so on, until we obtain some $a_m(x)$ with degree strictly less than $k = \deg(d(x))$. If we then let $r(x) = a_m(x)$, we see that

$$\begin{aligned} r(x) &= a(x) - q_1(x)d(x) - q_2(x)d(x) - \cdots - q_m(x)d(x) \\ &= a(x) - q(x)d(x), \end{aligned} \tag{3.4.6}$$

where $q(x) = q_1(x) + \cdots + q_m(x)$. Therefore, $a(x) = q(x)d(x) + r(x)$, as desired.

As for the uniqueness part of the theorem, the proof is analogous to the proof of the uniqueness part of the Division Theorem 2.3.1, so we leave it to you (Problem 3.4.5). □

Ask Yourself 3.4.5. Does our proof of Theorem 3.4.4 really only use the abstract definition of field (Definition 3.2.7)? Conversely, do we actually need to assume coefficients in a field to make the proof of Theorem 3.4.4 work? Can we instead make some kind of reasonable assumption on $d(x)$?

Remark 3.4.6. For the purposes of making money (which is, did I mention, our main purpose in this book), the most useful case of the entire discussion above is where $F = \mathbf{F}_2 = \{0, 1\}$. In that case, some of the most complicated parts of our discussion go away completely, since the only "fraction" you need to worry about is $\frac{1}{1} = 1$ — no big deal! So to avoid getting bogged down with fractions too much, our explicitly calculational problems will stick to the primes $p = 2, 3, 5$. It's definitely worth doing some problems with coefficients in $\mathbf{F}_p$ for primes other than $p = 2$, however, for two reasons.

- Odd primes, where $1 \neq -1$, force you to learn the correct signs that occur at various stages of long division.
- Since you can avoid fractions entirely with $F = \mathbf{F}_2 = \{0, 1\}$ and $F = \mathbf{F}_3 = \{0, 1, -1\}$, we'll consider a few problems in $F = \mathbf{F}_5$, where some fractions are needed from time to time. However, if you just remember that

$$\frac{1}{2} = 3 = -2, \qquad \frac{1}{3} = 2, \qquad \frac{1}{4} = 4 = -1, \tag{3.4.7}$$

  that will cover all of your fractional needs.

We now take a bit of a theory digression to prove that a few pieces of high school algebra carry over to polynomials with coefficients in a field F. We begin with the Remainder Theorem and its special case the Factor Theorem.

Corollary 3.4.7 (Remainder Theorem)**.** *Let F be a field, let $f(x) \in F[x]$ be a polynomial, and let α be an element of F. When we divide $f(x)$ by $(x - \alpha)$, the remainder is a constant, namely $r = f(\alpha)$ (the element of F obtained by substituting α for x in $f(x)$).*

Proof. Problem 3.4.6. □

As a special case of the Remainder Theorem, we immediately see the following.

Corollary 3.4.8 (Factor Theorem)**.** *Let F be a field, $f(x) \in F[x]$, and $\alpha \in F$. Then $(x-\alpha)$ divides $f(x)$ (i.e., with a remainder of 0) exactly when $f(\alpha) = 0$.* □

While the Factor Theorem is not too hard to prove, it's quite useful. For example, it implies the following very useful fact about polynomials with coefficients in a field.

Corollary 3.4.9 (Degree n has $\leq n$ solutions)**.** *Let F be a field and let $f(x) \in F[x]$ be a polynomial of degree $n \geq 1$. Then $f(x)$ has at most n distinct zeros in F, i.e., there are at most n distinct elements $\alpha \in F$ such that $f(\alpha) = 0$.*

Corollary 3.4.9 probably doesn't seem that interesting, until you realize that it's not true if you replace F with, say, $\mathbf{Z}/(m)$ for m not prime (Problem 3.1.2). For brevity, we use a proof by induction, but rest assured, if you're not familiar with induction, or even if you're just willing to accept the result, you can just skip the proof and move on.

Proof. For $n = 1$, $f(x) = ax - b$ $(a \neq 0)$ has exactly one zero, namely, $x = \dfrac{b}{a}$.

Proceeding by induction, suppose we know the theorem holds for all polynomials of degree at most n, and suppose $\deg(f(x)) = n+1$. If $f(x)$ has no zeros, then certainly the theorem holds, so suppose $f(\alpha) = 0$ for some $\alpha \in F$. By the Factor Theorem 3.4.8, we have that

$$f(x) = q(x)(x - \alpha) \tag{3.4.8}$$

for some $q(x) \in F[x]$ of degree n. If $f(\beta) = 0$ for some $\beta \in F$, $\beta \neq \alpha$, then

$$0 = q(\beta)(\beta - \alpha), \tag{3.4.9}$$

so dividing both sides by $(\beta - \alpha) \neq 0$ (since F is a field), we see that β is a zero of $q(x)$. By induction, there are at most n possibilities for β, which means that $f(x)$ has at most $n+1$ distinct zeros (the zeros of $q(x)$ along with α). The corollary follows. □

Problems

3.4.1. Solve the following long division with remainder problems in $\mathbf{F}_2[x]$. Don't just compute with ordinary rational numbers and reduce at the very end; instead, represent each coefficient that appears as you go along as one of the integers $\{0, 1\}$.

(a) Divide $x^4 + x^3 + x^2 + x + 1$ by $x + 1$, with remainder.

(b) Divide $x^5 + x^3 + x^2 + x + 1$ by $x^2 + x + 1$, with remainder.

(c) Divide $x^6 + x^4 + x^3 + x + 1$ by $x^2 + 1$, with remainder.

(d) Divide $x^7 + x^5 + x^4 + x^3 + x^2 + x + 1$ by $x^3 + x + 1$, with remainder.

(e) Divide $x^8 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$ by $x^3 + x^2 + 1$, with remainder.

(f) Divide $x^9 + x^7 + x^6 + x^5 + x^4 + x^2 + 1$ by $x^4 + x^3 + 1$, with remainder.

3.4.2. Solve the following long division with remainder problems in $\mathbf{F}_3[x]$. Don't just compute with ordinary rational numbers and reduce at the very end; instead, represent each coefficient that appears as you go along as one of the integers $\{-1, 0, 1, 2\}$. (Remember, sometimes -1 is easier to work with than 2.)

(a) Divide $x^4 + 2x^3 + x^2 + x + 2$ by $x - 1$, with remainder.

(b) Divide $x^4 + x^3 + 2x^2 + 1$ by $x^2 + 2x + 2$, with remainder.

(c) Divide $x^5 + x^4 + x^3 + 2x + 2$ by $2x^2 + 1$, with remainder.

(d) Divide $x^6 + 2x^4 + x^3 + x^2 + x + 1$ by $x^2 - x + 1$, with remainder.

(e) Divide $x^7 + x^6 + 2x^5 + x^4 + 2x^2 + x + 2$ by $2x^2 + x + 2$, with remainder.

(f) Divide $x^7 + 2x^6 + x^5 + x^4 + 2x^3 + x^2 + 1$ by $2x^3 - x^2 + 1$, with remainder.

3.4.3. Solve the following long division with remainder problems in $\mathbf{F}_5[x]$. Don't just compute with ordinary rational numbers and reduce at the very end; instead, represent each coefficient that appears as you go along as one of the integers $\{-2, -1, 0, 1, 2, 3, 4\}$. (Remember, sometimes -1 is easier to work with than 4 and -2 is easier to work with than 3.)

(a) Divide $x^4 + 4x^3 + x^2 + 3x + 2$ by $2x - 2$, with remainder.

(b) Divide $x^4 + 2x^3 + 3x^2 + 2x + 4$ by $2x^2 + x - 1$, with remainder.

(c) Divide $3x^5 + 2x^4 + 4x^3 + 2x + 1$ by $3x^2 + 2$, with remainder.

(d) Divide $2x^6 + 2x^4 + x^3 + 3x^2 + 2x + 1$ by $3x^2 - x + 1$, with remainder.

(e) Divide $3x^7 + x^6 + 2x^5 + x^4 + 2x^2 + x + 2$ by $2x^2 + 3x + 1$, with remainder.

(f) Divide $x^8 + 2x^6 + 3x^5 + 4x^4 + 2x^3 + 4x^2 + 3x + 2$ by $x^3 + 2x^2 + 3x + 4$, with remainder.

3.4.4. Turn the idea of the proof of the existence part of the Division Theorem for Polynomials 3.4.4 into an induction argument.

3.4.5. (*Proves Theorem 3.4.4*) Let F be a field, and suppose that $a(x), d(x)$ are polynomials in $F[x]$ with $d(x) = 0$. Suppose also that $q_i(x)$ and $r_i(x)$ $(i = 1, 2)$ are polynomials in $F[x]$ such that

$$a(x) = q_1(x)d(x) + r_1(x), \qquad a(x) = q_2(x)d(x) + r_2(x), \tag{3.4.10}$$

with $\deg(r_1(x)), \deg(r_2(x)) < \deg(d(x))$. Prove that $r_1(x) = r_2(x)$ and $q_1(x) = q_2(x)$. (Suggestion: How big can $\deg(r_2(x) - r_1(x))$ be?)

3.4.6. (*Proves Corollary 3.4.7*) Let F be a field, let $f(x) \in F[x]$ be a polynomial, and let α be an element of F. Prove that when we divide $f(x)$ by $(x - \alpha)$, the remainder is the constant $r = f(\alpha)$. (Suggestion: Apply the Division Theorem for Polynomials.)

3.5 The Euclidean algorithm for polynomials

In case you haven't guessed where this is all going, here's the upshot:

> For a field F, the ring $F[x]$ works just like the ring of ordinary integers, except replacing the Division Theorem 2.3.1 with the Division Theorem for Polynomials 3.4.4.

For a dramatic example of this principle, in this section, we'll develop the *Euclidean algorithm for polynomials*, which computes the GCD of two polynomials in, essentially, the same way the ordinary Euclidean algorithm computes the GCD of two integers. We begin with the fundamental definitions of divisibility (compare Definitions 2.1.4 and 2.2.1).

Definition 3.5.1. Let F be a field, and let $f(x), g(x), d(x) \in F[x]$ be polynomials with coefficients in F. To say that $d(x)$ *divides* $f(x)$ in $F[x]$ means that $f(x) = q(x)d(x)$ for some $q(x) \in F[x]$. Similarly, to say that $d(x)$ is a *common divisor* of $f(x)$ and $g(x)$ means that $d(x)$ divides both $f(x)$ and $g(x)$.

We note that divisibility has the same fundamental properties for polynomials as it does for integers; see, for example, Problems 3.5.1 and 3.5.2.

Coming to the definition of greatest common divisor, we immediately run into a problem we never really had to face with integers.

Definition 3.5.2. Let F be a field, and let $f(x), g(x) \in F[x]$ be polynomials with coefficients in F, at least one of which is nonzero. To say that $d(x) \in F[x]$ is a *greatest common divisor* of $f(x)$ and $g(x)$ means that $d(x)$ is a common divisor of $f(x)$ and $g(x)$ of highest possible degree.

Ask Yourself 3.5.3. Can you find a field F and some $f(x), g(x) \in F[x]$ such that there is more than one greatest common divisor of $f(x)$ and $g(x)$? How different can you make them?

After a while, you might think of an example like the following: Take $F = \mathbf{R}$, $f(x) = x^2 + 3x + 2$, $g(x) = x^2 - 1$. Then since

$$f(x) = (x+2)(x+1), \qquad g(x) = (x-1)(x+1), \tag{3.5.1}$$

and it turns out that nothing similar is possible with a polynomial of degree 2 or higher, $x + 1$ is a GCD of $f(x)$ and $g(x)$. However, since

$$f(x) = (2x+4)\left(\tfrac{1}{2}x + \tfrac{1}{2}\right), \qquad g(x) = (2x-2)\left(\tfrac{1}{2}x + \tfrac{1}{2}\right), \tag{3.5.2}$$

$\frac{1}{2}x + \frac{1}{2}$ is also a GCD of $f(x)$ and $g(x)$. In fact, a similar calculation shows that $(cx + c)$ is a GCD of $f(x)$ and $g(x)$ for *any* constant $c \neq 0$, so there's definitely more to think about in the way of ambiguities of the GCD than the $\pm$ factor we saw back with integers (Section 2.2).

Unsettling! Fortunately, the polynomial version of the Euclidean algorithm not only gives a method for computing $\gcd(f(x), g(x))$, it also shows that the above ambiguity (nonzero scalar multiples) is the only one that can occur. Even better, having gone to some effort in setting up our framework, we now reap the rewards, in that the Euclidean Algorithm for Polynomials is not just essentially the same algorithm that we had for integers, but is also verified using the same proof and yields the same theoretical consequences.

Algorithm 3.5.4 (The Euclidean Algorithm)**.** Let F be a field, $a(x), b(x) \in F[x]$, and $\deg(a(x)) \geq \deg(b(x)) \geq 0$.

1. *Initialize.* Let $r_{-1}(x) = a(x)$ and $r_0(x) = b(x)$.
2. *Main loop.* For $i = 1, 2, \dots,$ apply the Division Theorem to divide $r_{i-2}(x)$ by $r_{i-1}(x)$ with quotient $q_i(x)$ and remainder $r_i(x)$, or in other words,
$$r_{i-2}(x) = q_i(x)r_{i-1}(x) + r_i(x) \qquad \text{with } \deg(r_i(x)) < \deg(r_{i-1}(x)). \tag{3.5.3}$$
Stop, after N divisions, as soon as you get a remainder $r_N(x) = 0$.
3. *Claim.* The last nonzero remainder $r_{N-1}(x)$ is $\gcd(a(x), b(x))$.

As with the ordinary Euclidean Algorithm 2.4.1, if we write out the iterations of the polynomial Euclidean Algorithm 3.5.4 in order, we get something like:

$$\begin{aligned}
r_{-1}(x) &= q_1(x)r_0(x) + r_1(x) && [\deg(r_1(x)) < \deg(r_0(x))]\\
r_0(x) &= q_2(x)r_1(x) + r_2(x) && [\deg(r_2(x)) < \deg(r_1(x))]\\
r_1(x) &= q_3(x)r_2(x) + r_3(x) && [\deg(r_3(x)) < \deg(r_2(x))]\\
&\vdots\\
r_{N-3}(x) &= q_{N-1}(x)r_{N-2}(x) + r_{N-1}(x) && [\deg(r_{N-1}(x)) < \deg(r_{N-2}(x))]\\
r_{N-2}(x) &= q_N(x)r_{N-1}(x)
\end{aligned} \tag{3.5.4}$$

Same as before, really! Except that the degree decreases each time, not the absolute value.

Example 3.5.5. Let $a(x) = x^4 + 2x^3 + 3x^2 + 4x + 2$ and $b(x) = x^3 + x + 2$ be polynomials in $\mathbf{F}_5[x]$. Applying the Euclidean Algorithm, and keeping in mind that $2(3) = 1$ in $\mathbf{F}_5$ (i.e., 2 and 3 are inverses), we get:

$$\begin{aligned}
x^4 + 2x^3 + 3x^2 + 4x + 2 &= (x + 2)(x^3 + x + 2) + (2x^2 + 3)\\
x^3 + x + 2 &= (3x)(2x^2 + 3) + (2x + 2)\\
2x^2 + 3 &= (x + 4)(2x + 2)
\end{aligned} \tag{3.5.5}$$

Note that we're leaving out a lot of computation at each step, because each step involves a polynomial division with remainder! In any case, since $2x+2$ is the last nonzero remainder, $\gcd(x^4 + 2x^3 + 3x^2 + 4x + 2, x^3 + x + 2) = 2x + 2$, and the algorithm finishes in 3 steps.

After that warmup, we're now ready to verify that the Euclidean Algorithm works.

Theorem 3.5.6. *The polynomial Euclidean Algorithm 3.5.4 terminates after finitely many steps, and the result is actually equal to* $\gcd(a(x), b(x))$.

Proof. Keeping the notation of the Euclidean Algorithm 3.5.4, since degrees of nonzero polynomials are nonnegative integers, that the the number of steps in the Euclidean Algorithm is bounded above by $\deg(r_0) + 1$, so it will eventually stop.

So that was slightly different than before, but the proof of the correctness of the answer is almost exactly the same! I'll leave the details to you, but briefly:

- If $d(x)$ is a common divisor of $a(x)$ and $b(x)$, starting from the top and working down, we see that $d(x)$ divides $r_{N-1}(x)$, meaning $\deg(r_{N-1}(x))$ is as large as the degree of any common divisor (Problem 3.5.6).
- On the other hand, working up from the bottom, we see that $r_{N-1}(x)$ is actually a common divisor of $a(x)$ and $b(x)$ (Problem 3.5.7).

Therefore, by definition, $r_{N-1}(x) = \gcd(a(x), b(x))$. □

Again, our proof of Theorem 3.5.6 yields a fact that seems even more nonobvious than its integer analogue.

Corollary 3.5.7. *For a field F and nonzero $a(x), b(x) \in F[x]$, any common divisor of $a(x)$ and $b(x)$ also divides* $\gcd(a(x), b(x))$ *(as obtained from the Euclidean algorithm).* □

Armed with Corollary 3.5.7, we can now resolve the question of Ask Yourself 3.5.3. We first need an analogue of Definition 2.1.6.

Definition 3.5.8. Let F be a field and $a(x), b(x) \in F[x]$. To say that $a(x)$ and $b(x)$ are *associates*, or that $a(x)$ and $b(x)$ are the same *up to associates*, means that $a(x) = u(x)b(x)$, where $u(x)$ is a unit in $F[x]$. In other words, two polynomials are the same up to associates exactly when each is a nonzero constant multiple of the other.

Corollary 3.5.9. *For a field F and nonzero $a(x), b(x) \in F[x]$, any two greatest common divisors of $a(x)$ and $b(x)$ are associates. In other words,* $\gcd(a(x), b(x))$ *is determined exactly up to associates.*

For example, in Example 3.5.5, we found that $\gcd(x^4 + 2x^3 + 3x^2 + 4x + 2, x^3 + x + 2) = 2x + 2$, but we could just as correctly say that the GCD is $x + 1$, $4x + 4$, or $3x + 3$.

Proof. Let $d(x) = \gcd(a(x), b(x))$ as obtained from the Euclidean algorithm, and let $f(x)$ be some other GCD of $a(x)$ and $b(x)$. By definition of GCD, we must have $\deg(f(x)) = \deg(d(x))$, and by Corollary 3.5.7, $f(x)$ divides $d(x)$. But polynomials that divide each other must be associates (Problem 3.5.8). The corollary follows. □

We finish this section with two Real-life Applications (3.5.14 and 3.5.15) of the polynomial Euclidean Algorithm. First, however, we should introduce the problem those applications solve. To begin with, we define the analogue of prime numbers for polynomials.

Definition 3.5.10. Let F be a field, and let $p(x) \in F[x]$ be a polynomial with coefficients in F. To say that $p(x)$ is *irreducible* means that $p(x)$ is not a unit in $F[x]$ and, if $p(x) = f(x)g(x)$ for $f(x), g(x) \in F[x]$, then one of $f(x), g(x)$ must be a unit in $F[x]$. If $f(x) \in F[x]$ is *not* irreducible, then we say that $f(x)$ is *reducible*.

Ask Yourself 3.5.11. Play with Definition 3.5.10 for a while, as you should do with any significant new definition. How is Definition 3.5.10 an analogue of the usual definition of prime number? Can you come up with some examples of irreducible $p(x) \in \mathbf{Q}[x]$? What does it mean to say that $p(x)$ is not a unit? Why are linear polynomials $(x - \alpha)$ $(\alpha \in F)$ always irreducible? What are some examples of factorizations $p(x) = f(x)g(x)$ for irreducible $p(x)$? Reducible $p(x)$?

Just as any integer can be factored into primes, in essentially only one way, we can prove that:

Theorem 3.5.12 (Unique Factorization for polynomials)**.** *For a field F, any $f(x) \in F[x]$ can be factored into irreducibles in essentially only one way.*

But we'll wait to prove that fact until Section 4.3. Note that Theorem 3.5.12 has the vague term "essentially" in it, and whenever you see a term like that (in math or in life), that's a signal to count the silverware (metaphorically speaking). However, it turns out that there's nothing dishonest about that "essentially"; it just refers to the fact that, for example, $x^2 - 1$ can be factored both as $(x+1)(x-1)$ and $(2x-2)(\frac{1}{2}x+\frac{1}{2})$, but the second factorization isn't really any different than the first. (In fact, we actually see the same kind of factoring ambiguity in the integers: 6 is both $2 \cdot 3$ and $(-3) \cdot (-2)$.)

In any case, we can now state the problem that motivates our two applications of the polynomial Euclidean Algorithm.

Motivating Problem 3.5.13. Given $F = \mathbf{Q}$ or $\mathbf{F}_p$ and $f(x) \in F[x]$, factor $f(x)$ into irreducibles.

For some small cases of Motivating Problem 3.5.13, and of the related problem of finding irreducible polynomials of a given degree, see Problems 3.5.10 and 3.5.11.

You can think of Motivating Problem 3.5.13 as an analogue to the perhaps better-known problem of factoring integers, which has long been studied for its importance to cryptography (see your favorite book on cryptography, such as [reference?]). As we'll see, it's also a stepping stone to solving related cryptography problems like the *discrete logarithm problem*, which we'll discuss in Chapter 10.

Returning to our main topic, we can now explain two real-life applications of the Euclidean Algorithm.

Real-life Application 3.5.14. Many algorithms for factoring a polynomial $f(x) \in F[x]$ begin with the assumption that $f(x)$ is *square-free*, or in other words, that $f(x)$ has no repeated irreducibles in its (unique) factorization. It is a fact that if $f'(x)$ is the derivative of $f(x)$, computed in the usual algebraic manner (even if $F = \mathbf{F}_p$!!), then $\dfrac{f(x)}{\gcd(f(x), f'(x))}$

is a square-free polynomial with the same irreducible factors as $f(x)$ (Problem 3.5.13). Therefore, the polynomial Euclidean Algorithm provides a critical first step in factoring polynomials. (See [reference?].)

Real-life Application 3.5.15. Extending Real-life Application 3.5.14, in the case where $F = \mathbf{F}_p$, *Berlekamp's Algorithm* actually reduces factoring $f(x) \in \mathbf{F}_p[x]$ to a series of GCD computations. As we shall see later, being able to factor polymomials in $\mathbf{F}_p[x]$ is very useful when computing in what is known as a *Galois field*, and in fact, factoring can be used to solve a problem known as the *discrete logarithm problem* in a Galois field. See [reference?].

Problems

3.5.1. Suppose F is a field and $d(x), a(x), b(x) \in F[x]$. Prove that if $d(x)$ divides $a(x)$ and $d(x)$ divides $b(x)$, then $d(x)$ divides $a(x) + b(x)$.

3.5.2. Suppose F is a field and $d(x), a(x), b(x) \in F[x]$. Prove that if $d(x)$ divides $a(x)$, then $d(x)$ divides $a(x)b(x)$.

3.5.3. For the following polynomials $a(x), b(x) \in \mathbf{F}_2[x]$, use the Euclidean Algorithm to find $\gcd(a(x), b(x))$.

(a) $a(x) = x^5 + x^2 + x$, $b(x) = x^4 + x^3 + x$.
(b) $a(x) = x^5 + x^4 + x^2 + x + 1$, $b(x) = x^4 + x^2$.
(c) $a(x) = x^7 + x^6 + x^5 + x^3 + x^2 + x$, $b(x) = x^6 + x^5 + x^4 + x^3 + x$.
(d) $a(x) = x^7 + x^6 + x^5 + x^4 + 1$, $b(x) = x^6 + x^5 + x^4 + x^2 + x + 1$.

3.5.4. For the following polynomials $a(x), b(x) \in \mathbf{F}_3[x]$, use the Euclidean Algorithm to find $\gcd(a(x), b(x))$.

(a) $a(x) = 2x^4 + 2x^3 + x^2 + x$, $b(x) = x^3 + x^2 + 1$.
(b) $a(x) = 2x^5 + 2x^4 + 2x^3 + x^2 + x + 2$, $b(x) = 2x^4 + x^3 + 2$.
(c) $a(x) = x^7 + x^6 + 2x^4 + x^3 + x^2 + 2x$, $b(x) = 2x^6 + x^5 + x^4 + x^3 + x^2 + 2$.
(d) $a(x) = x^7 + x^6 + 2x^5 + x^4 + x^3 + 2x + 1$, $b(x) = x^6 + x^5 + x^4 + 2x^3 + x^2 + 2$.

3.5.5. For the following polynomials $a(x), b(x) \in \mathbf{F}_5[x]$, use the Euclidean Algorithm to find $\gcd(a(x), b(x))$.

(a) $a(x) = x^4 + 4x^3 + x^2 + 2x$, $b(x) = x^2 + 4x + 1$.
(b) $a(x) = 4x^4 + 2x^3 + x^2 + x + 1$, $b(x) = 3x^3 + 4x^2 + 3x + 1$.
(c) $a(x) = 4x^6 + 2x^5 + 2x^4 + x^2 + x$, $b(x) = 4x^5 + 2x^4 + x^3 + 2x$.
(d) $a(x) = x^7 + x^6 + 2x^4 + 4x^3 + 2x$, $b(x) = 2x^6 + 2x^5 + x^4 + x^3 + 3x^2 + 2x + 3$.

3.5.6. (*Proves Theorem 3.5.6*) In the the notation of the Euclidean Algorithm 3.5.4, suppose $d(x)$ is a common divisor of $a(x) = r_{-1}(x)$ and $b(x) = r_0(x)$. Prove that $d(x)$ also divides $r_1(x), r_2(x), \ldots, r_{N-1}(x)$.

3.5.7. (*Proves Theorem 3.5.6*) In the the notation of the Euclidean Algorithm 3.5.4, let $d(x) = r_{N-1}(x)$. Prove that $d(x)$ is a common divisor of $r_{N-1}(x)$ and $r_{N-2}(x)$, $r_{N-2}(x)$ and $r_{N-3}(x)$, ..., $r_0(x) = b(x)$ and $r_{-1}(x) = a(x)$.

3.5.8. (*Proves Corollary 3.5.9*) Let F be a field, and suppose $f(x), g(x) \in F[x]$ are polynomials such that $f(x)$ divides $g(x)$ and $g(x)$ divides $f(x)$. Prove that $f(x)$ and $g(x)$ must have the same degree, and in fact, that $f(x) = cg(x)$, where c is a nonzero constant polynomial (i.e., $c \in F$ and $c \neq 0$).

3.5.9. The goal of this problem is to prove that the Euclidean Algorithm computes $\gcd(a(x), b(x))$ in $O(n)$ time, where n is the larger of $\deg(a(x))$ and $\deg(b(x))$.

(a) Define a *monomial division with remainder* to be an operation of the form

$$f(x) = ax^k d(x) + r(x), \tag{3.5.6}$$

where $\deg(r(x)) < \deg(f(x))$. Explain, by way of an example, how an ordinary division with remainder is made from a number of monomial divisions with remainder.

(b) Prove that the polynomial Euclidean Algorithm ends after n monomial divisions with remainder, where n is the larger of $\deg(a(x))$ and $\deg(b(x))$. (If we take monomimal divisions as our unit of time, this gives $O(n)$.)

3.5.10. This problem establishes an algorithm (not necessarily an efficient one) for checking whether a polynomial of degree 2 or 3 in $\mathbf{F}_p[x]$ is irreducible, without doing any long division.

(a) Let $f(x)$ be a polynomial of degree 2 or 3 in $\mathbf{F}_p[x]$. Prove that if $f(x) = g(x)h(x)$, where neither $g(x)$ nor $h(x)$ is a unit, then one of $g(x)$ or $h(x)$ must have degree 1.

(b) Use part (a) and the Factor Theorem 3.4.8 to create an algorithm that, given a polynomial $f(x)$ in $\mathbf{F}_p[x]$ of degree 2, determines whether $f(x)$ is irreducible without doing any long division.

(c) Use part (a) and the Factor Theorem 3.4.8 to create an algorithm that, given a polynomial $f(x)$ in $\mathbf{F}_p[x]$ of degree 3, determines whether $f(x)$ is irreducible without doing any long division.

3.5.11. In this problem, you are to find all monic irreducible polynomials up to a certain degree in various $\mathbf{F}_p[x]$. Use Problem 3.5.10 in each part.

(a) Find all monic irreducible polynomials up to degree 3 in $\mathbf{F}_2[x]$.

(b) Find all monic irreducible polynomials up to degree 3 in $\mathbf{F}_3[x]$.

(c) Find all monic irreducible polynomials up to degree 2 in $\mathbf{F}_5[x]$.

3.5.12. This problem gives a purely algebraic proof of the product rule and a special case of the chain rule. (finish this later)

(defn derivative)

(a) product rule by grunge it out

(b) $(f(x))^k$ case of chain rule by induction

3.5.13. This problem requires you to remember the product and chain rules, either from calculus or from Problem 3.5.12.

(a) Suppose $f(x) \in \mathbf{Q}[x]$ has a repeated factor in its unique factorization; that is, suppose that
$$f(x) = q(x)p(x)^k \tag{3.5.7}$$
for some $p(x) \in \mathbf{Q}[x]$ and $k \geq 2$. Prove that $(p(x))^{k-1}$ divides $f'(x)$.

(b) Prove that $\dfrac{f(x)}{\gcd(f(x), f'(x))}$ is square-free.

3.6 Bezout's identity for polynomials

Following the tenets of the Church of Abstraction, we now repeat the material on Bezout's Identity over the integers from Section 2.5, entirely analogously. (Really, by mathematicians' standards, almost word for word.)

Theorem 3.6.1 (Bezout's Identity for polynomials)**.** *Let F be a field, and let $a(x), b(x) \in F[x]$ be polynomials with coefficients in F. The equation*
$$a(x)f(x) + b(x)g(x) = \gcd(a(x), b(x)) \tag{3.6.1}$$
has a solution $f(x), g(x) \in F[x]$.

Everything works almost exactly like it does in the proof of Theeorem 2.5.1.

Proof. Retaining the notation of the Euclidean Algorithm 3.5.4, we see that if we define an $F[x]$*-linear combination* of $a(x)$ and $b(x)$ to be a polynomial of the form $a(x)f(x)+b(x)g(x)$ for some $f(x), g(x) \in F[x]$, then our goal is to show that $r_{N-1}(x) = \gcd(a(x), b(x))$ is an $F[x]$-linear combination of $a(x)$ and $b(x)$.

To start, note that $r_{-1}(x) = a(x)$ and $r_0(x) = b(x)$ are each $F[x]$-linear combinations of $a(x)$ and $b(x)$. Then, since we can rewrite each step of the Euclidean Algorithm (see (3.5.4)) as
$$r_n(x) = r_{n-2}(x) - q_n(x)r_{n-1}(x), \tag{3.6.2}$$
working down the equations in (3.5.4), we see that the fact that each $r_n(x)$ is an $F[x]$-linear combination of $a(x)$ and $b(x)$ follows from the same fact for $r_{n-2}(x)$ and $r_{n-1}(x)$. The theorem follows. □

Because there's no need to invent a new name when we have a perfectly good name already, we again call the algorithm in the proof of Bezout's identity for polynomials *Euclidean Rewriting.* And again, to be explicit:

Algorithm 3.6.2 (Euclidean Rewriting)**.** Let F be a field, and let $a(x), b(x) \in F[x]$ be nonzero polynomials with coefficients in F. To solve the equation $a(x)f(x) + b(x)g(x) = \gcd(a(x), b(x))$ for $f(x)$ and $g(x)$:

1. Perform the Euclidean Algorithm 3.5.4 to calculate $\gcd(a(x), b(x))$. We use the notation of Algorithm 3.5.4 in the rest of what follows.

2. Rewrite each step of the Euclidean Algorithm in the form
$$r_i(x) = r_{i-2}(x) - q_i(x)r_{i-1}(x). \tag{3.6.3}$$

3. For i going from 1 to $N-1$, since $r_{i-2}(x)$ and $r_{i-1}(x)$ have already been expressed as $F[x]$-linear combinations of $a(x)$ and $b(x)$, use (3.6.3) to rewrite $r_i(x)$ as a $F[x]$-linear combination of $a(x)$ and $b(x)$. (Note that in the first step, we use the fact that $r_{-1}(x) = a(x)$ and $r_0(x) = b(x)$.)

Example 3.6.3. Consider $a(x) = x^4 + 3x^3 + 2x^2 + x + 3$ and $b(x) = x^3 + 3x^2 + 3x + 3$ in $\mathbf{F}_5[x]$ (i.e., all coefficients are taken mod 5). We use Euclidean Rewriting to find a solution $f(x), g(x) \in \mathbf{F}_5[x]$ to $a(x)f(x) + b(x)g(x) = \gcd(a(x), b(x))$. First, the Euclidean Algorithm gives:
$$\begin{aligned} x^4 + 3x^3 + 2x^2 + x + 3 &= x(x^3 + 3x^2 + 3x + 3) + (4x^2 + 3x + 3) \\ x^3 + 3x^2 + 3x + 3 &= (4x + 4)(4x^2 + 3x + 3) + (4x + 1) \\ (4x^2 + 3x + 3) &= (x + 3)(4x + 1), \end{aligned} \tag{3.6.4}$$
so $\gcd(a(x), b(x)) = 4x + 1$, or up to associates, $x + 4$. Rewriting then gives us:
$$\begin{aligned} 4x^2 + 3x + 3 &= a(x) - xb(x) = a(x) + 4xb(x) \\ 4x + 1 &= b(x) - (4x + 4)(4x^2 + 3x + 3) \\ &= b(x) - (4x + 4)(a(x) + 4xb(x)) \\ &= (x + 1)a(x) + (4x^2 + 4x + 1)b(x). \end{aligned} \tag{3.6.5}$$

Again, as with integers, we have the following consequence of Bezout's Identity, which (as unlikely as it might seem) will again be useful later.

Corollary 3.6.4. *Let F be a field, and let $a(x), b(x), c(x) \in F[x]$ be nonzero polynomials with coefficients in F. The equation*
$$a(x)f(x) + b(x)g(x) = c(x) \tag{3.6.6}$$
has a solution $f(x), g(x) \in F[x]$ if and only if $\gcd(a(x), b(x))$ divides $c(x)$.

Problems

3.6.1. For the following polynomials $a(x), b(x) \in \mathbf{F}_p[x]$, use Euclidean Rewriting to find $f(x), g(x) \in \mathbf{F}_p[x]$ such that $a(x)f(x) + b(x)g(x) = \gcd(a(x), b(x))$. (When $p = 3$, use $-1, 0, 1$ for all coefficients.)

(a) In $\mathbf{F}_3[x]$, $a(x) = x^4 - x^3 + x^2 + x + 1$, $b(x) = x^4 + x + 1$.

(b) In $\mathbf{F}_2[x]$, $a(x) = x^5 + x^4 + x + 1$, $b(x) = x^4 + x^2 + x + 1$.

(c) In $\mathbf{F}_3[x]$, $a(x) = x^5 + x^2 - x + 1$, $b(x) = x^5 + x^4 + x^3 - x^2 + 1$.

(d) In $\mathbf{F}_2[x]$, $a(x) = x^6 + x^5 + x^4 + x^3 + 1$, $b(x) = x^4 + x^2 + 1$.

(e) In $\mathbf{F}_3[x]$, $a(x) = x^6 + x^4 - x^3 + x^2 + x + 1$, $b(x) = x^5 + x^3 - x^2 - x - 1$.

(f) In $\mathbf{F}_2[x]$, $a(x) = x^6 + x^4 + x^3 + x^2 + x + 1$, $b(x) = x^6 + x^5 + x^4 + x^3 + x + 1$.

Chapter 4

Rings and fields

Integers, rationals, complexes, reals
And polynomials and all their ideals
Real-valued functions and similar things
These are a few of my favorite rings

— Cambridge University maths department, mid-80's (sung to the tune of "My Favorite Things" from *The Sound of Music*)

4.1 Why abstraction?

You may be wondering:

> What was that all about? What is this stuff even good for? Can you really make money computing GCDs, or even factoring polynomials?

On the other hand, you might also be wondering:

> What actually is a ring, exactly? How can you tell if something is a ring or not? The Euclidean Algorithm works for integers, and it works for polynomials with coefficients in a field (whatever that is); does it work in any other situations?

If you're in the second camp of wanting more precision and generalization, this chapter is what you've been waiting for. However, if you're still skeptical about the value of theory, let's step back a minute and talk about what happened in the previous two chapters, and how we'll redo all that in this chapter, following our central dogma:

$$\text{Abstraction} \Rightarrow \text{Simplification} \Rightarrow \text{Generalization} \Rightarrow \text{Power}$$

- **Abstraction:** The first step in abstract mathematics, at least logically speaking[*], is to replace the specific examples in which we're interested (integers, polynomials) with a more general object defined by certain axioms (a *ring*, or by the end of this chapter, a *Euclidean domain*).

- **Simplification:** Having axiomatized our favorite examples, we look at what we know about them, strip out all of the inessential features (e.g., details about negatives or the GCD of two polynomials only being defined up to multiplication by a nonzero constant), and either replace those inessentials with another abstraction (e.g., the idea of *associates*) or leave them out entirely. Note that "simpler" doesn't mean easier — abstraction can be more difficult to grasp than specific examples! But in a literal sense, the abstract version of something generally has fewer details and less going on, and can therefore be fairly described as simpler.

- **Generalization:** Because the abstract version of an idea is simpler, it has the potential to apply to examples other than the ones we already know.

- **Power:** That's the power of the abstract approach: If we can understand an idea (in this chapter, GCD and factorization) purely in terms of an axiomatically defined object, then the results we get apply to anything satisfying those axioms.

Full disclosure, this particular case of the axiomatic method (rings and Euclidean domains) doesn't actually apply to that many useful examples other than the ones we've already seen (integers and polynomials), so in that sense, it's not the strongest advertisement for the power part of our central dogma. However, we needed to establish our fundamental examples of $\mathbf{F}_p = \mathbf{Z}/(p)$ and $F[x]$ and we needed to introduce the language of rings and fields, so, well, you gotta start somewhere!

Anyway, sermon over for now, and back to our regularly scheduled program.

4.2 Rings and fields

We still need one preliminary definition before the main event.

Definition 4.2.1. A *binary operation* on a set S takes two inputs in S and produces one output in S, and is usually written as an operator like $+$ or $\cdot$. For example, ordinary addition and multiplication are binary operations on the real numbers.

Note that part of Definition 4.2.1 is that binary operations must have the property of *closure*. That is, for a binary operation to be a legitimate operation on a set S, given two inputs in S, it must actually produce an output in S.

At long last, we now finally define what a ring is.

[*]Truth be told, though, in historical terms, or when you make up new abstract math yourself, the process is much more like what has happened here: First we have one example, then someone notices similarities in another example, and only the does someone think to generalize our known examples in an axiomatic object.

Definition 4.2.2 (Ring)**.** A *ring* is a choice of a set R and a choice of two binary operations, $+$ and $\cdot$, on R, such that the following eight axioms all hold.

- ($+$ *associative*) For any $a, b, c \in R$, $(a+b)+c = a+(b+c)$.
- ($+$ *commutative*) For any $a, b \in R$, $a+b = b+a$.
- (*Zero*) There exists some $0 \in R$ such that for all $a \in R$, $0+a = a = a+0$.
- (*Negatives*) For every $a \in R$, there exists some $-a \in R$ such that $(-a)+a = 0 = a+(-a)$.
- ($\cdot$ *associative*) For any $a, b, c \in R$, $(ab)c = a(bc)$.
- ($\cdot$ *commutative*) For any $a, b \in R$, $ab = ba$.
- (*One*) There exists some $1 \in R$ such that for all $a \in R$, $1a = a = a1$.
- (*Distributive*) For any $a, b, c \in R$, $a(b+c) = ab+ac$ and $(a+b)c = ac+bc$.

We also define $a-b$ to be an abbreviation for $a+(-b)$.

As I told you back in Section 1.1, a mathematical definition tells you what an object *is*, not what it means, and indeed, I wouldn't expect most people reading the nonsense in Definition 1.1 for the first time to have an idea of what's going on there. So here are some ways to interpret Definition 1.1.

- The first thing to note is that the $+$ and $\cdot$ defined on some set R might not be the definitions of $+$ and $\cdot$ you're used to. As an extreme example, we can choose the real numbers $\mathbf{R}$ as the set for our ring and define the sum of $x, y \in \mathbf{R}$ to be $\min(x, y)$ and the product to be $x+y$, and we *almost* get a ring out of that; see Problem 4.2.1. (This is called a *tropical* structure on the real numbers, and is surprisingly useful; see [reference?].)
- The first four axioms basically say that addition and subtraction work for the elements of any ring much as they do for ordinary real numbers.
- The next three axioms basically say that multiplication on its own (without involving addition or subtraction yet) works as it does for real numbers.
- Note that none of the axioms of a ring involve both addition and multiplication except for the distributive law, and so the distributive law is the key to what makes a ring a ring. For example, nowhere in the axioms of a ring R does it say that $0r = 0$ for all $r \in R$; that's actually a consequence of the axioms, and the distributive law plays a prominent role in that fact (Problem 4.2.4) and, well, basically everything else in ring theory.

- To sum up the previous three points, if R is a ring, the axioms of a ring generally allow us to manipulate expressions involving elements of R as if they were expressions in high school algebra, with one major exception. For example, if R is a ring and $a, b, c, d \in R$, then applying the distributive law twice gives us the "FOIL formula":

  $$(a+b)(c+d) = a(c+d) + b(c+d) = ac + ad + bc + bd.$$

  The exception to this general principle is that unless an element a has a multiplicative inverse (see Definition 4.2.9 below), we can't divide by a.

- One hitch in the definition of a ring is that different authors define ring differently. The two biggest variations are:

  – Many authors don't require multiplication in a ring to be commutative. In their terms, our rings are all *commutative rings*; conversely, in the few cases where we want to describe an object that has all of the properties of a ring except commutativity of multiplication, we call that object a *noncommutative ring.*

  – Many authors also do not require a ring to contain a "1" element, whereas we do. Those authors distinguish rings containing a 1 element by calling them *rings with unity.* Probably the best way to think about this question is: Do we want the even integers to be included in the definition of ring? For our purposes, I prefer to require rings to contain 1, as it simplifies many statements, even with the downside that the even integers are no longer a ring under our definition.

  In short: When we say ring, we mean what others might call a "commutative ring with unity".

Example 4.2.3. As the song says at the beginning of the chapter, the integers $\mathbf{Z}$, the rational numbers $\mathbf{Q}$, the real numbers $\mathbf{R}$, and the complex numbers $\mathbf{C}$ are all rings, and you've been taught that (without using the word ring) your entire mathematical life.[†] Also, if R is a ring, you can go through and check that $R[x]$ (Definition 3.3.1) satisfies the axioms of a ring. (Though to be honest, that verification is pretty boring, so we'll just leave it out entirely.)

Example 4.2.4. The integers mod n (Definition 3.1.3) also form a ring, but here the axioms are not as straightforward to check. For example, associativity of addition, while not necessarily difficult to prove, is at the very least a notational mess. (Try it yourself!) So instead of spending a lot of time on that, we'll delay proving that $\mathbf{Z}/(n)$ is a ring until Section 7.2, where the proof will be muich more natural, thanks to some further abstract nonsense to be developed in Chapter 7.

Remark 4.2.5. Now that we've finally defined what a ring is, it seems like a good time to confess that the name ring doesn't come from the "arena" idea we've been discussing since

[†]In fact, you can make a pretty strong case that much of high school algebra is ring theory.

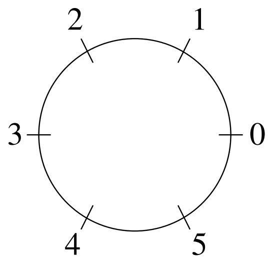

Figure 4.2.1: Where the name "ring" comes from

Section 2.1. The name actually comes from thinking of $\mathbf{Z}/(n)$ as the integers wrapped in a circle, as illustrated for $n = 6$ in Figure 4.2.1. (The arena idea is still a useful way to think about rings, though!)

You may still be asking yourself, now that we've finally defined what a ring is, what can we do with that idea? And the truth is, not much on its own, other than a few very basic things (see Problems 4.2.2–4.2.4). What actually happens is, to understand a given class of examples we want to understand, we figure out the key property that makes all of them work a certain way (the hard part!), create a definition based on that property, and then prove results about rings with that property. For example, we have the following more standard name for a ring with the zero factor property (Definition 3.3.7).

Definition 4.2.6. To say that a ring R is a *domain* (or sometimes, an *integral domain*) means that if $a, b \in R$ and $ab = 0$, then either $a = 0$ or $b = 0$. On the other hand, if $a, b \in R$, both $a \neq 0$ and $b \neq 0$, and we still have $ab = 0$, we say that a and b are *zero divisors* in R.

Theorem 3.3.8 can then be restated as:

Corollary 4.2.7. *If R is a domain and $f(x), g(x) \in R[x]$, then*

$$\deg(f(x)g(x)) = \deg(f(x)) + \deg(g(x)), \tag{4.2.1}$$

where $-\infty$ plus anything is $-\infty$. □

Domains also have not only the Zero Factor Property, but also the more general *cancellation* property.

Theorem 4.2.8. *If R is a domain, $a, b, c \in R$, $a \neq 0$, and $ab = ac$, then $b = c$.*

Proof. Problem 4.2.5. □

The definitions of inverse, unit, and field are as before, though we repeat them here for convenience.

Definition 4.2.9. Let R be a ring. For $a \in R$, an *inverse of* a is some $b \in R$ such that $ab = 1$. Since an element can have only one inverse, we use a^{-1} to denote *the* inverse of a. To say that a is a *unit* in R means that a has an inverse in R.

Definition 4.2.10. A *field* is a ring R in which every nonzero element is a unit and $1 \neq 0$. In other words, to say that a nonzero ring R is a field means that for every $a \neq 0$ in R, there exists some $b \in R$ such that $ab = 1$.

Example 4.2.11. As you well know, the rationals $\mathbf{Q}$ and the real numbers $\mathbf{R}$ are both fields. You may be less familiar with the fact that the complex numbers are a field, which follows because

$$\frac{1}{a+bi} = \frac{a-bi}{(a+bi)(a-bi)} = \left(\frac{a}{a^2+b^2}\right) - \left(\frac{b}{a^2+b^2}\right). \tag{4.2.2}$$

In addition, Corollary 3.2.8 shows that $\mathbf{Z}/(p)$ is a field.

As mentioned earlier, fields have the zero factor property.

Theorem 4.2.12. *If F is a field, then F is a domain.*

Proof. Problem 4.2.6. □

Finally, we at last justify the use of fractions throughout Chapter 3.

Definition 4.2.13. Let R be a ring, and suppose that $a, b \in R$ and b is a unit. We define the fraction $\dfrac{a}{b}$ to be ab^{-1}.

What justifies Definition 4.2.13 is that fractions, as defined there, follow the usual rules of fractions of ordinary numbers.

Theorem 4.2.14. *Let R be a ring, and suppose that $a, b, c, d \in R$ and that b and d are units in R. If fractions are defined as in Definition 4.2.13, then*

$$\frac{ad}{bd} = \frac{a}{b} \qquad \left(\frac{a}{b}\right)\left(\frac{c}{d}\right) = \frac{ac}{bd} \qquad \frac{a}{b} + \frac{c}{d} = \frac{ad+bc}{bd} \tag{4.2.3}$$

Proof. Problem 4.2.7. □

Problems

4.2.1. Define operations $\oplus$ ("tropical addition") and $\otimes$ ("tropical multiplication") on the extended real numbers $\mathbf{R} \cup \{\infty\}$ by

$$\begin{aligned} a \oplus b &= \min(a, b), \\ a \otimes b &= a + b. \end{aligned} \tag{4.2.4}$$

Prove that the tropical operations turn $\mathbf{R} \cup \{\infty\}$ into a *semiring*, i.e., prove that all of the axioms of a ring are satisfied except for negatives, if we take the "zero" element to be ∞ and the "one" element to be 0.

4.2.2. Prove that if R is a ring, and 0 and $0'$ are both zero elements of R, then $0 = 0'$.

4.2.3. Let R be a ring, let a be a nonzero element of R, and suppose that b and c are each inverses of a. Prove that $b = c$. (In other words, multiplicative inverses are unique in a ring.)

4.2.4. Prove that if R is a ring and $a \in R$, then $0a = 0$. (Suggestion: Consider $0 + 0$ and use the Distributive Law.)

4.2.5. (*Proves Theorem 4.2.8*) Prove that if R is a domain (Definition 4.2.6), $a, b, c \in R$, $ab = ac$, and $a \neq 0$, then $b = c$.

4.2.6. Prove that if F is a field (Definition 4.2.10), $a, b \in R$, $ab = 0$, and $a \neq 0$, then $b = 0$. (This is logically equivalent to the Zero Factor Property.)

4.2.7. Let R be a ring, and suppose that $a, b, c, d \in R$ and that b and d are units in R. Carefully justify each step by the axioms of a ring and what has already been proven about inverses, etc.

(a) Prove that $(bd)^{-1} = b^{-1}d^{-1}$. (Suggestion: What times bd equals 1? Use Problem 4.2.3.)

(b) Prove that $\dfrac{ad}{bd} = \dfrac{a}{b}$.

(c) Prove that $\left(\dfrac{a}{b}\right)\left(\dfrac{c}{d}\right) = \dfrac{ac}{bd}$.

(d) Prove that $\dfrac{a}{b} + \dfrac{c}{d} = \dfrac{ad + bc}{bd}$.

4.3 Factoring and Euclidean domains

And now, a dirty secret about abstraction.

> The abstract version of a body of knowledge is the most interesting special case(s), with unnecessary stuff stripped away.

Again, you may object (for instance, in this section!) that there's nothing new in the abstract version, but to review:

- **Simplification:** Isolating what's important in a set of ideas is interesting in its own right.
- **Generalization:** Concepts developed through abstraction apply in other cases.
- **Power:** As a hammer can be used to hit many kinds of nails, an abstract body of knowledge can be used to solve many different problems.

So let's get to it. First, we need to generalize the basic ideas of factoring that you learned in K–12 to domains, i.e., rings with the zero factor property. (The zero factor property avoids a number of annoying complications.)

Definition 4.3.1. Let R be a domain and $a, b, d \in R$. To say that d *divides* a means that $a = qd$ for some $q \in R$. To say that d is a *common divisor* of a and b means that d divides both a and b.

One definition that actually has to be changed substantively for general rings is the definition of "greatest" common divisor, as many rings don't have a natural definition of size. Instead, we incorporate Corollaries 2.4.5 and 3.5.7 as part of the definition.

Definition 4.3.2. Let R be a domain and $a, b \in R$. To say that d is a *greatest common divisor* of a and b means that two things hold:

- d is a common divisor of a and b; and
- If e is a common divisor of a and b, then e divides d.

Definition 4.3.3. To say that $a, b \in R$ are *associates* means that $a = ub$ for some unit $u \in R$.

As with integers and polynomials with field coefficients, the point of units is that they don't affect divisibility in any substantive way; in particular, associate ring elements are the same with respect to their factorization properties (Problem 4.3.1).

Definition 4.3.4. Let R be a domain. To say that $r \in R$ is *irreducible* means that r is not a unit and that if $r = ab$ for $a, b \in R$, then one of a and b must be a unit.

Ask Yourself 4.3.5. What happens when you specialize Definitions 4.3.1–4.3.4 to $R = \mathbf{Z}$ and $R = F[x]$ (F a field)? For example, we get a slightly different definition of a *prime* integer p: To say that $p \in \mathbf{Z}$ is prime means that $p \neq \pm 1$ and if $p = ab$, then one of a or b must be ± 1. Not quite the definition you learned in school — we explicitly allow negative primes, which are equivalent to their associated positive versions.

We can now describe our abstract, generalized version of the Euclidean Algorithm, using a classic move in the world of abstract nonsense: We isolate the one thing that makes the Euclidean Algorithm work and turn it into an axiom (Definition 4.3.7). That way, if R is a domain that satisfies the axiom, then everything we've developed about the Euclidean Algorithm will work in R.

We start with a preliminary definition.

Definition 4.3.6. Let R be a domain. A *size function* on R is a function $\sigma : R \to \mathbf{Z} \cup \{-\infty\}$ such that for all nonzero $r \in R$, $\sigma(r) \geq 0$ and $\sigma(r) > \sigma(0)$. In other words, a size function σ takes elements of R as its inputs, outputs nonnegative integers for nonzero inputs, and has $\sigma(0)$ strictly smaller than any other output.

Definition 4.3.7. A *Euclidean domain* is a domain R with a size function σ that satisfies the following axiom: For $a, d \in R$, $d \neq 0$, there exist $q, r \in R$ such that

$$a = qd + r \qquad \text{with } \sigma(r) < \sigma(d). \tag{4.3.1}$$

In other words, a Euclidean domain is a domain where some version of the Division Theorem holds. (Compare Theorems 2.3.1 and 3.4.4.)

As always, especially with slightly strange-looking definitions like Definitions 4.3.6 and 4.3.7, it's important to check that there are actually examples. (There are a number of well-known, if apocryphal, stories in mathematics about wonderful axiom systems with precisely zero examples.)

Example 4.3.8. The ring $R = \mathbf{Z}$ with the size function $\sigma(r) = |r|$ is a Euclidean domain, because of Theorem 2.3.1. Furthermore, if F is a field, then the ring $F[x]$ with the size function $\sigma(f(x)) = \deg(f(x))$ is a Euclidean domain, because of Theorem 3.4.4.

Unfortunately, and this is a shortcoming of the idea, there are not a lot of other examples of Euclidean domains! (We'll see some good reasons why below, as well as in Section 7.4.) One other prominent example is the *Gaussian integers* $\mathbf{Z}[i]$, defined by

$$\mathbf{Z}[i] = \{a + bi \mid a, b \in \mathbf{Z}\}. \tag{4.3.2}$$

In other words, the Gaussian integers are to the complex numbers as the ordinary integers are to the real numbers. In any case, the point is, if we take

$$\sigma(a + bi) = (a + bi)(a - bi) = a^2 + b^2, \tag{4.3.3}$$

then the Gaussian integers are a Euclidean domain; see Problem 4.3.3.

The point of a Euclidean domain, of course, is that the Euclidean algorithm works in any Euclidean domain; in fact, the general Euclidean domain looks just like it did for integers.

Algorithm 4.3.9 (The Euclidean Algorithm)**.** Let R be a Euclidean domain, and let a and b be nonzero elements of R.

1. *Initialize.* Let $r_{-1} = a$ and $r_0 = b$.
2. *Main loop.* For $i = 1, 2, \dots$, apply the definition of Euclidean domain (Definition 4.3.7) to divide r_{i-2} by r_{i-1} with quotient q_i and remainder r_i, or in other words,

$$r_{i-2} = q_i r_{i-1} + r_i \qquad \text{with } \sigma(r_i) < \sigma(r_{i-1}). \tag{4.3.4}$$

   Stop, after N divisions, as soon as you get a remainder $r_N = 0$.
3. *Claim.* The last nonzero remainder r_{N-1} is a greatest common divisor of a and b.

Theorem 4.3.10. *Let R be a Euclidean domain, and let a and b be nonzero elements of R.*

- *The Euclidean Algorithm 4.3.9 terminates after finitely many steps, and the result $r_{N-1} = gcd(a,b)$ is actually a greatest common divisor of a and b.*

- *(Bezout) There exist $x, y \in R$ such that*

$$ax + by = \gcd(a,b). \tag{4.3.5}$$

Since we have seen this proof twice already, I'll leave most of the details to you.

Proof. Keeping the notation of the Euclidean Algorithm 4.3.9, since the size of r_i decreases by at least 1 at each step, the number of steps in the Euclidean Algorithm is bounded above by $\sigma(r_0) + 1$, so it will eventually stop. The other details are essentially the same; see Problem 4.3.4. □

Just so this chapter isn't a complete repeat, here's a new fact about Euclidean domains.

Theorem 4.3.11 (Unique factorization in Euclidean domains)**.** *Let R be a Euclidean domain, and let a be a nonzero, non-unit element of R. Then a can be factored as a product of irreducible elements of R in essentially one way. That is, $a = p_1 \cdots p_k$ for some irreducible elements $p_i \in R$; and if*

$$a = p_1 \cdots p_k = q_1 \cdots q_r \tag{4.3.6}$$

with all p_i, q_j irreducible, then $k = r$, and we can rearrange the q_j so that for $1 \leq i \leq k$, we have that p_i is an associate of q_i.

Now Theorem 4.3.11 isn't completely new to you, as you probably learned the case $R = \mathbf{Z}$ back in grade school, and you've certainly been assuming that it's true in the case $R = \mathbf{R}[x]$ since high school algebra. What's perhaps more interesting is that there are very reasonable-looking domains R where the conclusion of the theorem *doesn't* hold. For example, take

$$R = \mathbf{Z}[\sqrt{-5}] = \left\{ a + b\sqrt{-5} \mid a, b \in \mathbf{Z} \right\}. \tag{4.3.7}$$

Then

$$6 = 2 \cdot 3 = (1 + \sqrt{-5})(1 - \sqrt{-5}), \tag{4.3.8}$$

where all four elements $2, 3, (1 + \sqrt{-5}), (1 - \sqrt{-5})$ are irreducible. In other words, 6 can be factored in R in two essentially different ways. Weird! So I hope it seems at least a little more remarkable that unique factorization works in such new rings as $\mathbf{F}_p[x]$ (p prime).

In any case we're not quite ready to prove Theorem 4.3.11 yet, as we'll need to build up more abstraction in Chapter 7; even then, we'll only work out most of the details in Appendix A. For now, it's enough for you to get an idea of exactly what Theorem 4.3.11 means; see Problem 4.3.6.

Problems

4.3.1. associates have same divisibility properties

(a) divides, then associate divides

(b) associates of irreducibles are irreducibles

4.3.2. Prove GCD unique up to associates, if one exists.

4.3.3. Prove Gaussian integers are an ED.

4.3.4. details of euclidean algorithm

4.3.5. details of bezout

4.3.6. This problem is meant to give you a better idea of what unique factorization really means in $\mathbf{F}_p[x]$.

(a) Note that in $\mathbf{F}_{13}[x]$,

$$6x^2 - 6x + 6 = (3x - 4)(2x + 5) = (2)(7)(3x - 4)(2x + 5). \tag{4.3.9}$$

However, Theorem 4.3.11 says that any two factorizations of $6x^2 - 6x + 6$ into irreducibles must use the same number of irreducible factors. Explain why this isn't a contradiction.

(b) Note that in $\mathbf{F}_{11}[x]$,

$$x^2 + x + 2 = (x - 4)(x + 5) = (4x - 2)(3x - 1). \tag{4.3.10}$$

However, Theorem 4.3.11 supposedly says that any two factorizations of x^2+x+2 into irreducibles must use the "same" irreducibles. Explain why this isn't a contradiction.

Chapter 5

Linear algebra

It is upon the length and breadth and span of a body sustained in muscular action that dance invokes its image.

— Merce Cunningham

All the women, who are independent
Throw your hands up at me

— Destiny's Child, "Independent Women — Part 1"

ALL YOUR BASE ARE BELONG TO US

— the arcade video game *Zero Wing*

5.1 A data compression problem

Linear algebra is such a useful (and lucrative) field of math to know that there are a zillion problems we could use as motivation for the rest of this chapter. To pick one, let's start with our main use of linear algebra in this book.

Consider the collection $\mathcal{C} = \{a, b, c, d, e, f, g, h\}$ of *bitstrings* (strings of 0s and 1s):

$$\begin{array}{llll} a = 00000 & b = 00111 & c = 01011 & d = 01100 \\ e = 10010 & f = 10101 & g = 11001 & h = 11110 \end{array} \tag{5.1.1}$$

Looks pretty random, right? Well, it's not obvious, but it turns out that $\mathcal{C}$ does have one interesting *closure* property: If you add any two strings together by adding their corresponding *bits* (mod 2)[*], you get another bitstring in $\mathcal{C}$. For example,

$$b + e = 00111 + 10010 = 10101 = f, \tag{5.1.2}$$

or for a less interesting example, $h + h = 00000 = a$. It follows that, for example, if we only need to know the list of bitstrings in $\mathcal{C}$, we could leave a and f off of the list and deduce that

[*]Or if you're a computer scientist, taking the XOR of the bits.

they're in $\mathcal{C}$ because of the closure property. Think of this as a sort of data compression property — we didn't need that whole list to define $\mathcal{C}$.

Anyway, this begs the question:

Motivating Problem 5.1.1. What is the *smallest* number of bitstrings of $\mathcal{C}$ from which we could recover all of $\mathcal{C}$, just by knowing that $\mathcal{C}$ has the closure property?

Ask Yourself 5.1.2. Just knowing that $b, c, e, g \in \mathcal{C}$, can you recover all of the other bitstrings in $\mathcal{C}$? How about if you just know that $b, c, e \in \mathcal{C}$? How about if you just know that $b, c \in \mathcal{C}$? Try adding up some bitstrings and see!

Call a set of bitstrings $\mathcal{B}$ a *minimal recovery set* if we can recover $\mathcal{C}$ from $\mathcal{B}$ and the closure property, but if you remove any element of $\mathcal{B}$, this is no longer true. We'll see later (Problem 5.6.5) that $\{b, c, e\}$ is actually a minimal recovery set, as is $\{d, f, h\}$. On the other hand, you can't actually recover all of $\mathcal{C}$ from $\{c, e, g\}$ and the closure property (try it!). That begs a whole bunch of other questions:

Motivating Problem 5.1.3. How can you tell if you can recover $\mathcal{C}$ from a given set of bitstrings? Is there a more efficient method than adding bitstrings until we stop getting something new?

Motivating Problem 5.1.4. What's an efficient way to tell if $\mathcal{B}$ is a *minimal* recovery set?

Motivating Problem 5.1.5. Are all minimal recovery sets the same size, or are some minimal recovery sets size 3 and others (say) size 2?

Motivating Problem 5.1.6. Here's a challenge: Find all 28 possible minimal recovery sets for $\mathcal{C}$. Minimal recovery sets never need a, so that's a start, but there are still 127 nonempty subsets of $\{b, c, d, e, f, g, h\}$, so brute force might take a while. Is there a better way to find those minimal subsets?

Rest assured, by Section 5.6, we'll be able to resolve all of these questions. (See Problem 5.6.5 for a full answer.) What may be a little surprising, however, if you're familiar with linear algebra, is that the above motivating problems are actually the foundational questions of linear algebra.

So with all that in mind, and especially if you *haven't* seen any linear algebra before, it's time for some pictures.

5.2 Linear algebra in three cartoons

Before we get into linear algebra in the abstract version we'll use to make money, let's take a big-picture look at linear algebra in ordinary 3-space, a.k.a. $\mathbf{R}^3$. (Again, if you've never seen linear algebra before, fear not; we really will cover everything starting from scratch.)

The point of linear algebra in $\mathbf{R}^3$ is to study the *vector space* $\mathbf{R}^3$ and its subspaces. There are four types of subspaces of $\mathbf{R}^3$, namely, the subspace containing just 0, lines

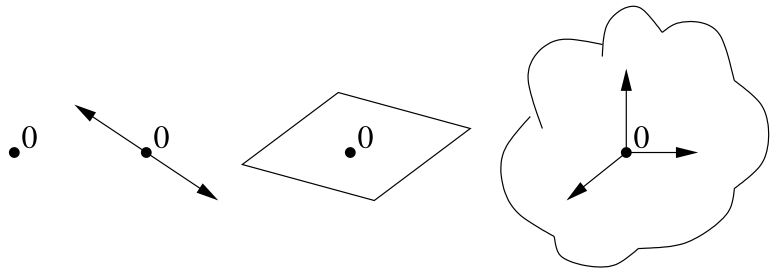

Figure 5.2.1: Subspaces of $\mathbf{R}^3$

through the origin, planes through the origin, and $\mathbf{R}^3$ itself, as shown in Figure 5.2.1. (The cloud at the end is meant to represent all of $\mathbf{R}^3$.) These spaces can be characterized by their *dimensions*, which are 0, 1, 2, and 3, respectively.

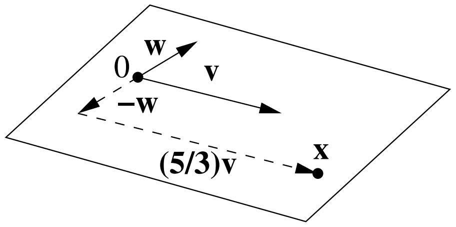

Figure 5.2.2: Bases define coordinates

Note that a subspace of dimension greater than 0 contains infinitely many *vectors*, represented in Figure 5.2.1 as points in $\mathbf{R}^3$. To analyze a subspace W algebraically, we need to be able to describe W by a finite data set, called a *basis*. A basis describes a subspace W by defining coordinates on W. For example, Figure 5.2.2 shows a 2-dimensional subspace W of $\mathbf{R}^3$, which is described by the indicated vectors $\mathbf{v}$ and $\mathbf{w}$ because every point in W is equal to $a\mathbf{v} + b\mathbf{w}$ for some $a, b \in \mathbf{R}$, as illustrated for $\mathbf{x} \in W$ and $a = \frac{5}{3}$, $b = -1$. Note that W is 2-dimensional and we need exactly 2 vectors to describe W through coordinates; in fact, we will later define the dimension of a subspace W to be the smallest number of vectors required to describe W in this way.

And here we come to an important and often confusing point:

Even though there are only 2 vectors in a basis for a 2-dimensional subspace W of $\mathbf{R}^3$, W contains far more than 2 vectors. In fact, W contains infinitely many vectors!

Think of it as infinite data compression, a more extreme version of the data compression we saw in Section 5.1: We're encoding an infinite data set (the set of vectors in W) into a finite set, and we're even trying to make that finite set as small as possible.

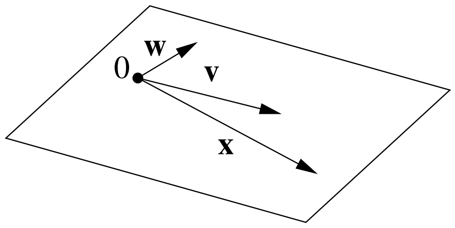

Figure 5.2.3: A dependent set of vectors

Returning to bases, the property of being able to express every vector in W as a *linear combination* $a\mathbf{v} + b\mathbf{w}$ is called *spanning*. More precisely, if that property holds, we say that $\{\mathbf{v}, \mathbf{w}\}$ *spans* W. But here's one tricky thing about that: If $\{\mathbf{v}, \mathbf{w}\}$ spans W, we can add another vector $\mathbf{x}$ and still be able to express every vector of W as a linear combination $a\mathbf{v} + b\mathbf{w} + c\mathbf{x}$; see Figure 5.2.3. However, the resulting coordinates (a, b, c) will no longer be unique, because the vectors $\mathbf{v}$, $\mathbf{w}$, and $\mathbf{x}$ are *linearly dependent*, or in other words, have some linear relationship among themselves. (In fact, $\frac{5}{3}\mathbf{v} - \mathbf{w} - \mathbf{x} = \mathbf{0}$, exactly as in Figure 5.2.2.) So to provide maximally useful coordinates, a spanning set also needs to be *linearly independent*, like $\{\mathbf{v}, \mathbf{w}\}$. And that's exactly what a *basis* is:

> A basis for a subspace W is a linearly independent set that also spans W. You can use a basis for W to describe exactly which vectors are contained in W in terms of coordinates.

See Theorem 5.3.15 for a precise version of the above idea.

So that's it, that's linear algebra! Well OK, of course there's much more to it, especially in terms of calculations, which are the part of linear algebra you're most likely to remember if you've seen linear algebra before. But that's the foundational idea: describing subspaces in terms of bases. Anyway, we now go on to doing the same thing, but with an arbitrary field F instead of $\mathbf{R}$, and with definitions instead of pictures — the advantage being that thanks to abstraction, we'll be able to solve problems like the ones posed in Section 5.1. Still, please do refer back to this section if and when you feel like you're drowning in abstraction.

5.3 The foundations of linear algebra

The main part of linear algebra that we'll need is the foundational theoretical ideas. Don't get me wrong — the calculations are important! But for us, the calculations are motivated by theory, so we'll start there. First, we describe the "arena" in which we do linear algebra.

Definition 5.3.1. Let F be a field and $n \in \mathbf{N}$. We define

$$F^n = \left\{ \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \middle| \, x_i \in F \right\}, \tag{5.3.1}$$

and we call the elements of F^n *vectors*; in particular, we call $\mathbf{0} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$ the *zero vector* in F^n. When we work with vectors, we often call the elements of F itself *scalars*.

We also define two operations on F^n. The first is *vector addition*: For $\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \in F^n$, we define

$$\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} x_1 + y_1 \\ \vdots \\ x_n + y_n \end{bmatrix}. \tag{5.3.2}$$

And the second is *scalar multiplication*: For $a \in F$ and $\begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in F^n$, we define

$$a \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} ax_1 \\ \vdots \\ ax_n \end{bmatrix}. \tag{5.3.3}$$

Example 5.3.2. For us, the most important case of F^n is where $F = \mathbf{F}_2$. The reason is, since a vector in F^n is a column of n 0's and 1's, we can identify F^n with the set of all *bitstrings of length* n, e.g., we can think of 001011101, a bitstring of length 9, as the element $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$ of $\mathbf{F}_2^9$. You would not believe how important and useful that idea is! (Though Section 5.1 starts to give an idea of why this case is important, and I hope you'll agree after a few more chapters.)

As you saw in the first cartoon of Section 5.2, our main object of study in F^n is the following concept.

Definition 5.3.3. For $n \in \mathbf{N}$, a *subspace* of F^n is a subset W of F^n that satisfies three properties:

1. W contains the zero vector $\mathbf{0}$;
2. (Closed under $+$) For any $\mathbf{v}, \mathbf{w} \in W$, we have $\mathbf{v} + \mathbf{w} \in W$; and

3. (Closed under scalar multiplication) For any $\mathbf{v} \in W$ and $a \in F$, we have $a\mathbf{v} \in W$.

At this juncture, you may want to review how to prove closure of a set under an operation (Section 1.3.5).

Example 5.3.4. The two extreme examples of subspaces of F^n are, on the one hand, F^n itself, and on the other hand, the *zero subspace* $\{\mathbf{0}\}$.

Example 5.3.5. Let $\mathcal{C}$ be the subset of $\mathbf{F}_2^5$ discussed in Section 5.1. There, we observered that $\mathcal{C}$ contains 00000 and is closed under addition (bitwise sum mod 2). Since the only scalars in $\mathbf{F}_2$ are 0 and 1, $\mathcal{C}$ is also closed under scalar multiplication, so by Definition 5.3.3, $\mathcal{C}$ is actually a a sub*space* of $\mathbf{F}_2^5$.

Before getting to the foundational ideas of *span*, *linear independence*, *basis*, and *dimension*, we have the following definition, which we use in the definition of all of those ideas.

Definition 5.3.6. Let $\mathbf{v}_1, \dots, \mathbf{v}_k$ be vectors in F^n. An *F-linear combination* of $\mathbf{v}_1, \dots, \mathbf{v}_k$, or when the context is clear, simply a *linear combination* of $\mathbf{v}_1, \dots, \mathbf{v}_k$, is a vector of the form

$$a_1\mathbf{v}_1 + \cdots + a_k\mathbf{v}_k \tag{5.3.4}$$

for some scalars $a_i \in F$. The scalars a_i in the linear combination (5.3.4) are called the *coefficients* of that linear combination. If all of the coefficients a_i are equal to 0, we call (5.3.4) a *trivial* linear combination; otherwise, if at least one of the $a_i \neq 0$, we call (5.3.4) a *nontrivial* linear combination.

As in $\mathbf{R}^3$ (see Section 5.2), our main goal is to describe subspaces in terms of a smaller set of data, as follows.

Definition 5.3.7 (Span)**.** Let $\mathbf{v}_1, \dots, \mathbf{v}_k$ be vectors in F^n. The *span* of $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is defined to be the set of all F-linear combinations of $\mathbf{v}_1, \dots, \mathbf{v}_k$. In other words,

$$\operatorname{span}\{\mathbf{v}_1, \dots, \mathbf{v}_k\} = \{a_1\mathbf{v}_1 + \cdots + a_k\mathbf{v}_k \mid a_i \in F\}. \tag{5.3.5}$$

Conversely, if W is a subspace of F^n, to say that $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ *spans* W, or that $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is a *spanning set* for W, means that $W = \operatorname{span}\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$. In other words, $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ spans W exactly when both of the following conditions hold:

1. Each of the vectors $\mathbf{v}_1, \dots, \mathbf{v}_k$ is contained in W.
2. Every $\mathbf{x} \in W$ is a linear combination of $\mathbf{v}_1, \dots, \mathbf{v}_k$.

As a special case, we define the span of the empty set of vectors to be the zero subspace $\{\mathbf{0}\}$.

It's important to note that the span of a subset of F^n actually is a sub*space* of F^n:

Theorem 5.3.8. *If* $\mathbf{v}_1, \dots, \mathbf{v}_k$ *are vectors in* F^n*, then* $\operatorname{span}\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ *is a subspace of* F^n*.*

Proof. Problem 5.3.2. □

Remark 5.3.9. Note that if the vectors $\mathbf{v}_1, \dots, \mathbf{v}_k$ are contained in a subspace W, the condition $W = \operatorname{span}\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ of Definition 5.3.7 boils down to the statement: "If $\mathbf{x} \in W$, then $x = a_1\mathbf{v}_1 + \dots + a_k\mathbf{v}_k$ for some $a_i \in F$." So, for example, if you know that $\{\mathbf{u}, \mathbf{v}, \mathbf{w}\}$ spans a subspace W, then you know that $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ are in W, and if $\mathbf{x} \in W$, then $\mathbf{x} = a\mathbf{u} + b\mathbf{v} + c\mathbf{w}$ for some $a, b, c \in F$.

On the other hand, suppose we only know that $\mathbf{u}, \mathbf{v}, \mathbf{w} \in W$ and you then want to prove that $\{\mathbf{u}, \mathbf{v}, \mathbf{w}\}$ spans W, given certain assumptions. In that case, your proof should go something like this:

> **Assume:** (certain assumptions)
> **Assume:** $\mathbf{x} \in W$.
> (stuff)
> **Conclusion:** $\mathbf{x} = a\mathbf{u} + b\mathbf{v} + c\mathbf{w}$ for some $a, b, c \in F$.

Next, as we saw in the last cartoon of Section 5.2, if a spanning set satisfies some kind of linear relationship, then it doesn't produce unique "coordinates". We therefore want spanning sets to be *linearly independent*, or more precisely:

Definition 5.3.10 (Linear independence)**.** Let $\mathbf{v}_1, \dots, \mathbf{v}_k$ be vectors in F^n. To say that $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is *linearly dependent* means that some nontrivial linear combination (Definition 5.3.6) of $\mathbf{v}_1, \dots, \mathbf{v}_k$ is equal to $\mathbf{0}$. In other words, $\mathbf{v}_1, \dots, \mathbf{v}_k$ is linearly dependent exactly when

$$a_1\mathbf{v}_1 + \dots + a_k\mathbf{v}_k = \mathbf{0} \tag{5.3.6}$$

for some choice of coefficients $a_1, \dots, a_k \in F$, not all of which are 0. A relationship like (5.3.6) with not all coefficients equal to 0 is called a *linear dependency.*

Conversely, to say that $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is *linearly independent* means that the only way to get

$$a_1\mathbf{v}_1 + \dots + a_k\mathbf{v}_k = \mathbf{0} \tag{5.3.7}$$

for coefficients $a_i \in F$ is if all of the $a_i = 0$ (i.e., $a_1 = \dots = a_k = 0$). In other words, to say that $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is linearly independent means that if (5.3.7) holds, then we must have all $a_i = 0$.

Again as a special case, we declare the empty set to be linearly independent.

Remark 5.3.11. While the final, "if-then" version of the definition of linear independence is not the most natural statement in the world, it works great for proofs. On the one hand, if you know that $\{\mathbf{u}, \mathbf{v}, \mathbf{w}\}$ is linearly independent, then given any linear combination of $\{\mathbf{u}, \mathbf{v}, \mathbf{w}\}$ that is equal to $\mathbf{0}$, you automatically know that the coefficients of that linear combination must all be equal to 0.

On the other hand, if you want to prove that $\{\mathbf{u}, \mathbf{v}, \mathbf{w}\}$ is linearly independent under certain assumptions, your proof should go something like this:

Assume: (certain assumptions)
Assume: $a\mathbf{u} + b\mathbf{v} + c\mathbf{w} = \mathbf{0}$ for some $a, b, c \in F$.
(stuff)
Conclusion: $a = 0$, $b = 0$, and $c = 0$.

Putting the ideas of spanning and linear independence together, we get:

Definition 5.3.12. Let W be a subspace of F^n. A *basis* for W is a linearly independent subset $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ of W that also spans W. To say that W has *dimension* k means that W has a basis with k vectors in it.

Example 5.3.13. Since we defined the span of the empty set to be the zero subspace, and we declared the empty set to be linearly independent, the empty set is a basis for the zero subspace. Since that basis contains no vectors, the zero space has dimension 0 (whew!).

At the other extreme, in F^n, let $\mathbf{e}_i$ be the vector with its ith entry equal to 1, and all other entries equal to 0. Then $\{\mathbf{e}_1, \dots, \mathbf{e}_n\}$ is a basis for F^n (Problem 5.3.3), which means that $\dim F^n = n$.

Example 5.3.14. Returning to the subspace $\mathcal{C}$ discussed in Section 5.1 and Example 5.3.5, it turns out that the subset $\{b, c, e\}$ of $\mathcal{C}$ is a basis for $\mathcal{C}$. (How could you go about checking this by brute force, using the fact that there are only finitely many vectors in $\mathcal{C}$ and only finitely many linear combinations of $\{b, c, e\}$? Try it!)

As described in Section 5.2 (see especially Figure 5.2.2), one way to understand a basis for a subspace W is that a fixed choice of basis gives a unique set of F-valued coordinates for every vector in W. More precisely:

Theorem 5.3.15. *Let W be a subspace of F^n, and let $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ be a basis for W. Then for every $\mathbf{w} \in W$, there exists a unique choice of $a_1, \dots, a_k \in F$ such that*

$$\mathbf{w} = a_1\mathbf{v}_1 + \dots + a_k\mathbf{v}_k. \tag{5.3.8}$$

We call the (unique) $a_1, \dots, a_k$ in (5.3.8) the *coordinates of* $\mathbf{w}$ *with respect to the basis* $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$. If the basis $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is understood, we just call $a_1, \dots, a_k$ the *coordinates of* $\mathbf{w}$. In these terms, Theorem 5.3.15 says that the vectors in W can be put in one-to-one correspondence with k-tuples $a_1, \dots, a_k$ of elements of F. In other words, vectors in W can be described uniquely by their coordinates $a_1, \dots, a_k$. (Compare the discussion at the end of Section 5.2.)

Proof. Because $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is a basis for W, $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ spans W, which means that (by definition!) we can find $a_1, \dots, a_k \in F$ to make (5.3.8) work. Therefore, all we have to resolve is the uniqueness of $a_1, \dots, a_k$.

So suppose that

$$\begin{aligned} \mathbf{w} &= a_1\mathbf{v}_1 + \dots + a_k\mathbf{v}_k \\ &= a_1'\mathbf{v}_1 + \dots + a_k'\mathbf{v}_k \end{aligned} \tag{5.3.9}$$

for $a_1, \dots, a_k, a'_1, \dots, a'_k \in F$. Then

$$(a_1 - a'_1)\mathbf{v}_1 + \cdots + (a_k - a'_k)\mathbf{v}_k = 0. \tag{5.3.10}$$

However, since $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is linearly independent, by definition, the only way that (5.3.10) can happen is if $a_1 - a'_1 = 0$, $a_2 - a'_2 = 0$, $\dots$, $a_k - a'_k = 0$. In other words, $a_i = a'_i$ for each i from 1 to k, which means that the supposedly different coordinates $a_1, \dots, a_k$ and $a'_1, \dots, a'_k$ are actually the same. The theorem follows. $\square$

dimension	coordinates
basis	
span	lin indep
linear combinations	

Figure 5.3.1: The foundations of linear algebra

So great, we've established the foundations of linear algebra in short order, as shown in Figure 5.3.1. Except:

Ask Yourself 5.3.16. If you think hard about the definition of dimension, you might have the following nightmares.

- Is it possible for a subspace W to have one basis with 5 vectors and another basis with 7 vectors? In other words, is it possible for the dimension of W to be both 5 and 7?
- Is it possible for F^8 to contain a subspace of dimension 10? In other words, is it possible for a smaller space to have a larger dimension?
- Can we find a subspace of F^n that doesn't have a basis at all?

Rest assured, none of the scenarios in Ask Yourself 5.3.16 actually occur! But the more you think about it, the less obvious that is.

Even once we determine that the nightmares never happen, there are also some computational problems that need to be solved.

Motivating Problem 5.3.17. Given a subspace W of F^n and vectors $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ that span W, how can we check that $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is a basis for W?

Motivating Problem 5.3.18. Given a subspace W of F^n, how can we find a basis for W?

In any case, it turns out that the same tool serves both to dispel the nightmares of Ask Yourself 5.3.16 and also to answer Motivating Problems 5.3.17–5.3.18: computational linear algebra, to which we turn next (Sections 5.4 and 5.5).

Problems

5.3.1. Let F be a field, and let F^n and $\mathbf{0}$ be as described in Definition 5.3.1. Prove that the following properties hold for any vectors $\mathbf{v}, \mathbf{w}, \mathbf{x} \in F^n$ and scalars $a, b \in F$.

(a) $(\mathbf{v} + \mathbf{w}) + \mathbf{x} = \mathbf{v} + (\mathbf{w} + \mathbf{x})$.

(b) $\mathbf{v} + \mathbf{w} = \mathbf{w} + \mathbf{v}$.

(c) $\mathbf{v} + \mathbf{0} = \mathbf{v}$.

(d) There exists some vector $(-\mathbf{v})$ such that $\mathbf{v} + (-\mathbf{v}) = \mathbf{0}$.

(e) $a(b\mathbf{v}) = (ab)\mathbf{v}$.

(f) $1\mathbf{v} = \mathbf{v}$.

(g) $a(\mathbf{v} + \mathbf{w}) = a\mathbf{v} + a\mathbf{w}$.

(h) $(a + b)\mathbf{v} = a\mathbf{v} + b\mathbf{v}$.

5.3.2. (*Proves Theorem 5.3.8*) Suppose $\mathbf{v}_1, \dots, \mathbf{v}_k$ are vectors in F^n, and let

$$W = \operatorname{span}\{\mathbf{v}_1, \dots, \mathbf{v}_k\} = \{a_1\mathbf{v}_1 + \cdots + a_k\mathbf{v}_k \mid a_i \in F\}. \tag{5.3.11}$$

In other words, a vector $\mathbf{x}$ of F^n is in W exactly when

$$\mathbf{x} = a_1\mathbf{v}_1 + \cdots + a_k\mathbf{v}_k \tag{5.3.12}$$

for some choice of coefficients $a_i \in F$.

(a) Explain why $\mathbf{0}$ is in W. (Suggestion: What a_i should you choose to get $\mathbf{0}$ in (5.3.12)?)

(b) Suppose $\mathbf{x}$ and $\mathbf{y}$ are in W. Explain why $\mathbf{x} + \mathbf{y}$ must be in W. (Suggestion: Both $\mathbf{x}$ and $\mathbf{y}$ have a form similar to (5.3.12); what does $\mathbf{x} + \mathbf{y}$ look like? You may also want to review Section 1.3.5.)

(c) Suppose $\mathbf{x}$ is in W and $a \in F$. Explain why $a\mathbf{x}$ must be in W.

5.3.3. Let $\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$, $\mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$, and $\mathbf{e}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$. Prove that $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ is a basis for $\mathbf{F}_{13}^3$, using the following steps.

(a) Prove that if $\mathbf{x} \in F^3$, then $\mathbf{x}$ is a linear combination of $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$. (See Remark 5.3.9.)

(b) Prove that $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ is linearly independent, using the definition of linear independence. (See Remark 5.3.11.)

5.3.4. Consider the subset

$$W = \left\{ \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} \in \mathbf{F}_{17}^3 \;\middle|\; x, y \in \mathbf{F}_{17} \right\}. \tag{5.3.13}$$

of $\mathbf{F}_{17}^3$. Find a basis $\mathcal{B}$ for W, and prove that $\mathcal{B}$ is a basis for W, using the following steps.

(a) First, guess what $\mathcal{B}$ might be, and explain how you know that the vectors of $\mathcal{B}$ are in W. (See Problem 5.3.3 for some ideas of what $\mathcal{B}$ might be.)

(b) Prove that if $\mathbf{x} \in W$, then $\mathbf{x}$ is a linear combination of $\mathcal{B}$. (See Remark 5.3.9.)

(c) Prove that $\mathcal{B}$ is linearly independent, using the definition of linear independence. (See Remark 5.3.11.)

(d) What is the dimension of W? Explain.

5.3.5. Consider the subset

$$W = \left\{ \begin{bmatrix} x_1 \\ 0 \\ 0 \\ x_4 \\ x_5 \\ 0 \\ 0 \\ 0 \end{bmatrix} \in \mathbf{F}_7^8 \;\middle|\; x_1, x_4, x_5 \in \mathbf{F}_7 \right\}. \tag{5.3.14}$$

of $\mathbf{F}_7^8$. Find a basis $\mathcal{B}$ for W, and prove that $\mathcal{B}$ is a basis for W, using the following steps.

(a) First, guess what $\mathcal{B}$ might be, and explain how you know that the vectors of $\mathcal{B}$ are in W. (See Problem 5.3.3 for some ideas of what $\mathcal{B}$ might be.)

(b) Prove that if $\mathbf{x} \in W$, then $\mathbf{x}$ is a linear combination of $\mathcal{B}$. (See Remark 5.3.9.)

(c) Prove that $\mathcal{B}$ is linearly independent, using the definition of linear independence. (See Remark 5.3.11.)

(d) What is the dimension of W? Explain.

5.3.6. Suppose W is a subspace of F^n, $\{\mathbf{u}, \mathbf{v}, \mathbf{w}\}$ spans W, and $\mathbf{x} \in W$. One of the following statements is always true:

- $\{\mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{x}\}$ spans W.
- $\{\mathbf{u}, \mathbf{v}\}$ spans W.

Choose the true statement and prove it.

5.3.7. Suppose $\{\mathbf{u}, \mathbf{v}, \mathbf{w}\}$ is a linearly independent subset of F^n, and $\mathbf{x} \in W$. One of the following statements is always true:

- $\{\mathbf{u}, \mathbf{v}, \mathbf{w}, \mathbf{x}\}$ is linearly independent.
- $\{\mathbf{u}, \mathbf{v}\}$ is linearly independent.

Choose the true statement and prove it.

5.3.8. In $\mathbf{F}_5^7$, suppose we have a set of vectors

$$B = \left\{ \begin{bmatrix} 1 \\ * \\ * \\ 0 \\ * \\ * \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ * \\ * \\ 0 \\ * \\ * \\ -2 \end{bmatrix}, \begin{bmatrix} 0 \\ * \\ * \\ -1 \\ * \\ * \\ 0 \end{bmatrix} \right\}, \tag{5.3.15}$$

where each $*$ represents an entry with an unspecified value. Use the definition of linear independence to prove that B is linearly independent.

5.3.9. In $\mathbf{F}_{11}^7$, suppose we have a set of vectors

$$B = \left\{ \begin{bmatrix} 3 \\ a \\ b \\ * \\ * \\ * \\ * \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \\ c \\ * \\ * \\ * \\ * \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 5 \\ * \\ * \\ * \\ * \end{bmatrix} \right\}, \tag{5.3.16}$$

where each $*$ represents an entry with an unspecified value, and $a, b, c \in \mathbf{F}_{11}$ are also unspecified. Use the definition of linear independence to prove that B is linearly independent.

5.3.10. Let W be the subspace of $\mathbf{F}_2^7$ defined by

$$W = \operatorname{span} \left\{ \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \right\}. \tag{5.3.17}$$

(a) What is the dimension d of W? What does d tell you about how many vectors there are in W?

(b) Write down every element of W.

5.3.11. Let W be the subspace of $\mathbf{F}_3^7$ defined by

$$W = \operatorname{span} \left\{ \begin{bmatrix} 1 \\ 1 \\ -1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ -1 \\ 0 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \right\}. \tag{5.3.18}$$

(a) How many vectors are there in W? Explain your answer.

(b) What is the dimension d of W? Explain your answer.

5.3.12. Let W be the subspace of $\mathbf{F}_2^9$ defined by

$$W = \operatorname{span}\left\{ \begin{bmatrix}1\\1\\0\\1\\0\\1\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\\0\\1\\0\\1\\1\\1\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\\0\\1\\0\\0\\1\end{bmatrix}, \begin{bmatrix}1\\0\\0\\1\\0\\0\\1\\0\\1\end{bmatrix} \right\}. \tag{5.3.19}$$

(a) How many vectors are there in W? Explain your answer.

(b) What is the dimension d of W? Explain your answer.

5.4 Matrices with entries in a field F

So yes, as promised at the end of Section 5.3, to understand spanning and linear independence and to dispel the nightmares of Ask Yourself 5.3.16, we finally have to do some calculations. And yes, if you've taken linear algebra, this may be the part of it you remember best! But don't get too comfortable; we still have to re-examine all of that material over arbitrary fields.

We start by defining matrices themselves. Throughout this section, let F be a field.

Definition 5.4.1. To say that A is an $n \times k$ *matrix over* F means that A is a box of nk elements of F, arranged as:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nk} \end{bmatrix}. \tag{5.4.1}$$

If we know the size of A, we abbreviate (5.4.1) as $A = [a_{ij}]$, which means that a_{ij} is the element in *row* i and *column* j.

For $n \times k$ matrices A and B over F, and $c \in F$, we define *matrix addition* and *scalar multiplication* by

$$\begin{aligned} A + B &= \begin{bmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nk} \end{bmatrix} + \begin{bmatrix} b_{11} & \cdots & b_{1k} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nk} \end{bmatrix} \\ &= \begin{bmatrix} a_{11}+b_{11} & \cdots & a_{1k}+b_{1k} \\ \vdots & \ddots & \vdots \\ a_{n1}+b_{n1} & \cdots & a_{nk}+b_{nk} \end{bmatrix} \end{aligned} \tag{5.4.2}$$

and

$$cA = c \begin{bmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nk} \end{bmatrix} = \begin{bmatrix} ca_{11} & \cdots & ca_{1k} \\ \vdots & \ddots & \vdots \\ ca_{n1} & \cdots & ca_{nk} \end{bmatrix}. \tag{5.4.3}$$

In abbreviated form, this becomes

$$[a_{ij}] + [b_{ij}] = [a_{ij} + b_{ij}], \tag{5.4.4}$$
$$c[a_{ij}] = [ca_{ij}]. \tag{5.4.5}$$

We also define a *zero matrix* to be a matrix whose entries are all equal to 0.

We denote the set of all $n \times k$ matrices over F by $M_{n\times k}(F)$, and in the most common case where $n = k$, we abbreviate $M_n(F) = M_{n\times n}(F)$. Note that F^n (Definition 5.3.1) is just the $n \times 1$ case of $M_{n\times k}(F)$, and conversely, we can think of $M_{n\times k}(F)$ as F^{nk} written in a different format. In particular, $n \times 1$ matrices are called *column vectors*, and $1 \times k$ matrices are called *row vectors*. If we want to refer to either column or row vectors, but not specify which, we simply talk about *vectors*.

The purpose of writing matrices in a box, as opposed to, say, a very long column, is to define matrix multiplication, which we do in stages as follows.

Definition 5.4.2. If $\mathbf{x} = [x_1 \ \ldots \ x_n]$ is a $1 \times n$ row vector over F and $\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$ is an $n \times 1$ column vector over F, we define the *row-column product*, or *dot product*, to be

$$\mathbf{x} \cdot \mathbf{y} = [x_1 \ \ldots \ x_n] \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = x_1y_1 + \cdots + x_ny_n. \tag{5.4.6}$$

Note that we can identify F with $M_1(F)$, so we can think of the result of the row-column product (5.4.6) as a 1×1 matrix.

Definition 5.4.3. Let $A = [a_{ij}]$ be an $n \times k$ matrix over F, and let $\mathbf{x} = [x_j]$ be a $k \times 1$ column vector over F. Let $\mathbf{r}_i = [a_{i1} \ \ldots \ a_{ik}]$ be the ith row of A. Then we define the *matrix-vector product* $A\mathbf{x}$ to be the $n \times 1$ column vector

$$A\mathbf{x} = \begin{bmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_n \end{bmatrix} \mathbf{x} = \begin{bmatrix} \mathbf{r}_1 \cdot \mathbf{x} \\ \vdots \\ \mathbf{r}_n \cdot \mathbf{x} \end{bmatrix}. \tag{5.4.7}$$

In other words, the ith entry of $A\mathbf{x}$ is the dot product $\mathbf{r}_i \cdot \mathbf{x}$.

Example 5.4.4. Taking $F = \mathbf{R}$, let $\mathbf{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ and $A = \begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \\ 13 & 14 & 15 \end{bmatrix}$. We have that

$$A\mathbf{x} = \begin{bmatrix} 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \\ 13 & 14 & 15 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 1(4)+2(5)+3(6) \\ 1(7)+2(8)+3(9) \\ 1(10)+2(11)+3(12) \\ 1(13)+2(14)+3(15) \end{bmatrix}. \tag{5.4.8}$$

Note that this final product can also be written as

$$\begin{bmatrix} 1(4)+2(5)+3(6) \\ 1(7)+2(8)+3(9) \\ 1(10)+2(11)+3(12) \\ 1(13)+2(14)+3(15) \end{bmatrix} = 1\begin{bmatrix} 4 \\ 7 \\ 10 \\ 13 \end{bmatrix} + 2\begin{bmatrix} 5 \\ 8 \\ 11 \\ 14 \end{bmatrix} + 3\begin{bmatrix} 6 \\ 9 \\ 12 \\ 15 \end{bmatrix}. \tag{5.4.9}$$

Example 5.4.4 leads to a crucial observation.

Remark 5.4.5. For us, perhaps the most important way of looking at matrix-vector multiplication (Definition 5.4.3) is as follows. If A is a $n \times k$ matrix whose k columns are the $n \times 1$ column vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$, and $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix}$ is a $k \times 1$ column vector, then

$$A\mathbf{x} = x_1\mathbf{v}_1 + \cdots + x_k\mathbf{v}_k. \tag{5.4.10}$$

Comparing (5.4.10) and Defintion 5.3.6, we see that:

> $A\mathbf{x}$ is the linear combination of the columns of A with coefficients given by the entries of $\mathbf{x}$.

We are now ready for the general definition of matrix multiplication.

Definition 5.4.6. Let A be an $n \times k$ matrix and let B be a $k \times s$ matrix over F. Let $\mathbf{r}_1, \ldots, r_n$ be the n rows of A, each a $1 \times k$ row vector, and let $\mathbf{b}_1, \ldots, \mathbf{b}_s$ be the s columns of B, each a $k \times 1$ column vector. We define the *matrix product* AB to be the $n \times s$ matrix

$$AB = [A\mathbf{b}_1 \ \ldots \ A\mathbf{b}_s] = \begin{bmatrix} \mathbf{r}_1 \cdot \mathbf{b}_1 & \cdots & \mathbf{r}_1 \cdot \mathbf{b}_s \\ \vdots & \ddots & \vdots \\ \mathbf{r}_n \cdot \mathbf{b}_1 & \cdots & \mathbf{r}_n \cdot \mathbf{b}_s \end{bmatrix}. \tag{5.4.11}$$

Unwrapping Defintitions 5.4.2 and 5.4.3, we see that if $A = [a_{ij}]$ and $B = [b_{j\ell}]$, then $AB = [c_{i\ell}]$ can also be defined by the formula

$$c_{i\ell} = \sum_{j=1}^{k} a_{ij} b_{j\ell}. \tag{5.4.12}$$

While (5.4.11) is probably the best way to remember matrix multiplication in practice, the formula (5.4.12) is useful for proving the following algebraic properties of matrix multiplication.

Theorem 5.4.7. *We have that*

$$\begin{aligned} (AB)C &= A(BC), & a(AB) &= (aA)B = A(aB), \\ A(B+C) &= AB + AC, & (A+B)C &= AC + BC, \end{aligned} \tag{5.4.13}$$

whenever $a \in F$ and A, B, C are matrices of the appropriate size to be multiplied as shown. In other words, matrix multiplication is associative, associates and commutes with scalar multiplcation, and is also distributive on both sides.

Proof. This is an application of (5.4.12) and "death by subscript"; see Problems 5.4.1 and 5.4.2. □

So now that we've finally defined the tool of matrix multiplication, let's go back and see how matrix multiplication connects with the foundational ideas of spanning, linear independence, bases, and dimension. To start this connection, we define yet two more ideas.

Definition 5.4.8. Let A be an $n \times k$ matrix over F. The *column space of* A, or $\operatorname{Col}(A)$, is defined to be the span of the columns of A (a subspace of F^n); and the *nullspace of* A, or $\operatorname{Null}(A)$, is a subset of F^k defined by

$$\operatorname{Null}(A) = \left\{ \mathbf{x} \in F^k \,\middle|\, A\mathbf{x} = \mathbf{0} \right\}. \tag{5.4.14}$$

Given an $n \times k$ matrix A, by Theorem 5.3.8, $\operatorname{Col}(A)$ is a subspace of F^n, since it's the span of a set of vectors. Nullspaces are also subspaces:

Theorem 5.4.9. *Let A be an $n \times k$ matrix over F. Then* $\operatorname{Null}(A)$ *(the nullspace of A) is a subspace of F^k.*

Proof. Problem 5.4.4. □

Also, Remark 5.4.5 now gives us a computationally useful description of linear independence. That is, if $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ are the columns of a matrix A:

$$\begin{aligned} & \text{The set } \{\mathbf{v}_1, \dots, \mathbf{v}_k\} \text{ is linearly independent.} \\ \Leftrightarrow\ & \text{The only linear combination of } \{\mathbf{v}_1, \dots, \mathbf{v}_k\} \text{ equal to } \mathbf{0} \\ & \text{is when all coefficients are equal to } 0. \\ \Leftrightarrow\ & \text{The only vector } \mathbf{x} \text{ such that } A\mathbf{x} = \mathbf{0} \text{ is } \mathbf{x} = \mathbf{0}. \\ \Leftrightarrow\ & \operatorname{Null}(A) = \{\mathbf{0}\}. \end{aligned}$$

In other words:

Theorem 5.4.10. *Let $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ be a subset of F^n, and let A be the $n \times k$ matrix whose columns are $\mathbf{v}_1, \dots, \mathbf{v}_k$. Then $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is linearly independent if and only if* $\mathrm{Null}(A) = \{\mathbf{0}\}$, *the zero subspace.* $\square$

The upshot is that we can now restate our main computational problems (Motivating Problems 5.3.17–5.8.1) in matrix form, as follows. We begin with the computational equivalent of Motivating Problem 5.3.17.

Motivating Problem 5.4.11. Given an $n \times k$ matrix A over F, determine if $\mathrm{Null}(A) = \{\mathbf{0}\}$, or in other words, determine if the columns $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ of A are linearly independent. Note that if $W = \mathrm{span}\,\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$, this determines if $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is a basis for W.

Motivating Problem 5.3.18 has two different matrix forms, depending on how W is initially given to us.

Motivating Problem 5.4.12. Given an $n \times k$ matrix A over F, find a basis for the column space $\mathrm{Col}(A)$ (Definition 5.4.8), the span of the columns $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ of A. Note that this solves the problem of finding a basis for the subspace spanned by a given set $\{\mathbf{v}_1, \dots, \mathbf{v}_k\} \subseteq F^n$.

Motivating Problem 5.4.13. Given an $n \times k$ matrix A over F, find a basis for $\mathrm{Null}(A)$ (Definition 5.4.8).

Remarkably, as we'll see in the next section, Motivating Problems 5.4.11–5.4.13 all have the same solution: *Gaussian reduction.*

Problems

5.4.1. (*Proves Theorem 5.4.7*) proof that matrix multiplication is associative and associates and commutes with scalar multiplication.

5.4.2. (*Proves Theorem 5.4.7*) In this problem, assume all matrices are over F.

(a) Suppose $A = [a_{ij}]$ is an $n \times k$ matrix and $B = [b_{j\ell}]$ and $C = [c_{j\ell}]$ are $k \times s$ matrices. Prove that $A(B + C) = AB + AC$. (Suggestion: Use (5.4.4) and (5.4.12) to find a formula for the i, ℓ-entry of each side of $A(B + C) = AB + AC$. Alternate problem: Do only the case $n = 3$, $k = 2$, $s = 4$.)

(b) Suppose $A = [a_{ij}]$ and $B = [b_{ij}]$ are $n \times k$ matrices and $C = [c_{j\ell}]$ is a $k \times s$ matrix. Prove that $(A + B)C = AC + BC$. (Suggestion: Similar to the other part of this problem. Alternate problem: Do only the case $n = 2$, $k = 4$, $s = 3$.)

5.4.3. $M_n(F)$ is a noncommutative ring.

5.4.4. (*Proves Theorem 5.4.9*) Let A be a $k \times n$ matrix with entries in F, and again, let $\mathrm{Null}(A)$ be the set of all $\mathbf{x} \in F^n$ such that $A\mathbf{x} = \mathbf{0}$.

(a) Explain why $\mathbf{0}$ is in $\mathrm{Null}(A)$.

(b) Suppose $\mathbf{x}$ and $\mathbf{y}$ are in Null(A). Explain why $\mathbf{x}+\mathbf{y}$ must be in Null(A). (Suggestion: What does it mean to say that $\mathbf{x}$ is in Null(A)? What does it mean to say that $\mathbf{x}+\mathbf{y}$ is in Null(A)? You may also want to review Section 1.3.5.)

(c) Suppose $\mathbf{x}$ is in Null(A) and $a \in F$. Explain why $a\mathbf{x}$ must be in Null(A).

5.5 Systems of linear equations (homogeneous case)

Again, throughout this section, let F be a field.

Definition 5.5.1. A *linear equation* is an equation of the form

$$a_1x_1 + \cdots + a_kx_k = b, \tag{5.5.1}$$

where $a_i, b \in F$ are constants and the x_i are the unknowns. A *system of n linear equations in k variables* has the form

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1k}x_k &= b_1, \\ &\vdots \\ a_{n1}x_1 + \cdots + a_{nk}x_k &= b_n. \end{aligned} \tag{5.5.2}$$

Note that if

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nk} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}, \tag{5.5.3}$$

then (5.5.2) can be rewritten as $A\mathbf{x} = \mathbf{b}$. In this section, and in the great majority of this book, we are only concerened with the case $\mathbf{b} = \mathbf{0}$, in which we say that the system (5.5.2) is *homogeneous*; in other words, a homogeneous system of linear equations is precisely one of the form $A\mathbf{x} = \mathbf{0}$. We also define the *matrix of the homogeneous linear system $A\mathbf{x} = \mathbf{0}$* to be A itself.

As you may recall if you've taken linear algebra, if a linear system is represented by a matrix of the following type, then it is particularly easy to solve.

Definition 5.5.2. Let A be a matrix over F. To say that A is in *row-echelon form*, or *REF*, means that:

1. The leftmost entry of each nonzero row of A is 1. (These are called *leading* 1*s*.)

2. The leading 1s move strictly to the right as we go down the rows of A. For example, if the leading 1 is the 3rd entry of row i, then if row $i+1$ is nonzero, its leading 1 must be entry j for some $j \geq 4$.

3. Any all-0 rows of A must be in its final row(s).

If A is in REF, we call the columns containing leading 1s the *pivot columns* of A. Note that by property 2, if A is in REF, all entries underneath a leading 1 must be 0. If A is in REF, and in addition, all entries *above* every leading 1 are 0, we say that A is in *reduced row-echelon form*, or *RREF*.

As promised, we now describe how to write down the solution to a homogeneous system of linear equations whose matrix is in RREF.

Algorithm 5.5.3. Suppose we have a homogeneous system of linear equations whose matrix A is in RREF, and suppose A is an $n \times k$ matrix.

1. Note that the jth column of A corresponds to the variable x_j. Call the variables corresponding to pivot columns of A the *pivot variables*, and call the other variables of the system the *free variables*.
2. Let $\{x_{j_1}, \dots, x_{j_f}\}$ be the f free variables. Add an equation $x_{j_m} = x_{j_m}$ for each free variable x_{j_m}; after eliminating the redundant $0 = 0$ equations, we now have n equations in n variables.
3. Rewrite each of the n equations as $x_i =$ (everything else in the equation moved to the right-hand side). We then see that our solution set is the set of all $\mathbf{x}$ of the form
$$\mathbf{x} = x_{j_1}\alpha'_1 + \cdots + x_{j_f}\alpha'_f, \tag{5.5.4}$$
where each α'_m is the negative of column j_m of A, with a 1 inserted at location j_m and a 0 inserted at every other location in $j_1, \dots, j_f$.

I know, that description is so awkward that it's probably only useful if you already know what it says! However, some examples should make things clearer.

Example 5.5.4. Consider the system with matrix
$$A = \begin{bmatrix} 1 & 2 & 0 & 0 & -3 \\ 0 & 0 & 1 & 0 & 5 \\ 0 & 0 & 0 & 1 & 7 \end{bmatrix} \tag{5.5.5}$$

Our free variables are x_2 and x_5, so adding the corresponding rows $x_2 = x_2$ and $x_5 = x_5$, we get
$$\begin{array}{ccccccccccc} x_1 & + & 2x_2 & & & & - & 3x_5 & = & 0, & \\ & & x_2 & & & & & & = & & x_2, \\ & & & & x_3 & & + & 5x_5 & = & 0, & \\ & & & & & x_4 & + & 7x_5 & = & 0, & \\ & & & & & & & x_5 & = & & x_5. \end{array} \tag{5.5.6}$$

Rearranging to put each equation in the form $x_i =$ (stuff), we get
$$\begin{array}{cccccc} x_1 & = & - & 2x_2 & + & 3x_5, \\ x_2 & = & & x_2, & & \\ x_3 & = & & & - & 5x_5, \\ x_4 & = & & & - & 7x_5, \\ x_5 & = & & & & x_5. \end{array} \tag{5.5.7}$$

Note all of the sign changes in the equations other than $x_2 = x_2$ and $x_5 = x_5$. Finally, rewriting in column vector form, we get

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = x_2 \begin{bmatrix} -2 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + x_5 \begin{bmatrix} 3 \\ 0 \\ -5 \\ -7 \\ 1 \end{bmatrix}. \tag{5.5.8}$$

Note where the 0s and 1s have been inserted, and again note the annoying sign changes we have to remember (because life is cruel). Also, we haven't specified the field F over which this is all happening, but it doesn't matter — the same calculation works no matter what F is.

The main feature of Algorithm 5.5.3 is that it gives a basis for $\mathrm{Null}(A)$, solving Motivating Problems 5.4.11 and 5.4.13 in the case where A is in RREF. More precisely:

Theorem 5.5.5. *In the notation of Algorithm 5.5.3, $\left\{\alpha'_1, \dots, \alpha'_f\right\}$ is a basis for $\mathrm{Null}(A)$, with exactly one basis vector for each free variable. In particular, if every column of A is a pivot column, then $\mathrm{Null}(A) = \mathbf{0}$.*

Proof. The notation for this proof in general is both icky and also hides the content of what's going on, so the best thing for you to do is to explain what happens in a concrete specific case; see Problem 5.5.1. □

So how can we put a given system of linear equations in RREF? To start, the following three operations on a system of linear equations, or equivalently, on its matrix A, do not change its solution set.

Definition 5.5.6. The *elementary operations* on a system of linear equations, or matrix, are:

1. Switch equation i and equation j, or equivalently, switch row i and row j of A.
2. For $a \in F$, $a \neq 0$, multiply both sides of equation i by a, or equivalently, multiply row i of A by a.
3. For $a \in F$, add a times equation i to equation j, or equivalently, add a times row i of A to row j.

Note that the proof that elementary operations do not change the $\mathrm{Null}(A)$ is just that all three operations are reversible, so in each case, the "old" system of equations implies the "new" one, and vice versa. Crucially, though, we need to use the fact that F is a field to reverse multiplication of row i by $a \neq 0$.

Gauss devised the following algorithm to combine the elementary operations to put any matrix in RREF.

Algorithm 5.5.7. *Gaussian reduction* is the following (recursive) algorithm for putting a $n \times k$ matrix A into REF.

1. If A is the $n \times k$ zero matrix, done.
2. Otherwise, swap the rows of A (elementary operation type 1) to get a leftmost nonzero entry $a \neq 0$ in the top row of A.
3. Multiply the top row of A by a^{-1} to make the leftmost nonzero entry of the top row equal to 1. (Here, we again rely on the assumption that F is a field.)
4. Add appropriate multiples of the top row of A to the other rows of A to make all entries underneath the leading 1 of the top row equal to 0.
5. Go back to step 1 and apply Gaussian reduction to the $k-1$ rows of A beneath the top row.

We can then put A into RREF by adding appropriate multiples of each nonzero row to the rows above it, to make all entries above each leading 1 also equal to 0. We call this final result RREF(A), the *RREF of* A.

Remark 5.5.8. It turns out to be true, though certainly it's far from obvious, that RREF(A) is unique, justifying our use of "the" in "the RREF of A." See [**?**] for a proof of uniqueness in the case where $F = \mathbf{R}$, which you can turn into a proof in the general case by replacing $\mathbf{R}$ with F throughout.

Again, an example will probably be clearer than the formal description of the algorithm. As with long division (see Remark 3.4.6), we note that the case in which we are most interested is when $F = \mathbf{F}_2$, but we also include examples and exercises with $F = \mathbf{F}_3$ or $\mathbf{F}_5$, so as not to obscure the roles of $\pm$ signs and division.

Example 5.5.9. Let $F = \mathbf{F}_5$, and keep in mind that mod 5, $3 = -2$, $4 = -1$, and $2 \cdot 3 = 1$. To put $A = \begin{bmatrix} 3 & 1 & 1 & 2 & 3 & 1 \\ 3 & 3 & 3 & 2 & 0 & 3 \\ 4 & 4 & 2 & 1 & 1 & 3 \end{bmatrix}$ in RREF, we do the following. First, multiply the new top row by $3^{-1} = 2$:

$$\to \begin{bmatrix} 1 & 2 & 2 & 4 & 1 & 2 \\ 3 & 3 & 3 & 2 & 0 & 3 \\ 4 & 4 & 2 & 1 & 1 & 3 \end{bmatrix} \tag{5.5.9}$$

Then add 2 times row 1 to row 2:

$$\to \begin{bmatrix} 1 & 2 & 2 & 4 & 1 & 2 \\ 0 & 2 & 2 & 0 & 2 & 2 \\ 4 & 4 & 2 & 1 & 1 & 3 \end{bmatrix} \tag{5.5.10}$$

And then add row 1 to row 3, to get:

$$\to \begin{bmatrix} 1 & 2 & 2 & 4 & 1 & 2 \\ 0 & 2 & 2 & 0 & 2 & 2 \\ 0 & 1 & 4 & 0 & 2 & 0 \end{bmatrix}. \tag{5.5.11}$$

Next, multiply row 2 by $2^{-1} = 3$:

$$\to \begin{bmatrix} 1 & 2 & 2 & 4 & 1 & 2 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 4 & 0 & 2 & 0 \end{bmatrix} \tag{5.5.12}$$

Subtract row 2 from row 3:

$$\to \begin{bmatrix} 1 & 2 & 2 & 4 & 1 & 2 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 3 & 0 & 1 & 4 \end{bmatrix} \tag{5.5.13}$$

And then multiply row 3 by $3^{-1} = 2$ to get:

$$\to \begin{bmatrix} 1 & 2 & 2 & 4 & 1 & 2 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 2 & 3 \end{bmatrix}. \tag{5.5.14}$$

This matrix is now in REF, so to put it into RREF, we need to clear the entries above the leading 1s. We do that by going from right to left and adding/subtracting multiples of the last row to the ones above it, and then the next-to-last row, and so on. To be precise, subtract row 3 from row 2 and 2 times row 3 from row 1 to get:

$$\to \begin{bmatrix} 1 & 2 & 0 & 4 & 2 & 1 \\ 0 & 1 & 0 & 0 & 4 & 3 \\ 0 & 0 & 1 & 0 & 2 & 3 \end{bmatrix} \tag{5.5.15}$$

And subtract 2 times row 2 from row 1 to get:

$$\to \begin{bmatrix} 1 & 0 & 0 & 4 & 4 & 0 \\ 0 & 1 & 0 & 0 & 4 & 3 \\ 0 & 0 & 1 & 0 & 2 & 3 \end{bmatrix}. \tag{5.5.16}$$

This is the RREF of our original matrix A.

Combining Algorithms 5.5.7, 5.5.3, and 5.8.4, we see that we can solve any system of linear equations over an arbitrary field F, just like we can when $F = \mathbf{R}$; see the problems for examples. Also, combining Algorithm 5.5.7 and Theorem 5.5.5, we have the following solution to Motivating Problem 5.4.11.

Corollary 5.5.10. *Let A be an $n \times k$ matrix over F. The columns of A are linearly independent if and only if all of the columns of the RREF of A are pivot columns (i.e., every column contains a leading 1).* □

Remark 5.5.11. If you still feel uncomfortable with things like fractions in $\mathbf{F}_p$, you might be tempted to do all of your row-reduction over $F = \mathbf{Q}$, and then reduce mod p at the end.

The problem with the approach is that it can produce the wrong answer! To give a toy example that explains the basic issue, consider $A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$. If we reduce A over $\mathbf{F}_3$, we get

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \to \begin{bmatrix} 1 & 2 \\ 0 & -3 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 0 & 0 \end{bmatrix}. \tag{5.5.17}$$

It follows that $\mathrm{Null}(A)$ has dimension 1. However, if we reduce A over $F = \mathbf{F}_p$ for any other prime value of p, or over $F = \mathbf{Q}$, we get an RREF of $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and a nullspace of dimension 0! Again, the moral is that when you reduce over $\mathbf{F}_p$, make sure you reduce as you go along, or you could get the wrong answer.

Looking back at Motivating Problems 5.4.11–5.4.13, we see that:

- Motivating Problem 5.4.11 (determining if a given set is linearly independent) is solved by Corollary 5.5.10.
- Motivating Problem 5.4.13 (finding a basis for $\mathrm{Null}(A)$) is solved as described in Theorem 5.5.5.

So the only one left to solve is Motivating Problem 5.4.12, which we address as follows.

Theorem 5.5.12 (Contraction)**.** *Let A be an $n \times k$ matrix over F. Then the pivot columns of the* original *matrix A, or in other words, the columns of A that correspond to pivot columns of* $\mathrm{RREF}(A)$*, form a basis for* $\mathrm{Col}(A)$*.*

Proof. Recall from Algorithm 5.5.3 (or really, Example 5.5.4) that every column of A either corresponds to a pivot variable or to a free variable in $\mathrm{RREF}(A)$. Call the latter type of column a *free column* of A. Then by choosing one of the free variables to be equal to 1 and all the others to be 0, we get a linear dependency (Definition 5.3.10) in the columns of A that expresses the corresponding free column of A as a linear combination of the pivot columns of A. It follows that the free columns of A don't add anything to the span of the pivot columns, which means that we can remove them with changing $\mathrm{Col}(A)$; in other words, the pivot columns of A span $\mathrm{Col}(A)$.

On the other hand, a linear combination of the pivot columns of A that is equal to $\mathbf{0}$ is equal to a linear combination of *all* of the columns of A with the free variables all set to 0. However, by Algorithm 5.5.3, if all of the free variables are equal to 0 in $A\mathbf{x} = \mathbf{0}$, then $\mathbf{x} = \mathbf{0}$, or in other words, all of the coefficients of the corresponding linear combination of the columns of A are equal to 0. It follows that the pivot columns of A are linearly independent, and so they form a basis for $\mathrm{Col}(A)$. □

Example 5.5.13. Consider the subspace $\mathcal{C}$ of $\mathbf{F}_2^5$ described in Section 5.1. We can now solve Motivating Problems 5.1.3 and 5.1.4 from that discussion, assuming some theory we'll prove in Section 5.6.

First, we observe that Theorem 5.3.15 implies:

A d-dimensional subspace of $\mathbf{F}_p^n$ contains exactly p^d vectors.

Therefore, since $\mathcal{C}$ has $8 = 2^3$ vectors in it, to find a basis for $\mathcal{C}$, we should look for a set $\mathcal{B}$ of 3 linearly independent vectors in $\mathcal{C}$. Then, once we find such a $\mathcal{B}$, Theorem 5.6.6 (which, again, we haven't proven yet) will show that $\mathcal{B}$ is a basis for $\mathcal{C}$.

We might, for example, try starting with the vectors $b = 00111$, $c = 01011$, $d = 01100$, and $e = 10010$, which are the columns of

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}. \tag{5.5.18}$$

Row reduction (details omitted) then shows that $\mathrm{RREF}(A) = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$. So, since the pivot columns of A are 1, 2, and 4, $\{b, c, e\}$ is a basis for $\mathcal{C}$.

The free/pivot setup of the proof of the Contraction Theorem 5.5.12 also gives a fact about the dimensions of $\mathrm{Col}(A)$ and $\mathrm{Null}(A)$ that can be rephrased as a result (Corollary 5.5.15) whose statement is made more succinct by the following definitions.

Definition 5.5.14. Let A be a $n \times k$ matrix over a field F. We define $\mathrm{rank}(A)$, the *rank of* A, to be $\dim(\mathrm{Col}(A))$, the dimension of the column space of A, and we define $\mathrm{nullity}(A)$, the *nullity of* A, to be $\dim(\mathrm{Null}(A))$, the dimension of the nullspace of A.

Since each of the n columns of a $n \times k$ matrix in RREF is either a pivot column or a free column, Theorems 5.5.5 and 5.5.12 together imply the following result.

Corollary 5.5.15 (Rank-Nullity Theorem)**.** *Let A be a $n \times k$ matrix over a field F. Then* $\mathrm{rank}(A) + \mathrm{nullity}(A) = k$. □

Example 5.5.16. Let

$$A = \begin{bmatrix} 1 & 2 & 3 & 6 & 0 \\ 0 & 0 & 1 & 3 & 2 \\ 4 & 1 & 2 & 1 & 1 \end{bmatrix}, \tag{5.5.19}$$

considered as a matrix over $\mathbf{F}_7$. A calculation (which you should do yourself) shows that $\mathrm{RREF}(A)$ is

$$\mathrm{RREF}(A) = \begin{bmatrix} 1 & 2 & 0 & -3 & 1 \\ 0 & 0 & 1 & 3 & 2 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{5.5.20}$$

To compute Null(A), we see that the free variables are x_2, x_4, and x_5, and our usual procedure gives us a basis $\left\{ \begin{bmatrix} -2 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 \\ 0 \\ -3 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \\ -2 \\ 0 \\ 1 \end{bmatrix} \right\}$ for Null(A); and on the other hand, the Contraction Theorem 5.5.12 implies that $\left\{ \begin{bmatrix} 1 \\ 0 \\ 4 \end{bmatrix}, \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} \right\}$ (read directly off the original matrix A) is a basis for Col(A). We see that

$$\operatorname{rank}(A) + \operatorname{nullity}(A) = 2 + 3 = 5, \tag{5.5.21}$$

confirming the Rank-Nullity Theorem 5.5.15.

Of course, all of this computation is only meaningful if dimension is meaningful — if a subspace can have both dimension 7 and dimension 54, then it's not really worth computing the dimension of a subspace! We'll finally resolve that question in the next section. For now, to finish this section, in Table 5.5.1, we summarize the solutions to the most common computational linear algebra problems we will face.

Problem to solve	Algorithm
Are columns of A linearly independent?	If all columns of RREF(A) are pivot, yes; otherwise, no.
Find basis for Col(A)	Use the columns of A corresponding to pivot columns of RREF(A).
Find basis for Null(A)	Solve $A\mathbf{x} = \mathbf{0}$ using Gaussian reduction and Algorithm 5.5.3.

Table 5.5.1: Solutions to motivating problems in linear algebra

Problems

5.5.1. (*Proves Theorem 5.5.5*) Let

$$A = \begin{bmatrix} 1 & 2 & 0 & 0 & -3 & 4 \\ 0 & 0 & 1 & 0 & 5 & -6 \\ 0 & 0 & 0 & 1 & 7 & 8 \end{bmatrix}, \tag{5.5.22}$$

where the entries of A are in $\mathbf{F}_{13}$.

(a) Write out the equations for $A\mathbf{x} = \mathbf{0}$.

(b) Find the set of vectors $\mathcal{B}$ that Algorithm 5.5.3 gives us as a candidate for a basis for Null(A). (Don't assume that $\mathcal{B}$ is a basis for Null(A), as the goal of the rest of this problem is to prove that $\mathcal{B}$ is a basis for Null(A).)

(c) Explain why (prove that) every $\mathbf{x} \in \text{Null}(A)$ is a linear combination of the vectors of $\mathcal{B}$. (See Remark 5.3.9.)

(d) Explain why (prove that) the vectors of $\mathcal{B}$ are linearly independent. (See Remark 5.3.11.)

5.5.2. Find the RREFs of the following matrices with entries in $\mathbf{F}_5$.

(a) $\begin{bmatrix} 0 & 4 & 1 & 2 & 1 & 4 \\ 4 & 1 & 4 & 1 & 3 & 2 \\ 0 & 4 & 3 & 1 & 2 & 4 \\ 4 & 2 & 1 & 0 & 0 & 2 \end{bmatrix}$

(b) $\begin{bmatrix} 4 & 2 & 3 & 2 & 4 & 3 \\ 3 & 3 & 4 & 1 & 0 & 0 \\ 0 & 1 & 4 & 4 & 2 & 2 \\ 4 & 3 & 0 & 0 & 3 & 0 \\ 3 & 0 & 4 & 0 & 3 & 4 \end{bmatrix}$

(c) $\begin{bmatrix} 3 & 3 & 0 & 0 & 4 & 3 & 1 \\ 3 & 2 & 1 & 0 & 0 & 3 & 2 \\ 0 & 2 & 3 & 3 & 2 & 0 & 4 \\ 1 & 4 & 2 & 4 & 4 & 4 & 1 \\ 3 & 0 & 3 & 2 & 2 & 4 & 1 \end{bmatrix}$

(d) $\begin{bmatrix} 1 & 1 & 3 & 2 & 1 & 1 & 0 & 0 \\ 1 & 4 & 3 & 3 & 4 & 4 & 2 & 2 \\ 4 & 4 & 2 & 3 & 3 & 4 & 1 & 1 \\ 4 & 4 & 4 & 2 & 0 & 3 & 0 & 0 \\ 2 & 3 & 3 & 0 & 3 & 2 & 3 & 1 \\ 4 & 4 & 3 & 0 & 0 & 1 & 2 & 2 \end{bmatrix}$

5.5.3. Find the RREFs of the following matrices with entries in $\mathbf{F}_3$.

(a) $\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 2 & 2 \\ 1 & 1 & 2 & 2 & 0 & 2 \\ 2 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$

(b) $\begin{bmatrix} 1 & 0 & 0 & 2 & 0 & 2 \\ 2 & 1 & 2 & 1 & 0 & 0 \\ 2 & 1 & 2 & 2 & 2 & 2 \\ 2 & 0 & 0 & 2 & 1 & 2 \\ 2 & 2 & 1 & 1 & 0 & 1 \end{bmatrix}$

(c) $\begin{bmatrix} 0 & 2 & 0 & 2 & 1 & 0 & 1 \\ 2 & 1 & 0 & 2 & 2 & 2 & 0 \\ 1 & 2 & 2 & 2 & 2 & 0 & 0 \\ 1 & 0 & 0 & 2 & 1 & 2 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$

(d) $\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 2 & 1 & 2 \\ 2 & 2 & 1 & 2 & 0 & 1 & 2 & 2 \\ 2 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 1 & 1 & 2 & 2 \\ 0 & 1 & 2 & 2 & 2 & 2 & 0 & 0 \\ 0 & 1 & 2 & 2 & 0 & 2 & 1 & 0 \end{bmatrix}$

5.5.4. Find the RREFs of the following matrices with entries in $\mathbf{F}_2$.

$$\text{(a)} \begin{bmatrix} 1&1&0&1&0&1&0\\ 1&1&1&0&1&0&0\\ 1&1&1&1&0&0&1\\ 0&0&0&1&1&0&0\\ 1&1&1&0&1&1&0 \end{bmatrix} \qquad \text{(b)} \begin{bmatrix} 0&0&0&0&1&1&0&1\\ 0&0&1&1&1&0&1&0\\ 1&1&1&1&0&0&0&0\\ 1&1&1&1&1&0&1&1\\ 1&1&0&0&1&0&0&1 \end{bmatrix}$$

$$\text{(c)} \begin{bmatrix} 1&0&0&1&0&0&1&0\\ 0&1&0&0&0&1&1&0\\ 1&1&0&1&1&1&1&1\\ 0&0&0&0&0&1&0&0\\ 1&1&1&1&0&1&0&0\\ 1&1&1&1&0&0&1&1 \end{bmatrix} \qquad \text{(d)} \begin{bmatrix} 0&1&1&0&0&1&0&0\\ 0&1&0&0&1&0&1&0\\ 1&0&1&1&1&0&0&0\\ 1&1&1&1&0&0&0&0\\ 0&1&0&0&1&0&1&0\\ 0&1&1&0&0&0&0&1\\ 0&0&0&1&1&0&1&0 \end{bmatrix}$$

5.5.5. For each of the following matrices with entries in $\mathbf{F}_5$, find bases for $\mathrm{Col}(A)$ and $\mathrm{Null}(A)$.

$$\text{(a) } A = \begin{bmatrix} 4&4&1&2&0\\ 2&2&4&0&4\\ 4&4&2&2&4 \end{bmatrix} \qquad \text{(b) } A = \begin{bmatrix} 1&4&2&0&0\\ 2&3&4&0&4\\ 4&1&3&0&3\\ 2&3&3&1&1 \end{bmatrix}$$

$$\text{(c) } A = \begin{bmatrix} 2&4&3&1&1&4\\ 4&1&0&4&0&2\\ 0&3&1&0&3&0\\ 1&1&4&0&0&2 \end{bmatrix} \qquad \text{(d) } A = \begin{bmatrix} 4&3&1&4&2&3\\ 0&0&4&4&0&1\\ 4&3&0&0&4&3\\ 3&1&4&1&2&3 \end{bmatrix}$$

5.5.6. For each of the following matrices A with entries in $\mathbf{F}_3$, find bases for $\mathrm{Col}(A)$ and $\mathrm{Null}(A)$.

$$\text{(a) } A = \begin{bmatrix} 2&1&2&1&1\\ 2&0&1&0&2\\ 2&2&2&2&2 \end{bmatrix} \qquad \text{(b) } A = \begin{bmatrix} 2&0&1&0&1\\ 0&1&2&0&2\\ 1&2&0&0&0\\ 0&2&1&1&0 \end{bmatrix}$$

$$\text{(c) } A = \begin{bmatrix} 1&0&2&0&2&2\\ 1&0&2&2&1&0\\ 0&1&1&1&1&1\\ 0&2&2&0&1&0 \end{bmatrix} \qquad \text{(d) } A = \begin{bmatrix} 2&0&1&0&0&0\\ 1&1&1&1&1&2\\ 1&2&1&1&1&1\\ 0&2&1&2&1&2\\ 2&0&1&0&0&0 \end{bmatrix}$$

5.5.7. For each of the following matrices A with entries in $\mathbf{F}_2$, find bases for $\mathrm{Col}(A)$ and

Null(A).

(a) $A = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$ (b) $A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$

(c) $A = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$ (d) $A = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$

5.6 Dimension and rank-nullity

To review our story so far, our remaining task in linear algebra is to dispel the three nightmares of Ask Yourself 5.3.16. We start with an innocent-looking observation that turns out to be exactly what we need to resolve our nagging theoretical worries.

Lemma 5.6.1. *Let A be a $n \times k$ matrix with $n < k$. Then $A\mathbf{x} = \mathbf{0}$ has at least one nonzero solution $\mathbf{x}$.*

Proof. Since $n < k$, RREF(A) has at least one free variable, which means that we can set that variable to 1, obtaining a nonzero solution. □

The key to understanding dimension is the following theorem.[†]

Theorem 5.6.2 (Comparison Theorem)**.** *Let W be a subspace of F^n. If $\{\mathbf{v}_1, \dots, \mathbf{v}_s\}$ spans W and $\{\mathbf{w}_1, \dots, \mathbf{w}_\ell\}$ is a linearly independent subset of W, then $\ell \leq s$.*

The Comparison Theorem is so important we'll prove it twice, first in the special case where F is a field containing finitely many elements, like $F = \mathbf{F}_p$.

Proof of finite field case of Comparison Theorem. Suppose F contains q elements. Since $\{\mathbf{w}_1, \dots, \mathbf{w}_\ell\}$ is a linearly independent subset of W, by Problem 5.6.1, W must contain at least q^ℓ distinct vectors. On the other hand, since $\{\mathbf{v}_1, \dots, \mathbf{v}_s\}$ spans W, every vector $\mathbf{w} \in W$ has the form

$$\mathbf{w} = a_1\mathbf{v}_1 + \cdots + a_s\mathbf{v}_s \tag{5.6.1}$$

for some choice of coefficients $a_i \in F$. Since there are q possible choices for each of s coefficients, there are at most q^s possible vectors of the form (5.6.1), which means that W contains at most q^s distinct vectors. So if $|W|$ is the number of elements of W, then

$$q^\ell \leq |W| \leq q^s, \tag{5.6.2}$$

which means that $\ell \leq s$. □

[†]To give credit where credit is due, I learned the name "Comparison Theorem," as well as all of the other theorem names in this section, from Messer [?], my favorite linear algebra textbook.

Proof of Comparison Theorem in general. Suppose $\{\mathbf{v}_1, \dots, \mathbf{v}_s\}$ spans W, $\{\mathbf{w}_1, \dots, \mathbf{w}_k\}$ is a subset of W, and $k > s$. Our goal is to show that $\{\mathbf{w}_1, \dots, \mathbf{w}_k\}$ must be linearly dependent.

Let A be the $n \times s$ matrix whose columns are $\mathbf{v}_1, \dots, \mathbf{v}_s$, and let B the $n \times k$ matrix whose columns are $\mathbf{w}_1, \dots, \mathbf{w}_k$. Since $\{\mathbf{v}_1, \dots, \mathbf{v}_s\}$ spans W, every $\mathbf{w} \in W$ is a linear combination of $\mathbf{v}_1, \dots, \mathbf{v}_s$. It follows from Remark 5.4.5 that for $1 \leq i \leq k$, there exists some $\mathbf{c}_i \in F^s$ such that $A\mathbf{c}_i = \mathbf{w}_i$. Therefore, if we let C be the $s \times k$ matrix whose columns are $\mathbf{c}_1, \dots, \mathbf{c}_k$, we have that

$$AC = A[\mathbf{c}_1 \ \cdots \ \mathbf{c}_k] = [A\mathbf{c}_1 \ \cdots \ A\mathbf{c}_k] = [\mathbf{w}_1 \ \cdots \ \mathbf{w}_k] = B. \tag{5.6.3}$$

However, since $s < k$, by Lemma 5.6.1, there exists some $\mathbf{x} \in F^k$ such that $C\mathbf{x} = \mathbf{0}$ and $\mathbf{x} \neq \mathbf{0}$. Furthermore,

$$B\mathbf{x} = AC\mathbf{x} = A\mathbf{0} = \mathbf{0}, \tag{5.6.4}$$

which means, by Theorem 5.4.10, that $\{\mathbf{w}_1, \dots, \mathbf{w}_k\}$ is linearly dependent. □

With Theorem 5.6.2 in hand, getting rid of our first nightmare (two different dimensions for the same subspace) is a matter of applying the definitions from Section 5.3 and the Comparison Theorem 5.6.2.

Corollary 5.6.3 (Dimension Theorem)**.** *Any two bases for W must have the same size k (i.e., W cannot have more than one dimension).*

Proof. Problem 5.6.2. □

Corollary 5.6.4. *If $\dim W = k$, any linearly independent set must have size $\leq k$ and any span set must have size $\geq k$.*

Proof. Problem 5.6.3. □

It remains to dispel the other nightmares of Ask Yourself 5.3.16: the possibility that a subspace of W might have a larger dimension than W and the possibility that some subspace of F^n might not have a basis at all. The following idea turns out to be enough to slay those monsters.

Definition 5.6.5. Let W be a subspace of F^n. A *maximal linearly independent subset of W* is a linearly independent subset $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ of W such that for any $\mathbf{x} \in W$, $\{\mathbf{v}_1, \dots, \mathbf{v}_k, \mathbf{x}\}$ is linearly dependent (not linearly independent).

Theorem 5.6.6. *Let W be a subspace of F^n, and suppose $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is a maxmimal linearly independent subset of W. Then $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is a basis for W.*

Proof. Since $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is linearly independent, by the definition of basis, it suffices to show that $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ spans W; that is, we want to show that for any $\mathbf{x} \in W$, $\mathbf{x}$ is a linear combination of $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$.

So suppose $\mathbf{x}$ is a vector in W. Since $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is a *maximal* linearly independent subset of W $\{\mathbf{v}_1, \dots, \mathbf{v}_k, \mathbf{x}\}$ is linearly dependent. In other words, there must exist coefficients $a_1, \dots, a_k, b \in F$, not all equal to 0, such that

$$a_1\mathbf{v}_1 + \cdots + a_k\mathbf{v}_k + b\mathbf{x} = \mathbf{0}. \tag{5.6.5}$$

Now, if $b = 0$, then

$$a_1\mathbf{v}_1 + \cdots + a_k\mathbf{v}_k = \mathbf{0} \tag{5.6.6}$$

with not all of the a_i equal to 0, contradicting our assumption that $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is linearly independent. It follows that $b \neq 0$, and therefore,

$$\mathbf{x} = -\frac{a_1}{b}\mathbf{v}_1 - \cdots - \frac{a_k}{b}\mathbf{v}_k. \tag{5.6.7}$$

The theorem follows. □

We can now start to get rid of the remaining nightmares.

Corollary 5.6.7. *If W is a subspace of F^n, then W has a basis.*

Proof. We can construct a basis B for W, at least in the abstract, as follows. Start with $B = \emptyset$.

1. If we can find some $\mathbf{v}_1 \in W$ such that $B \cup \{\mathbf{v}_1\} = \{\mathbf{v}_1\}$ is linearly independent, then let $B = \{\mathbf{v}_1\}$ and go on to step 2. Otherwise, B is maximal linearly independent.

2. If we can find some $\mathbf{v}_2 \in W$ such that $B \cup \{\mathbf{v}_2\} = \{\mathbf{v}_1, \mathbf{v}_2\}$ is linearly independent, then let $B = \{\mathbf{v}_1, \mathbf{v}_2\}$ and go on to step 3. Otherwise, B is maximal linearly independent.

3. (And so on.)

But remember, Corollary 5.6.4 says that since W is a subspace of F^n, which has dimension n, you can't find a linearly independent subset of W with more than n vectors in it! That means that the above process must stop at Step k for some $k \leq n$. When that happens, B is a maximal linearly independent subset of W, and by Theorem 5.6.6, B must then be a basis for W. The corollary follows. □

Finally, now that we know that every subspace has a basis, we can apply the Comparison Theorem 5.6.2 again to get the following result.

Corollary 5.6.8 (Subspace Size Theorem)**.** *If W is a subspace of a subspace V of F^n, then* $\dim W \leq \dim V \leq n$. *In particular, any subspace of F^n has dimension at most n.*

Proof. Problem 5.6.4. □

Problems

5.6.1. (*Proves Theorem 5.6.2*) Suppose F is a field with q elements, let W be a subspace of F^n, and let $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ be a linearly independent subset of W.

(a) Suppose $a_1, \ldots, a_k, b_1, \ldots, b_k \in F$ and

$$a_1\mathbf{v}_1 + \cdots + a_k\mathbf{v}_k = b_1\mathbf{v}_1 + \cdots + b_k\mathbf{v}_k. \tag{5.6.8}$$

Prove that $a_i = b_i$ for all i such that $1 \leq i \leq k$. (Suggestion: Use the definition of linear independence.)

(b) Explain why part (a) implies that span $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ (which is a subset of W) contains exactly q^k vectors.

5.6.2. (*Proves Corollary 5.6.3*) Let W be a subspace of F^n, and suppose that $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ and $\{\mathbf{u}_1, \ldots, \mathbf{u}_\ell\}$ are both bases for W. Prove that $k = \ell$. (Suggestion: Theorem 5.6.2.)

5.6.3. (*Proves Corollary 5.6.4*) Let W be a subspace of F^n such that $\dim W = k$.

(a) Suppose $\{\mathbf{v}_1, \ldots, \mathbf{v}_\ell\}$ is a linearly independent subset of W. Prove that $\ell \leq k$.

(b) Suppose $\{\mathbf{u}_1, \ldots, \mathbf{u}_s\}$ spans W. Prove that $s \geq k$.

(Suggestion: Theorem 5.6.2.)

5.6.4. (*Proves Corollary 5.6.8*) Suppose W and V are subspaces of F^n with $W \subseteq V$.

(a) Prove that $\dim W \leq \dim V$. (Suggestion: Start with bases for W and V.)

(b) Prove that $\dim V \leq n$.

Problems 5.6.5 and 5.6.6 both take a look back at the motivating situation discussed in Section 5.1, which, as we saw in Example 5.3.5, is really about a subspace $\mathcal{C}$ of $\mathbf{F}_2^5$.

5.6.5. Let $\mathcal{C}$ be the subspace defined in Section 5.1.

(a) What is the dimension of $\mathcal{C}$? Explain your answer.

(b) How can we express the idea of a minimal recovery set (as defined in Section 5.1) in terms of the theory we have developed since then?

(c) Answer Motivating Problem 5.1.1: What are the possible sizes of a minimal recovery set?

(d) Answer Motivating Problems 5.1.3 and 5.1.4: What's an efficient way to test a given set of bitstrings $\mathcal{B}$ to see if you can recover $\mathcal{C}$ from $\mathcal{B}$?

(e) Answer Motivating Problem 5.1.5: Is every minimal recovery set the same size? Explain.

5.6.6. This problem answers Motivating Problem 5.1.6 by finding a formula counting the number of bases of a k-dimensional subspace W of $\mathbf{F}_p^n$.

The rest of the problems in this section (Problems 5.6.7–5.6.9) establish some other foundational results in the theory of linear algebra. They aren't strictly necessary for the rest of this book, but they're certainly pretty interesting!

5.6.7. (*Proves Theorem 5.6.9*) The goal of this problem is to prove the following result, the catchy statement of which is due to Shahriar Shahriari [citation?].

Theorem 5.6.9 (Two Out of Three Theorem)**.** *Suppose W is a subspace of F^n with $\dim W = k$, and let S be a finite set of vectors in W. Suppose two of the following statements hold:*

1. *S spans W.*
2. *S is linearly independent.*
3. *S contains k vectors.*

Then the third statement must hold as well.

So suppose W is a subspace of F^n with $\dim W = k$, and let S be a finite set of vectors in W.

(a) Prove that if S spans W and is linearly independent, then S contains k vectors.
(b) Prove that if S spans W and contains k vectors, then S is linearly independent.
(c) Prove that if S is linearly independent and contains k vectors, then S spans W.

5.6.8. (*Proves Theorem 5.6.10*) The goal of this problem is to prove the following result.

Theorem 5.6.10 (Expansion Theorem)**.** *Given*

5.6.9. (*Proves Theorem 5.6.11*) The goal of this problem is to prove the following result.

Theorem 5.6.11 (Minimal spanning set)**.** *Let $\mathcal{B}$ be*

5.7 Row spaces and subspaces as nullspaces

Some theory used in passing later on. Enough to know that Theorem 5.7.4 is true and Corollary 5.7.5 is true.

Definition 5.7.1. Matrix transpose

Definition 5.7.2. row spaces

Theorem 5.7.3. Row(A) *is invariant under row operations*

Theorem 5.7.4. *Every subspace W of F^n with* $\dim W = k$ *is equal to* Null(A) *for some $n \times (n-k)$ matrix A.*

Corollary 5.7.5. *Row rank equals column rank. In particular, if A is a square matrix, then the rows of A are linearly independent if and only if the columns of A are linearly independent.*

Problems

5.7.1. $W^\perp$ is a subspace

5.7.2. $\dim W + \dim W^\perp = n$

5.7.3. Prove: every subspace is a nullspace

5.8 Systems of linear equations (inhomogeneous case)

[This is currently just a holding pen for various scraps cut from other sections and needs to be organized into something coherent.]

scrap:

Motivating Problem 5.8.1. Given a basis $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ for a subspace W of F^n, and a vector $\mathbf{v}$ in F^n, how can we determine if $\mathbf{v}$ is in W?

scrap: Similarly, comparing Remark 5.4.5 and the defintion of span (Definition 5.3.7), we have the following computational description of the span of a subset of F^m.

Theorem 5.8.2. *Let $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ be a subset of F^n, and let A be the $n \times k$ matrix whose columns are $\mathbf{v}_1, \dots, \mathbf{v}_k$. Then $\mathbf{b} \in F^n$ is contained in $\operatorname{span}\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ if and only if the matrix equation $A\mathbf{x} = \mathbf{b}$ has a solution.* □

scrap: Finally, considering Remark 5.4.5, we see that Motivating Problem 5.8.1 has the following matrix formulation.

Motivating Problem 5.8.3. Given an $n \times k$ matrix A over F and an $n \times 1$ column vector $\mathbf{b}$, determine if $A\mathbf{x} = \mathbf{b}$ has a solution.

Note that if

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nk} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}, \tag{5.8.1}$$

then (5.5.2) can be rewritten as $A\mathbf{x} = \mathbf{b}$, and we call $[A|\mathbf{b}]$ the *augmented matrix of the linear system* $A\mathbf{x} = \mathbf{b}$. (Note that the $|$ in the middle of $[A|\mathbf{b}]$ separates the matrix of coefficients A from the column vector of constants $\mathbf{b}$.) In those terms, we define the *solution space* of (5.5.2) to be the set of all $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix} \in F^k$ such that all equations in (5.5.2) hold.

The non-homogeneous case is slightly more complicated.

Algorithm 5.8.4. Suppose we have a homogeneous system of linear equations whose augmented matrix $[A|\mathbf{b}]$ is in RREF, and suppose A is an $n \times k$ matrix and $\mathbf{b}$ is an $n \times 1$ column vector.

1. Note that the jth column of A corresponds to the variable x_j. Call the variables corresponding to pivot columns of A the *pivot variables*, and call the other variables of the system the *free variables.*
2. If column $k+1$, the column of constants, is a pivot variable, then the bottommost nonzero translates to the equation $0=1$. In that case, the system has no solutions (empty solution set).
3. Otherwise, let $\{x_{j_1}, \dots, x_{j_f}\}$ be the f free variables. Add an equation $x_{j_m} = x_{j_m}$ for each free variable x_{j_m}; after eliminating the redundant $0=0$ equations, we now have n equations in n variables.
4. Rewrite each of the n equations as $x_i =$ (everything else in the equation moved to the right-hand side). We then see that our solution set is the set of all $\mathbf{x}$ of the form
$$\mathbf{x} = \mathbf{b}' + x_{j_1}\alpha'_1 + \dots + x_{j_f}\alpha'_f, \tag{5.8.2}$$
where each α'_m is the negative of column j_m of A, with a 1 inserted at location j_m and a 0 inserted at every other location in $j_1, \dots, j_f$; and $\mathbf{b}'$ is $\mathbf{b}$, with a 0 inserted at every location $j_1, \dots, j_f$.

Example 5.8.5. Similarly, consider the system with matrix

$$A = \left[\begin{array}{ccccc|c} 1 & 2 & 0 & 0 & -3 & 2 \\ 0 & 0 & 1 & 0 & 5 & -1 \\ 0 & 0 & 0 & 1 & 7 & 3 \end{array}\right] \tag{5.8.3}$$

Again, adding $x_2 = x_2$ and $x_5 = x_5$, we get

$$\begin{array}{ccccccccccc} x_1 & + & 2x_2 & & & & - & 3x_5 & = & 2, & \\ & & x_2 & & & & & & = & & x_2, \\ & & & & x_3 & & + & 5x_5 & = & -1, & \\ & & & & & x_4 & + & 7x_5 & = & 3, & \\ & & & & & & & x_5 & = & & x_5. \end{array} \tag{5.8.4}$$

Rearranging and rewriting in column vector form, we get

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ -1 \\ 3 \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} -2 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + x_5 \begin{bmatrix} 3 \\ 0 \\ -5 \\ -7 \\ 1 \end{bmatrix}. \tag{5.8.5}$$

Note that as promised, the column vector $\begin{bmatrix} 2 \\ 0 \\ -1 \\ 3 \\ 0 \end{bmatrix}$ is the column vector $\begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix}$ with 0 inserted in coordinates 2 and 5 (the coordinates of the free variables).

scrap: Motivating Problem 5.8.3 is solved by determining if $A\mathbf{x} = \mathbf{b}$ has a solution, by applying Gaussian Reduction and Algorithm 5.8.4.

We also note that Algorithm 5.8.4 solves Motivating Problem 5.8.3. In fact, we only need to check whether there are any rows of the form $0 = 1$; if yes, then the answer to the membership problem is no, and if no, then the answer is yes.

Problems

5.8.1. Consider the system of linear equations given by the augmented matrix

$$[A|\mathbf{b}] = \left[\begin{array}{ccccc|c} 7 & 2 & 5 & 4 & 4 & 4 \\ 7 & 1 & 2 & 2 & 2 & 4 \\ 5 & 4 & 3 & 2 & 2 & 2 \\ 4 & 1 & 2 & 6 & 3 & 1 \\ 2 & 7 & 7 & 5 & 6 & 7 \end{array}\right]. \tag{5.8.6}$$

(a) Find the solution set for $A\mathbf{x} = \mathbf{b}$, taking entries to be in the field $\mathbf{F}_2$.

(b) Same, but in the field $\mathbf{F}_3$.

(c) Same, but in the field $\mathbf{F}_5$.

(d) Same, but in the field $\mathbf{F}_7$.

5.8.2. Consider the system of linear equations given by the augmented matrix

$$[A|\mathbf{b}] = \left[\begin{array}{cccccc|c} 7 & 2 & 5 & 4 & 4 & 4 & 3 \\ 7 & 1 & 2 & 2 & 2 & 4 & 0 \\ 5 & 4 & 3 & 2 & 2 & 2 & 1 \\ 4 & 1 & 2 & 6 & 3 & 2 & 1 \\ 2 & 7 & 7 & 5 & 6 & 7 & 2 \end{array}\right]. \tag{5.8.7}$$

(a) Find the solution set for $A\mathbf{x} = \mathbf{b}$, taking entries to be in the field $\mathbf{F}_2$.

(b) Same, but in the field $\mathbf{F}_3$.

(c) Same, but in the field $\mathbf{F}_5$.

(d) Same, but in the field $\mathbf{F}_7$.

5.9 Applied and industrial topology

(to be filled in later; see Ghrist.)

While the focus of this chapter is mainly on theory, I can't resist sneaking in the following real-life question (adapted from de Silva and Ghrist [citation?]).

Suppose we have a network of cell phone booster towers, each of which covers a certain radius, as shown in Figure 5.9.1. As you can see, if you increase the radius covered by each tower, more ground will be covered, but it seems like the tower locations have been chosen in a way that there's a hole in coverage that persists even as we increase the coverage radius for each tower quite a bit.

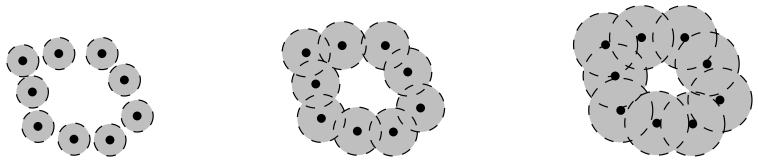

Figure 5.9.1: Cell phone coverage as tower coverage radius changes

Motivating Problem 5.9.1. There is a coverage hole in Figure 5.9.1 that seems to persist for many sizes of tower coverage radius. Is there a way to detect that hole algorithmically/automatically?

Full disclosure: The algorithm used to solve Motivating Problem 5.9.1 comes from math that is far deeper than what we discuss in this chapter. What we *can* describe in this chapter is how to make the underlying calculations of that algorithm faster and more accurate.

First, however, it's time for some cartoons.

Chapter 6

Cheaper: Error-correcting codes

Two weekends in a row I came in and found that all my stuff had been dumped and nothing was done. . . . And so I said, "Damn it, if the machine can detect an error, why can't it locate the position of the error and correct it?"

— Richard W. Hamming, as quoted in *From Error-Correcting Codes Through Sphere Packings to Simple Groups*, Thomas M. Thompson

6.1 The idea of an error-correcting code

Let's start by stating the main problem that error-correcting codes solve in terms that you may not know; then we'll gradually fill in the meaning of those terms.

Motivating Problem 6.1.1. Suppose we are transmitting a message over a noisy channel. Is there some way that we can detect, or better yet, correct errors that occur in transmission?

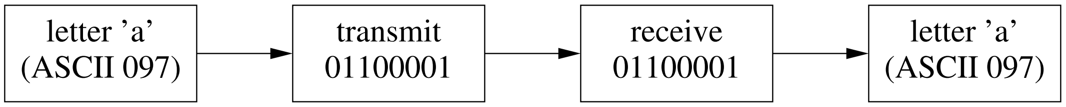

Figure 6.1.1: Successfully sending the message 'a'

By the term *message* in Motivating Problem 6.1.1, we just mean a finite sequence, or *string*, of letters. More conveniently, we can encode letters as numbers by some agreed-upon scheme; for example, the ASCII code for the lower-case letter 'a' is 97; for a more complicated example, the Unicode for the "slightly smiling face" emoji, as of this writing in 2019, is U+1F642 (using hexadecimal digits). In fact, by writing those numbers in binary (e.g., for 'a', 01100001), we can think of any message we want to send as a string of binary digits (0's and 1's), or *bits*, break those *bitstrings* into blocks of some fixed length n, and look at what happens one block at a time. See Figure 6.1.1 for a picture of what happens in a successful transmission of 'a'.

Next, the term *noisy channel* in Motivating Problem 6.1.1 refers to any means of communication that might change some of the 0's in a bitstring to 1's, and vice versa. In other

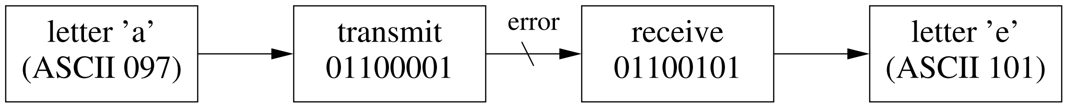

Figure 6.1.2: Sending ‘a’, receiving ‘e’

words, an *error* in the transmission of a block of n bits is a sent 0 that is received as a 1, or vice versa. For example, Figure 6.1.2 shows how if one particular bit is flipped from a 0 to a 1, a transmitted message of ‘a’ (ASCII 97) becomes a received message of ‘e’ (ASCII 101).

We can therefore restate Motivating Problem 6.1.1 in the following much more specific form.

Motivating Problem 6.1.2. Suppose we are transmitting a bitstring of length n and one or more errors occurs in transmission. Is there some way that we can detect that an error or errors has occurred? Better yet, is there some way that we can correct an error or errors?

In the terms of Motivating Problem 6.1.2, the basic idea of *error-correcting codes* is that we can detect, or even correct, transmission errors by having both sender and receiver agree ahead of time that only certain bitstrings of length n are valid messages, or *codewords*. The reason for choosing only certain bitstrings to be codewords is that if a non-codeword is received in transmission, the receiver will know that something has gone wrong. Moreover, if we choose our code very carefully, the receiver may actually be able to figure out which codeword the sender most likely intended.

For example, suppose we have n data bits $x_1, \dots, x_n$ to transmit. We can add a *parity check* bit

$$x_0 = x_1 + \cdots + x_n \pmod 2 \tag{6.1.1}$$

to our message, and transmit the bitstring $(x_0, x_1, \dots, x_n)$. (Note that the (mod 2) here indicates that we are computing in $\mathbf{F}_2$.) In other words, the codewords in the *parity check code of length* $n+1$ are all bitstrings $(x_0, x_1, \dots, x_n)$ such that (6.1.1) holds, or in other words, such that

$$x_0 + x_1 + \cdots + x_n = 0 \tag{6.1.2}$$

in $\mathbf{F}_2$. (Remember, in $\mathbf{F}_2$, $+ = -$.)

Now, suppose sender and receiver agree ahead of time to use the parity check code of length $n+1$, and in transmission, bit x_n is flipped from 0 to 1, or 1 to 0. Since $1+1=0$ in $\mathbf{F}_2$, we can model this one-bit error by adding 1 to x_n in $\mathbf{F}_2$; that is, the receiver will receive the bitstring $(x_0, x_1, \dots, x_n + 1)$. However, if the receiver then sums this bitstring, they get

$$x_0 + x_1 + \cdots + x_{n-1} + (x_n + 1) = 1, \tag{6.1.3}$$

and comparing (6.1.2), the receiver will then know that an error has occurred. Note, however, that the receiver will not know which bit has been flipped, because flipping any bit x_i (including x_0) has the same effect on the sum as flipping x_n. Note also that if *two* errors occur in transmission, everything will appear to be fine; more generally, any odd number of errors can be detected, and any even number of errors cannot be. All errors look the same, however, so Hamming's question remains: Can we do something more clever and *correct* an error that occurs?

In this chapter and in Chapter 8, we'll study efficient methods for transmitting bitstrings with error correction (e.g., in Section 6.3). However, just as the value of the Euclidean Algorithm becomes clearer when you compare its peformance to a naive algorithm for computing GCDs, it will help to set a benchmark for error correction coming from the following scheme.

Naive Algorithm 6.1.3. Suppose we want to transmit a single bit $x \in \mathbf{F}_2$. In the *repetition code* of length 3, we simply repeat our data bit 3 times. (I said this would be inefficient!) In other words, to send the message bit 0, we transmit the bitstring 000, and to send the message bit 1, we transmit 111. And indeed, this code corrects one error: If we transmit 000 and the receiver receives (say) 010, as long as no more than one error has occurred, the receiver can conclude that the original message was 000. (This procedure, of assuming that the bit that shows up more often is the intended message bit, is known as *majority logic* decoding.)

So the question arises, can we do better than the repetition code? That is:

Motivating Problem 6.1.4. Can we transmit bitstrings in blocks of some length n, with one error-correction per block, at a cost of less than 3 transmitted bits per 1 message bit?

Motivating Problem 6.1.4 also explains the title of this chapter: The point is not just that we can do error-correction in communication, but that, as we'll see by the end of this chapter, we can do it using fewer transmitted bits per message bit, i.e., *cheaper*.

Problems

6.1.1. This problem refers to variations of the repetition code.

(a) Suppose that, instead of repeating our one message bit 3 times, we repeat it 5 times, and decode using majority logic. Then how many errors can we correct?

(b) Suppose we repeat our one message bit 6 times. How many errors can we *detect*? In other words, if we transmit (say) 000000, what is the largest number of errors that can occur such that the receiver knows that our intended message is more likely, or at least equally likely, to be 0 than 1? Similarly, how many errors can we correct?

(c) Generalize the above to the repetition code where we repeat n times.

6.2 Binary linear codes

So remember all of that linear algebra stuff from Chapter 5? Let's use that to restate the basic ideas of Section 6.1, but in abstract terms.

Definition 6.2.1. We define a *bit* to be an element of $\mathbf{F}_2$, and we define a *bitstring of length n* to be an element of $\mathbf{F}_2^n$.

Remember that in the previous section, we described a code of length n as a choice of possible correctly transmitted bitstrings of length n? Well, to be precise:

Definition 6.2.2. A *code* is a subset $\mathcal{C}$ of $\mathbf{F}_2^n$. Elements (vectors) of a code are called *codewords*, and again, we think of codewords as the words that are possible correctly transmitted messages.

Note that codes in general (Definition 6.2.2) need not be particularly algebraic at all. However, since this is an algebra book, we'll focus on the following large and important class of codes.

Definition 6.2.3. A *binary linear code $\mathcal{C}$ of length n* is a subspace $\mathcal{C}$ of $\mathbf{F}_2^n$.

In other words, a binary linear code is a way to declare that only certain bitstrings of length n are valid codewords, with the valid codewords forming a subspace $\mathcal{C}$ of $\mathbf{F}_2^n$. Consequently, for the rest of this chapter, we assume, without having to restate it, that all linear algebra (matrices, vectors, etc.) is over $\mathbf{F}_2$ and all arithmetic occurs in $\mathbf{F}_2$.

To discuss how, and how well, binary linear codes work, it helps to have the following standard framework.

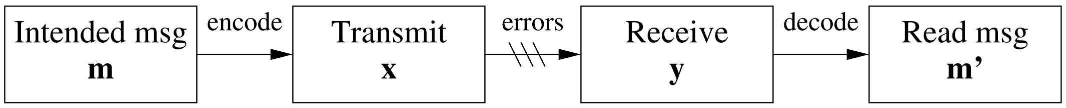

Figure 6.2.1: Standard framework for discussing error-correcting codes

Definition 6.2.4 (Standard framework)**.** When we discuss a binary linear code in operation, we use the following *standard framework*, illustrated in Figure 6.2.1.

Suppose a sender, named Xavier, wants to send bitstrings, broken up into blocks, to a receiver, named Yolanda, using the binary linear code $\mathcal{C}$. To establish common terminology and notation, here's what happens next.

1. We denote the message bitstring that Xavier wants to send by $\mathbf{m}$.

2. Xavier maps the message $\mathbf{m}$ to some codeword $\mathbf{x} \in \mathcal{C}$, a process known as *encoding.* In other words, $\mathbf{x}$ is a function of $\mathbf{m}$. This could be done, for example, by letting $\mathbf{x} = G\mathbf{m}$ for some matrix G, though other schemes are sometimes more natural or useful.

3. Xavier transmits $\mathbf{x}$ over the communications channel, and Yolanda receives $\mathbf{y}$. If no errors have occured in transmission, $\mathbf{y} = \mathbf{x}$, but if errors have occurred, then $\mathbf{y} \neq \mathbf{x}$.

4. Yoland interprets the received transmission $\mathbf{y}$ as a received message $\mathbf{m}'$, a process known as *decoding.* A communication is successful exactly when $\mathbf{m}' = \mathbf{m}$. Note that decoding can sometimes be broken down into two steps:

(a) First, Yolanda *corrects* $\mathbf{y}$ to a valid codeword $\mathbf{y}' \in \mathcal{C}$, using methods that are more often a matter of cleverness, and not just algebra.

(b) Yolanda then *reads* $\mathbf{y}'$ as a message $\mathbf{m}'$, possibly by letting $\mathbf{m}' = G'\mathbf{y}'$.

In the terms of the standard framework, we can model transmission errors in the following linear-algebraic manner. Let $\mathbf{e}_i$ be the vector in $\mathbf{F}_2^n$ whose ith coordinate is 1 and whose other coordinates are all 0. In that notation, if exactly one error happens in transmission, flipping bit i, then

$$\mathbf{y} = \mathbf{x} + \mathbf{e}_i. \tag{6.2.1}$$

(Remember: Adding 1 (mod 2) flips 0 to 1 and 1 to 0.) Similarly, if errors happen in bits i and j, we have

$$\mathbf{y} = \mathbf{x} + \mathbf{e}_i + \mathbf{e}_j. \tag{6.2.2}$$

We next look at specific examples of binary linear codes. We usually define a binary linear code $\mathcal{C}$ in one of the following two ways.

Definition 6.2.5. Let G be an $n \times k$ matrix over $\mathbf{F}_2$. To say that G is the *generator matrix* of a binary linear code $\mathcal{C}$ of length n means that $\mathcal{C} = \operatorname{Col}(G)$. In other words, generator matrices define codes as column spaces.

Definition 6.2.6. Let H be a $k \times n$ matrix over $\mathbf{F}_2$. (Note that the k and and the n are swapped, compared to the dimensions in Definition 6.2.5.) To say that H is the *parity check matrix* of a binary linear code $\mathcal{C}$ of length n means that $\mathcal{C} = \operatorname{Null}(H)$. In other words, parity check matrices define codes as nullspaces.

By the definition of nullspace (Definition 5.4.8), if H is the parity check matrix of a code $\mathcal{C}$ of length n, then $\mathbf{x} \in \mathbf{F}_2^n$ is a codeword exactly when $H\mathbf{x} = \mathbf{0}$. It follows that, for a non-codeword $\mathbf{x}$, the vector $H\mathbf{x} \neq \mathbf{0}$ describes why $\mathbf{x}$ is not a codeword, so it helps to give that vector a name.

Definition 6.2.7. Let H be a parity check matrix for a code $\mathcal{C}$ of length n. For $\mathbf{x} \in \mathbf{F}_2^n$, we define $H\mathbf{x}$ to be the *syndrome* of $\mathbf{x}$. Again, codewords in $\mathcal{C}$ are precisely the vectors $\mathbf{x}$ with syndrome equal to $\mathbf{0}$.

Returning to the two examples of families of codes given in Section 6.1, it turns out that one example is conveniently defined using a parity check matrix, and the other is conveniently defined using a generator matrix.

Example 6.2.8 (Parity check code)**.** The *parity check code of length* $n+1$ is defined to be the nullspace $\mathcal{C}$ of the $1 \times (n+1)$ matrix $H = [1 \ \dots \ 1]$. In other words, $\mathbf{x} \in \mathbf{F}_2^{n+1}$ is in $\mathcal{C}$ exactly when $H\mathbf{x} = 0$ (a 1×1 matrix!). Problem 6.2.1 gives an equivalent description of $\mathcal{C}$ in terms of a generator matrix.

To send the message bitstring $\mathbf{m} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$, Xavier adds the parity check bit

$$x_0 = x_1 + \cdots + x_n \tag{6.2.3}$$

and transmits $\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$. When Yolanda receives $\mathbf{y}$, she can check the validity of $\mathbf{y}$ by calculating $H\mathbf{y}$. If $\mathbf{y} = \mathbf{x}$, then $H\mathbf{y} = H\mathbf{x} = 0$, but if bit i has been flipped, or in other words, $\mathbf{y} = \mathbf{x} + \mathbf{e}_i$, then

$$H\mathbf{y} = H\mathbf{x} + H\mathbf{e}_i = 0 + 1 = 1. \tag{6.2.4}$$

Unfortunately, since the same error message is produced no matter which bit is flipped, Yolanda cannot tell which bit has been flipped, and she can only ask Xavier to resend. More generally, any odd number of errors can be detected, but not corrected, and any even number of errors will not be detected. (See Problem 6.2.2.)

Example 6.2.9 (Repetition code)**.** Let n be an integer. The *repetition code of length n* is defined to be the span $\mathcal{C}$ of the (single) column of the $n \times 1$ generator matrix

$$G = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}. \tag{6.2.5}$$

Since the only possible linear combinations of $\{G\}$ are G itself and the zero vector, $\mathcal{C} = \{\mathbf{0}, G\}$. Problem 6.2.3 gives an equivalent description of $\mathcal{C}$ in terms of a parity check matrix.

To send the message bit $m \in \mathbf{F}_2$, Xavier transmits the vector mG; that is, if $m = 1$, Xavier transmits $\mathbf{x} = G$, and if $m = 0$, Xavier transmits $\mathbf{x} = \mathbf{0}$. When Yolanda receives $\mathbf{y}$, she can check the validity of $\mathbf{y}$ by check if all of its bits are equal. If not all of the bits of $\mathbf{y}$ are equal, then Yolanda can correct $\mathbf{y}$ to $\mathbf{y}'$ by the *majority logic* method of choosing whichever of 0 or 1 occurs more often. If $n = 2m + 1$ is odd, she may thereby correct up to m errors, and if $n = 2m$ is even, she can correct up to $m - 1$ errors and detect, but not correct, m errors; see Problem 6.1.1.

Problems

6.2.1. Let $\mathcal{C}$ be the nullspace of the 1×9 matrix

$$H = [1 \ \ldots \ 1]. \tag{6.2.6}$$

In other words, let $\mathcal{C}$ be the parity check code of length 9 (8 data bits, 1 parity check bit).

(a) Use our standard methods to find a basis $\mathcal{B}$ for $\mathcal{C}$.

(b) What is the dimension of $\mathcal{C}$? In general, what will the dimension of the parity check code of length n be?

(c) Let G be the matrix whose columns are the vectors in the basis $\mathcal{B}$ from part (a). For a given message bitstring $\mathbf{m}$, explain why the encoding procedure of Example 6.2.8 is equivalent to transmitting $\mathbf{x} = G\mathbf{m}$.

6.2.2. Let $\mathcal{C}$ be the parity check code of length 9, i.e., the nullspace of the 1×9 matrix

$$H = [1 \ \dots \ 1]. \tag{6.2.7}$$

Use linear algebra to explain why, when you use $\mathcal{C}$, any odd number of errors will be detected, but any even number of errors will not be detected. (Suggestion: How can you express the received message $\mathbf{y}$ in terms of the transmitted message $\mathbf{x}$?)

6.2.3. Let $\mathcal{C}$ be the repetition code of length 5 (Example 6.2.9).

(a) What is the dimension of $\mathcal{C}$? Explain your answer.

(b) Let $\mathcal{C}'$ be the code of length 5 consisting of all $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_5 \end{bmatrix}$ such that $x_1 = x_2$ (i.e., the first two coordinates of $\mathbf{x}$ are the same). Find a parity check matrix $H_{1,2}$ (so named because it should contain the conditions that set $x_1 = x_2$) for $\mathcal{C}'$. (Suggestion: What system of linear equations defines $\mathcal{C}'$?)

(c) Now suppose H is a $k \times n$ parity check matrix for $\mathcal{C}$ (still the repetition code of length 5). What is the value of n? What is the *smallest* possible value of k? (Suggestion: Rank-nullity.)

(d) Find a parity check matrix for $\mathcal{C}$ of smallest possible size, as found in part (c).

6.3 The Hamming 7- and 8-codes

As described in the epigraph of this chapter, legend has it that Hamming invented error-correcting codes out of frustration with being able to detect errors automatically (by a parity check code). After quite a bit of experimention (see Thompson [?] for an entertaining and enlightening account), here's what Hamming eventually devised to solve his problem.

Definition 6.3.1. The *Hamming 7-code* $\mathcal{H}_7$ can be defined in one of two ways:

- $\mathcal{H}_7$ is the nullspace of the parity check matrix

$$H_7 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{6.3.1}$$

- $\mathcal{H}_7$ is the column space of the generator matrix

$$G_7 = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{6.3.2}$$

Note that the parity check matrix H_7 has the very useful property that the ith column of H_7 is precisely the binary digits of the number i, written upside down (i.e., the top digit is the 1's digit, then the 2's, then the 4's), which just happens to imply that H_7 is in RREF. Applying our usual methods shows that that the columns of the generator matrix G_7 are a basis for $\mathrm{Null}(H_7)$. (For a related phenomenon, see Problem 6.3.1.)

Following our standard framework (Definition 6.2.4), here's how the Hamming 7-code is used in practice.

Algorithm 6.3.2. Suppose Xavier wants to send a message bitstring $\mathbf{m} \in \mathbf{F}_2^4$ to Yolanda.

1. Xavier encodes $\mathbf{m}$ by taking $\mathbf{x} = G_7\mathbf{m}$. That is, bits x_3, x_5, x_6, and x_7 are precisely the contents of $\mathbf{m}$, and, reading (6.3.1) as a system of linear equations, the other bits x_1, x_2, and x_4 satisfy
$$\begin{aligned} x_1 &= x_3 + x_5 + x_7 \\ x_2 &= x_3 + x_6 + x_7 \\ x_4 &= x_5 + x_6 + x_7. \end{aligned} \tag{6.3.3}$$

2. Xavier transmits $\mathbf{x}$ and Yolanda receives $\mathbf{y}$.

3. Yolanda then decodes $\mathbf{y}$, using the following procedure.

   (a) First, Yolanda corrects $\mathbf{y}$ to a codeword $\mathbf{y}'$ as follows. Let $\mathbf{s} = H_7\mathbf{y} \in \mathbf{F}_2^3$ be the syndrome of $\mathbf{y}$ (Definition 6.2.7). If $\mathbf{s} = \mathbf{0}$, then $\mathbf{y}$ is a codeword of $\mathcal{H}_7$, so Yolanda chooses $\mathbf{y}' = \mathbf{y}$ (the most reasonable assumption). Otherwise, Yolanda reads $\mathbf{s}$ as the binary digits of a number i, assumes that a transmission error has occurred in bit i, and chooses $\mathbf{y}'$ to be $\mathbf{y}$ with its ith bit flipped (i.e., $\mathbf{y}' = \mathbf{y} + \mathbf{e}_i$).

   (b) To finish, Yolanda reads the message $\mathbf{m}'$ off of bits 3, 5, 6, and 7 of $\mathbf{y}'$.

Example 6.3.3. Suppose Xavier wants to send the message $\mathbf{m} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$. Using matrix multiplication or (6.3.3), he calculates

$$\mathbf{x} = G_7\mathbf{m} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \tag{6.3.4}$$

(check this yourself) and transmits $\mathbf{x}$. If a transmission error now occurs in, for example, bit 3, Yolanda then receives

$$\mathbf{y} = \mathbf{x} + \mathbf{e}_3 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}. \tag{6.3.5}$$

In the correction stage, Yolanda calculates

$$H_7\mathbf{y} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \tag{6.3.6}$$

which she interprets as $i = 1(2^0) + 1(2^1) + 0(2^2) = 3$. She therefore chooses $\mathbf{y}' = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$ (i.e., $\mathbf{y}$ with its 3rd bit changed) and reads the message $\mathbf{m}' = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$ from bits 3, 5, 6, and 7 of $\mathbf{y}'$.

The remarkable feature of the Hamming 7-code is that Example 6.3.3 is no accident: Any single transmission error can be corrected. To be precise, we have the following theorem.

Theorem 6.3.4. *In our standard framework (Definition 6.2.4), if $\mathbf{y} = \mathbf{x} + \mathbf{e}_i$ (i.e., a single transmission error occurs, in bit i), then the syndrome $\mathbf{s} = H_7\mathbf{y}$ is precisely the binary digits of i. Consequently, any single transmission error can be corrected (by flipping the ith bit back).*

Proof. Since $\mathbf{x} \in \mathcal{H}_7 = \mathrm{Null}(H_7)$, $H_7\mathbf{x} = \mathbf{0}$. Therefore,

$$\mathbf{s} = H_7\mathbf{y} = H_7(\mathbf{x} + \mathbf{e}_i) = H_7\mathbf{x} + H_7\mathbf{e}_i = \mathbf{0} + H_7\mathbf{e}_i = H_7\mathbf{e}_i, \tag{6.3.7}$$

the ith column of H_7. (Remember the description of matrix-vector multiplcation as a linear combination of columns (Remark 5.4.5).) However, H_7 is defined to be the matrix whose ith column is the binary digits of i (Definition 6.3.1), so the theorem follows. □

We can also extend $\mathcal{H}_7$ in the following way.

Definition 6.3.5. The *Hamming 8-code* $\mathcal{H}_8$ is defined to be the nullspace of the parity check matrix

$$H_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}. \tag{6.3.8}$$

Note that if we delete the first row and first column of H_8, we get H_7, the parity check matrix for the Hamming 7-code. Therefore, to be consistent with the Hamming 7-code, we write an arbitrary element of $\mathcal{H}_8$ as $\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_7 \end{bmatrix} \in \mathcal{H}_8$.

The key properties of the Hamming 8-code are summarized in the following theorem; the details are left for you to explore in Problems 6.3.4–6.3.6.

Theorem 6.3.6. *The Hamming 8-code $\mathcal{H}_8$ is the Hamming 7-code $\mathcal{H}_7$, extended by a parity check bit x_0; and $\mathcal{H}_8$ corrects 1 error and detects 2 errors.*

Proof. Problem 6.3.4 explains why x_0 is a parity check bit; Problem 6.3.5 determines a generator matrix for $\mathcal{H}_8$ and works out its dimension; and Problem 6.3.6 describes decoding. □

Problems

6.3.1. Let H_7 and G_7 be the parity check and generator matrices for the Hamming 7-code (Definition 6.3.1).

(a) Find the matrix product H_7G_7.

(b) Now suppose H and G are parity check and generator matrices for the same code $\mathcal{C}$, and suppose that H is in RREF and the columns of G are linearly independent. Suppose also that $\mathcal{C}$ has length n and dimension k. What will the matrix product HG be? (For example, what will the size of HG be?)

6.3.2. Let $\mathcal{H}_7$ be the Hamming 7-code (Definition 6.3.1).

(a) Write out all of the codewords in $\mathcal{H}_7$.

(b) Find all shortest nonzero codewords in $\mathcal{H}_7$ (i.e., the nonzero codewords with the fewest number of 1's).

6.3.3. Suppose Yolanda is receiving transmissions sent using the Hamming 7-code, and she receives $\mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$. Correct $\mathbf{y}$ to a codeword $\mathbf{y}'$, if necessary, and read off the message bits 3, 5, 6, and 7 to find the intended message $\mathbf{m}'$. Do the same for $\mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$ and $\mathbf{y} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$.

6.3.4. Let $\mathcal{H}_7$ and $\mathcal{H}_8$ be the Hamming 7-code and 8-code, respectively (Definitions 6.3.1 and 6.3.5). Prove that $\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_7 \end{bmatrix} \in \mathcal{H}_8$ exactly when $\begin{bmatrix} x_1 \\ \vdots \\ x_7 \end{bmatrix} \in \mathcal{H}_7$ and $x_0 = x_1 + \cdots + x_7$ (i.e., x_0 is a parity check bit for $\begin{bmatrix} x_1 \\ \vdots \\ x_7 \end{bmatrix}$). (Suggestion: Compare the matrices H_7 and H_8.)

6.3.5. Let $\mathcal{H}_8$ be the Hamming 8-code (Definition 6.3.5).

(a) Use the fact that $\mathcal{H}_8 = \text{Null}(H_8)$ to find a basis for $\mathcal{H}_8$.

(b) Find the dimension of $\mathcal{H}_8$.

6.3.6. Let $\mathcal{H}_8$ be the Hamming 8-code (Definition 6.3.5), let $\mathbf{x} \in \mathcal{H}_8$ be a transmitted codeword, and let $\mathbf{y}$ be the corresponding received transmission. Let $\mathbf{s} = H_8\mathbf{y} = \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}$ be the syndrome of $\mathbf{y}$.

(a) Suppose $\mathbf{y}$ is $\mathbf{x}$ with a single bit flipped, or in other words, $\mathbf{y} = \mathbf{x} + \mathbf{e}_i$ $(0 \leq i \leq 7)$. Explain why it must be the case that $s_0 = 1$, and explain how to use (s_1, s_2, s_3) to figure out what i is. (Suggestion: Imitate the proof of Theorem 6.3.4.)

(b) Suppose $\mathbf{y}$ is $\mathbf{x}$ with two bits flipped, or in other words, $\mathbf{y} = \mathbf{x} + \mathbf{e}_i + \mathbf{e}_j$ $(0 \leq i, j \leq 7$, $i \neq j)$. Explain why it must be the case that $s_0 = 0$, and explain why we cannot have $(s_1, s_2, s_3) = (0, 0, 0)$.

(c) Suppose again that $\mathbf{y} = \mathbf{x} + \mathbf{e}_i + \mathbf{e}_j$. Find two different pairs i, j with $0 \leq i, j \leq 7$ and $i \neq j$ that produce the same syndrome $H_8\mathbf{y}$. (This is why we can only detect two errors, not correct them.)

6.4 Hamming distance and error correction

Now, even after going through the details (or maybe even more so after going through the details!), it may seem somewhat miraculous that the Hamming 7- and 8-codes work the way they do. If you're in an enterprising mood, however, you might wonder: How can we come up with some other code that's just as good, or maybe even better? To tackle that question, it helps to have a concrete measure of what makes a code "good".

Notation 6.4.1. An $[n, k, d]$ *binary code* is a binary linear code $\mathcal{C}$ such that:

- $\mathcal{C}$ has length n;
- $\dim \mathcal{C} = k$; and
- d is the smallest number of nonzero coordinates appearing in a nonzero codeword of $\mathcal{C}$.

The numbers n, k, and d are called the *length*, *dimension*, and *minimum distance* of $\mathbf{C}$, respectively.

Example 6.4.2. Let $\mathcal{C}$ be the parity check code of length $n+1$ (Example 6.2.8). By rank-nullity (Corollary 5.5.15), $\dim C = n$, and since vectors are codewords of $\mathcal{C}$ exactly when they contain an even number of 1's, the minimum distance of $\mathcal{C}$ is 2. In other words, $\mathcal{C}$ is an $[n+1, n, 2]$ code.

Example 6.4.3. Let $\mathcal{C}$ be the repetition code of length n (Example 6.2.9). By the definition of dimension, $\dim C = 1$, and since the all-1's vector is the only nonzero codeword of $\mathcal{C}$, the minimum distance of $\mathcal{C}$ is n. In other words, $\mathcal{C}$ is an $[n, 1, n]$ code.

Note that if $\mathcal{C}$ is an $[n, k, d]$ code, then k gives the number of "message bits" contained in each codeword, and, as we will see momentarily, d determines the number of errors that can be detected or corrected (at least in principle). The game of error-correcting can therefore be summarized as trying to maxmimize both k and d for a given length n.

Example 6.4.4. By listing all of the vectors of $\mathcal{H}_7$, we see that $\mathcal{H}_7$ has minimum distance 3. Combining that with what we already know about $\mathcal{H}_7$, we see that $\mathcal{H}_7$ is a $[7, 4, 3]$ code. Similarly, Problems 6.3.5 and 6.4.1 together show that $\mathcal{H}_8$ is an $[8, 4, 4]$ code.

To analyze the relationship between the minimum distance of a code and its error-correction properties, we'll need the following fundamental idea.

Definition 6.4.5. Let $\mathbf{x}, \mathbf{y}$ be vectors in $\mathbf{F}_2^n$ (or more generally, F^n for a field F). The *Hamming distance* between $\mathbf{x}$ and $\mathbf{y}$ is:

$$\begin{aligned} d(\mathbf{x},\mathbf{y}) &= \text{the number of coordinates in which } \mathbf{x} \text{ and } \mathbf{y} \text{ differ} \\ &= \text{the number of nonzero coordinates in } \mathbf{x}-\mathbf{y} \qquad (6.4.1) \\ &= \text{the number of coordinate changes needed to go from } \mathbf{x} \text{ to } \mathbf{y}. \end{aligned}$$

The *Hamming weight* $\mathrm{wt}(\mathbf{x})$ of $\mathbf{x}$ is the number of nonzero coordinates of $\mathbf{x}$. In other words,

$$\mathrm{wt}(\mathbf{x}) = d(\mathbf{x}, 0), \qquad d(\mathbf{x},\mathbf{y}) = \mathrm{wt}(\mathbf{x}-\mathbf{y}). \tag{6.4.2}$$

Example 6.4.6. The Hamming distance between $\mathbf{x} = \begin{bmatrix}1\\0\\0\\1\\0\end{bmatrix}$ and $\mathbf{y} = \begin{bmatrix}1\\1\\0\\1\\1\end{bmatrix}$ is 2 because $\mathbf{x}$ and $\mathbf{y}$ differ in exactly coordinates 2 and 5.

The following idea may help you picture why we call Hamming distance a "distance".

Definition 6.4.7. A *Hamming path of length k* in $\mathbf{F}_2^n$ is a sequence $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_k \in \mathbf{F}_2^n$ such that for $1 \le i \le k$, the vectors $\mathbf{x}_{i-1}$ and $\mathbf{x}_i$ differ in exactly one coordinate (i.e., $\mathbf{x}_i - \mathbf{x}_{i-1}$ has exactly one nonzero coordinate). We also say that the path $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_k$ goes from $\mathbf{x}_0$ to $\mathbf{x}_k$.

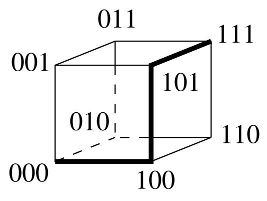

Figure 6.4.1: A Hamming path in $\mathbf{F}_2^3$

For example, Figure 6.4.1 shows a Hamming path of length 3 between 000 and 111 in $\mathbf{F}_2^3$. To be precise, the edges of the cube represent all Hamming paths of length 1, and the highlighted path is the sequence $000, 100, 101, 111$.

Theorem 6.4.8. *For $\mathbf{x}, \mathbf{y} \in \mathbf{F}_2^n$, the Hamming distance $d(\mathbf{x},\mathbf{y})$ is precisely the length of a shortest Hamming path from $\mathbf{x}$ to $\mathbf{y}$.*

Proof. Suppose $d(\mathbf{x},\mathbf{y}) = k$, or in other words, $\mathbf{x}$ and $\mathbf{y}$ differ in exactly k coordinates. In that case, one the one hand, by starting with $\mathbf{x}$ and changing one coordinate at a time, we get a Hamming path from $\mathbf{x}$ to $\mathbf{y}$ of length k; and on the other hand, if $\mathbf{x} = \mathbf{x}_0, \dots, \mathbf{x}_m = \mathbf{y} \in \mathbf{F}_2^n$ is a Hamming path of length m from $\mathbf{x}$ to $\mathbf{y}$, then since we can only change one coordinate when we go from $\mathbf{x}_{i-1}$ to $\mathbf{x}_i$, the length m of this path must be at least k. The theorem follows. $\square$

Corollary 6.4.9. *Hamming distance is translation invariant. That is, for* $\mathbf{x}, \mathbf{y}, \mathbf{c} \in \mathbf{F}_2^n$*, we have that*

$$d(\mathbf{x}+\mathbf{c}, \mathbf{y}+\mathbf{c}) = d(\mathbf{x}, \mathbf{y}). \tag{6.4.3}$$

Proof. If $\mathbf{x}_{i-1}$ and $\mathbf{x}_i$ differ in exactly one coordinate, then so do $\mathbf{x}_{i-1}+\mathbf{c}$ and $\mathbf{x}_i+\mathbf{c}$, since

$$(\mathbf{x}_i+\mathbf{c}) - (\mathbf{x}_{i-1}+\mathbf{c}) = \mathbf{x}_i - \mathbf{x}_{i-1}. \tag{6.4.4}$$

Therefore, $\mathbf{x} = \mathbf{x}_0, \dots, \mathbf{x}_k = \mathbf{y} \in \mathbf{F}_2^n$ is a Hamming path from $\mathbf{x}$ to $\mathbf{y}$ if and only if $\mathbf{x}+\mathbf{c} = \mathbf{x}_0+\mathbf{c}, \mathbf{x}_1+\mathbf{c}, \dots, \mathbf{x}_k+\mathbf{c} = \mathbf{y}+\mathbf{c} \in \mathbf{F}_2^n$ is a Hamming path from $\mathbf{x}+\mathbf{c}$ to $\mathbf{y}+\mathbf{c}$, and the corollary follows from Theorem 6.4.8. □

The mathematical version of a measure of distance is the following abstract construction.

Definition 6.4.10. A *metric* on a set X is a function $d : X \times X \to \mathbf{R}$ (i.e., two inputs in X, output is a real number) that satisfies the following four axioms for all $x, y, z \in X$:

1. $d(x, y) \geq 0$.
2. $d(x, y) = 0$ if and only if $x = y$.
3. $d(x, y) = d(y, x)$.
4. (Triangle inequality) $d(x, z) \leq d(x, y) + d(y, z)$.

Theorem 6.4.11. *Hamming distance* $d(\mathbf{x}, \mathbf{y})$ *is a metric on* $\mathbf{F}_2^n$*.*

Proof. The first and third axioms of Definition 6.4.10 hold for Hamming distance because the number of coordinates in which $\mathbf{x}$ and $\mathbf{y}$ differ is a nonnegative number defined symmetrically in $\mathbf{x}$ and $\mathbf{y}$. The second axiom holds because two vectors differ in 0 coordinates if and only if they are equal. The last axiom holds because of Problem 6.4.2. □

The following theorem quantifies the idea that if a binary linear code $\mathcal{C}$ has a higher minimum distance, then $\mathcal{C}$ will be able correct and detect more errors, at least in principle. (Compare Problem 6.1.1.) Namely, we can use the following error-correction algorithm.

Algorithm 6.4.12. The *nearest neighbor* error-correction method, applied to a code $\mathcal{C}$, is the following procedure for error correction, expressed in the terms of our standard framework (Definition 6.2.4).

- If there is a unique $\mathbf{y}' \in \mathcal{C}$ such that $d(\mathbf{y}, \mathbf{y}')$ is minimized, we correct $\mathbf{y}$ to $\mathbf{y}'$. (For example, if $\mathbf{y} \in \mathcal{C}$, then $\mathbf{y}' = \mathbf{y}$ is the unique codeword that minimizes $d(\mathbf{y}, \mathbf{y}')$, as $d(\mathbf{y}, \mathbf{y}) = 0$.)
- If there is more than one vector $\mathbf{y}' \in \mathcal{C}$ such that $d(\mathbf{y}, \mathbf{y}')$ is minimized, we state that $\mathbf{y}$ has been detected as an erroneous transmission, but cannot be corrected. (The idea is that the different $\mathbf{y}'$ minimizing $d(\mathbf{y}, \mathbf{y}')$ are equally probable as intended transmissions.)

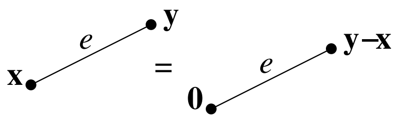

Figure 6.4.2: A distance between codewords anywhere also occurs at $\mathbf{0}$

Theorem 6.4.13. *Let $\mathcal{C}$ be a binary linear code with minimum distance d. Then the nearest neighbor method, applied to $\mathcal{C}$, corrects $\lfloor (d-1)/2 \rfloor$ errors and detects $\lfloor d/2 \rfloor$ errors.*

Proof. By Theorem 6.4.8, if $d(\mathbf{x}, \mathbf{y}) = e$, then $d(\mathbf{0}, \mathbf{y} - \mathbf{x}) = d(\mathbf{x}, \mathbf{y}) = e$ (see Figure 6.4.2). It follows that the minimum distance d of $\mathcal{C}$ is precisely the minimum distance between any two distinct codewords $\mathbf{x}, \mathbf{y} \in \mathcal{C}$.

So now, suppose $e \leq \lfloor (d-1)/2 \rfloor$, or in other words, $2e + 1 \leq d$. The error-correcting claim in the theorem means precisely that the nearest neighbor method (Algorithm 6.4.12) can correct e errors.

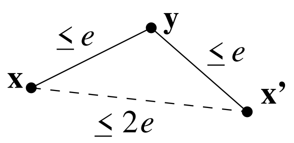

Figure 6.4.3: Two close neighbors implies two codewords close to each other

The question now is, how could the nearest neighbor method fail to correct up to e errors? Well, in order for that to happen, there would have to be some possible received transmission $\mathbf{y}$ with two neighboring codewords $\mathbf{x}$ and $\mathbf{x}'$ such that both $\mathbf{x}$ and $\mathbf{x}'$ are distance $\leq e$ away from $\mathbf{y}$, or in other words, $d(\mathbf{x}, \mathbf{y}) \leq e$ and $d(\mathbf{x}', \mathbf{y}) \leq e$. However, in that case, by Theorem 6.4.11 and the triangle inequality (Definition 6.4.10), we would have

$$d(\mathbf{x}, \mathbf{x}') \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{x}') \leq 2e, \tag{6.4.5}$$

violating our assumption that the minimum distance between two codewords is at least $2e + 1$ (see Figure 6.4.3). Therefore, $\mathbf{y}$ can have at most one neighbor within distance e, which means that the nearest neighbor method corrects up to e errors.

The analogous proof that the nearest neighbor method detects $\lfloor d/2 \rfloor$ errors is left to you (Problem 6.4.3). □

Note that the nearest neighbor method is not necessarily efficient, as it may require making a lookup table with a nearest neighbor listed for each vector of $\mathbf{F}_2^n$ (where n is the length of $\mathcal{C}$). In fact, for any particular code, it may be better to use a different method that ends up having the same effect, like our decoding method for the Hamming 7-code. The point of Theorem 6.4.13 is that for an $[n, k, d]$ code, there is at least *some* method for correcting $\lfloor (d-1)/2 \rfloor$ errors and detecting $\lfloor d/2 \rfloor$ errors, so it is valuable to look for codes with as large a minimum distance d as possible.

Problems

6.4.1. Recall that $\mathcal{H}_7$ has minimum distance 3 (Problem 6.3.2) and that $\mathcal{H}_8$ contains exactly the same codewords as $\mathcal{H}_7$, extended by a parity bit x_0 (Problem 6.3.4).

(a) Prove that $\mathcal{H}_8$ has minimum distance 4.

(b) Now suppose that $\mathcal{C}$ is an $[n, k, d]$ code, where d is odd, and that $\mathcal{C}_+$ is a code of length $n+1$ consisting of the codewords in $\mathcal{C}$, each extended by a parity bit. Prove that $\mathcal{C}_+$ has minimum distance $d+1$.

6.4.2. Prove that if $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbf{F}_2^n$, then Hamming distance $d(-,-)$ satisfies

$$d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}). \tag{6.4.6}$$

(Suggestion: Use Theorem 6.4.8.)

6.4.3. Let $\mathcal{C}$ be a binary linear code of minimum distance d, and suppose that $2e \leq d$. Prove that if $\mathbf{x}$ is a transmitted codeword and $\mathbf{y}$ is a received transmission obtained by applying e errors to $\mathbf{x}$, then there is no codeword $\mathbf{x}' \in \mathcal{C}$ that is strictly closer to $\mathbf{y}$ than $\mathbf{x}$. (Suggestion: Proceeding by contradiction, suppose $\mathbf{x}'$ is a codeword in $\mathcal{C}$ that is closer to $\mathbf{y}$ than $\mathbf{x}$ is, and imitate the proof of Theorem 6.4.13.)

6.4.4. Let $\mathcal{C}$ be a binary linear code of length n and $\mathbf{c} \in \mathcal{C}$. For a nonnegative integer r, we define $B_r[\mathbf{c}]$, the *closed r-ball around* $\mathbf{c}$ to be

$$B_r[\mathbf{c}] = \left\{\mathbf{x} \in \mathbf{F}_2^n \mid d(\mathbf{x}, \mathbf{c}) \leq r\right\}. \tag{6.4.7}$$

A code is said to be *perfect* if it is the disjoint union of the closed r-balls $B_r[\mathbf{c}]$ for some fixed r and all $\mathbf{c} \in \mathcal{C}$. The goal of this problem is to prove that the Hamming 7-code $\mathcal{H}_7$ is perfect, with $r = 1$.

(a) Suppose $\mathbf{c} \in \mathcal{H}_7$. How many vectors $\mathbf{x}$ are there in $\mathbf{F}_2^7$ such that $d(\mathbf{x}, \mathbf{c}) \leq 1$? Describe all such $\mathbf{x}$.

(b) Prove that if $\mathbf{x} \in \mathbf{F}_2^7$, $\mathbf{x}$ cannot be contained in two distinct 1-balls $B_1[\mathbf{c}]$, $B_1[\mathbf{c}']$ of codewords $\mathbf{c}, \mathbf{c}' \in \mathcal{H}_7$.

(c) Prove that every $x \in \mathbf{F}_2^7$ is contained in the 1-ball $B_1[\mathbf{c}]$ of some $\mathbf{c} \in \mathcal{H}_7$. (Suggestion: How many vectors are there in the union of all 1-balls $B_1[\mathbf{c}]$ of some $\mathbf{c} \in \mathcal{C}$?)

Chapter 7

Ideals, quotients, and finite fields

I see you're a man with ideals. I better be going before you've still got them.

— Mae West

We're building something here, detective. We're building it from scratch. All the pieces matter.

— Det. Lester Freamon, *The Wire*

7.1 Ideals

We come to a truly fundamental definition in ring theory.

Definition 7.1.1. Let R be a (commutative) ring. An *ideal* of R is a subset I of R satisfying the following three axioms:

1. (Zero) The zero element of R is contained in I.
2. (Closed under addition) If $a, b \in I$, then $a + b \in I$.
3. (Closed under R-multiplication) If $a \in I$ and $r \in R$, then $ra \in I$.

That definition doesn't look like much, does it? A little bit like subspace (Definition 5.3.3), perhaps. Well, as with any other piece of pure abstraction, if you want to start to understand what ideals are, let's look at some examples.

Example 7.1.2 (Even integers)**.** Let $R = \mathbf{Z}$, and let I be the set of even integers. Then I is an ideal because:

1. 0 is even.
2. The sum of two even numbers is even.
3. The last axiom is worth considering in more detail. The point isn't just that that the product of even numbers is even; the point is that the product of an *arbitrary* integer (even or odd) and an even number is even.

Example 7.1.3 (Silly examples)**.** For a ring R, the set $\{0\}$ is an ideal of R called the *zero ideal.* The ring R is also an ideal of itself.

Example 7.1.4 (Multiples of m)**.** For a fixed $m \in \mathbf{Z}$, the set

$$(m) = m\mathbf{Z} = \{km \mid k \in \mathbf{Z}\} \tag{7.1.1}$$

is an ideal of $\mathbf{Z}$.

That should start to look familiar; remember $\mathbf{Z}/(m)$, our notation for the integers mod m?

Example 7.1.5 (Multiples of $m(x)$)**.** Similarly, if F is a field, for a fixed $m(x) \in F[x]$, the set

$$(m(x)) = \{k(x)m(x) \mid k(x) \in F[x]\} \tag{7.1.2}$$

is an ideal of $F[x]$.

Examples 7.1.4 and 7.1.5 are both particular cases of the following construction.

Definition 7.1.6. For a ring R and a fixed $d \in R$, the set

$$(d) = \{rd \mid r \in R\} \tag{7.1.3}$$

is called the *principal ideal generated by d.*

Definition 7.1.7. For a ring R and fixed $c, d \in R$, the set

$$(c, d) = \{rc + sd \mid r, s \in R\} \tag{7.1.4}$$

is called the *ideal generated by c and d.*

See Problems 7.1.1 and 7.1.2. for proofs that the sets (d) and (c, d) are actually ideals.

Example 7.1.8 (Substitution kernel)**.** For F a field and $\alpha \in F$, the set

$$I_\alpha = \{f(x) \in F[x] \mid f(\alpha) = 0\} \tag{7.1.5}$$

is an ideal of $F[x]$ (Problem 7.1.3). Note that the substitution of $a \in F$ into a polynomial $f(x) \in F[x]$ satisfies two important properties:

- The value that you get when you plug α into $f(x) + g(x)$ is $f(\alpha) + g(\alpha)$; and
- The value that you get when you plug α into $f(x)g(x)$ is $f(\alpha)g(\alpha)$.

Since we've chosen to make polynomials look like functions, those properties seem obvious, right? But remember that polynomials aren't functions; they're algebraic objects defined by setting certain rules on how to multiply an unspecified object x, along with elements of F (Definition 3.3.1). In that context, you might even start to worry whether the above "obvious" properties are actually true! Fortunately, they are, and the proof is relatively straightforward; see Example 7.5.3 in Section 7.5. For now, you should feel free to rely on those properties in Problem 7.1.3.

So by now, you should have some idea of what an ideal *is*, but probably still no idea of why on earth you should care. For that, we'll need to see what you can *do* with ideals, which we'll start to see in the next section.

To finish this section, however, we mention the following small but important (and nonobvious) property of ideals.

Theorem 7.1.9. *Let R be a ring, and let I be an ideal of R. Then I is closed under taking negatives; that is, if $a \in I$, then $-a \in I$.*

Proof. If $a \in I$, then since $-1 \in R$ (in any ring), and I is closed under multiplication by elements of R, $(-1)a = -a \in I$. □

Problems

7.1.1. Let R be a commutative ring and $a \in R$, and define

$$(d) = \{rd \mid r \in R\}. \tag{7.1.6}$$

The goal of this problem is to prove that (d) is an ideal of R.

(a) Explain how you know that $0 \in (d)$.

(b) What do two random elements of (d) look like? Explain why their sum must be in (d).

(c) For $s \in R$ and $a \in (d)$, explain why $sa \in (d)$.

7.1.2. Let R be a commutative ring and $c, d \in R$, and define

$$(c, d) = \{rc + sd \mid r, s \in R\}. \tag{7.1.7}$$

The goal of this problem is to prove that (c, d) is an ideal of R.

(a) Explain how you know that $0 \in (c, d)$.

(b) What do two random elements of (c, d) look like? Explain why their sum must be in (c, d).

(c) For $t \in R$ and $a \in (c, d)$, explain why $ta \in (c, d)$.

7.1.3. Let F be a field and $\alpha \in F$, and define

$$I_\alpha = \{f(x) \in F[x] \mid f(\alpha) = 0\}. \tag{7.1.8}$$

The goal of this problem is to prove that I_α is an ideal of $F[x]$.

(a) Explain how you know that $0 \in I_\alpha$.

(b) Suppose $f(x), g(x) \in I_\alpha$. Explain why $f(x) + g(x) \in I_\alpha$.

(c) For $h(x) \in F$ and $f(x) \in I_\alpha$, explain why $h(x)f(x) \in I_\alpha$.

7.2 Quotient rings

One of the main purposes of defining ideals is to be able to kill them. That is to say, our first (and most important) application of ideals is the construction of what are known as *quotient rings*, or in other words, rings in which we have set certain ideals equal to 0. In particular, after working with $\mathbf{Z}/(m)$ since back in Chapter 3, we finally give a real definition of $\mathbf{Z}/(m)$!

First, however, we need another new piece of abstraction, one just as important as the definition of ideals. (We'll see an important variation on this definition in Chapter 10.)

Definition 7.2.1. Let R be a ring, and let I be an ideal of R. For $r \in R$, we define the *additive coset* $r + I$ to be

$$r + I = \{r + a \mid a \in I\}. \tag{7.2.1}$$

If the context is clear, instead of saying "additive coset", we just say *coset*.

Example 7.2.2. As always with ideals, let's start with the even numbers; that is, take $R = \mathbf{Z}$ and the ideal $I = (2)$ (Definition 7.1.4), or in other words, the even numbers. Since

$$I = \{\dots, -4, -2, 0, 2, 4, 6, \dots\}, \tag{7.2.2}$$

we see that

$$1 + I = \{\dots, -3, -1, 1, 3, 5, 7, \dots\} \tag{7.2.3}$$

and

$$2 + I = \{\dots, -2, 0, 2, 4, 6, 8, \dots\}. \tag{7.2.4}$$

For a less interesting example, $0 + I = I$.

However, the way we wrote the above examples is a little misleading, because the ... hide the fact that if we extend the lists for I and $2 + I$ a little bit, we see that as *sets*,

$$2 + I = \{\dots, -4, -2, 0, 2, 4, 6, 8, \dots\} = I = 0 + I. \tag{7.2.5}$$

Similarly, as sets,

$$3 + I = \{\dots, -3, -1, 1, 3, 5, 7, 9 \dots\} = 1 + I. \tag{7.2.6}$$

Continuing analogously, we see that $a + I = b + I$ exactly when a and b are either both even or both odd. In other words, even though there might appear to be infinitely many cosets of I in R, one for each $r \in \mathbf{Z}$, in fact, there are only *two:* $0 + I$ and $1 + I$.

Example 7.2.3. As we'll see, the ideal $(5) = 5\mathbf{Z}$ (Example 7.1.4) has exactly 5 cosets in $\mathbf{Z}$:

$$\begin{aligned} 0 + (5) &= (5) = \{\dots, 0, 5, 10, 15, \dots\}, \\ 1 + (5) &= \{\dots, 1, 6, 11, 16, \dots\}, \\ 2 + (5) &= \{\dots, 2, 7, 12, 17, \dots\}, \\ 3 + (5) &= \{\dots, 3, 8, 13, 18, \dots\}, \\ 4 + (5) &= \{\dots, 4, 9, 14, 19, \dots\}. \end{aligned} \tag{7.2.7}$$

Note that $5 + (5) = (5)$ (write it out!), and more generally, you should start to see that the cosets of (5) resemble the structure of $\mathbf{Z}/(5)$. And of course, there's nothing special about 5; for any positive integer m, the ideal $(m) = m\mathbf{Z}$ has exactly m cosets, which we can list in exactly the same way (Problem 7.2.1).

Our next task is to show that belonging to the same coset generalizes the idea of "congruent mod m" (Definition 3.1.4).

Theorem 7.2.4. *Let R be a ring, let I be an ideal of R, and let $r, s \in R$. Then the following are equivalent (one statement holds if and only the other holds):*

1. *$r + I = s + I$ (i.e., the cosets $r + I$ and $s + I$ are the same set).*
2. *$r \in s + I$.*
3. *$r - s \in I$.*
4. *$r = s + a$ for some $a \in I$.*

If you look back at Definition 3.1.4, you'll see that conditions (3) and (4) are generalizations of the definition of being congruent mod m. In fact, you may find it convenient to think of being in the same coset of I as being "congruent mod I".

Proof. To prove that the four statements are equivalent, we need to show that (1) implies (2), (2) implies (3), (3) implies (4), and (4) implies (1). We do two implications, and leave the others for you.

Suppose (1) holds. That means that every element of the form $r + a$ with $a \in I$ is equal to some $s + b$ for some $b \in I$, and vice versa. In particular, this holds for $a = 0 \in I$, so $r = r + 0 = s + b$ for some $b \in A$. By definition of coset, $r \in s + I$, which means that (2) holds.

For (2) implies (3), see Problem 7.2.2.

Suppose (3) holds. Then $r - s = a$ for some $a \in I$, which means that $r = s + a$ for some $a \in I$.

For (4) implies (1), see Problem 7.2.3. □

It will also be useful to have a name for condition (2) of Theorem 7.2.4.

Definition 7.2.5. Let R be a ring, let I be an ideal of R, and let $r, s \in R$. To say that r is a *representative of the coset $s + I$* means that $r \in s + I$. Note that by Theorem 7.2.4, a coset can be represented by any of its elements, because r and s are contained in the same coset exactly when $r + I = s + I$.

The question of different representatives for the same coset is the heart of the following key definition.

Definition 7.2.6. Let R be a ring and let I be an ideal of R. We define the *quotient ring* R/I as follows.

- **Set:** The elements of R/I are the cosets of I in R. Note that if r and s represent the same coset of I, then the cosets $r + I$ and $s + I$ are actually the same element of R/I, since $r + I = s + I$.
- **Addition:** For $r + I, s + I \in R/I$, we define the sum
$$(r + I) + (s + I) = (r + s) + I. \tag{7.2.8}$$
- **Multiplication:** For $r + I, s + I \in R/I$, we define the product
$$(r + I)(s + I) = rs + I. \tag{7.2.9}$$

The zero element of R/I is $0 + I = I$, and the one element is $1 + I$.

Remark 7.2.7. Let R be a ring and let I be an idea of R. Since I is the zero element of the ring R/I, we can think of the quotient ring R/I as "R with I set to 0". Note that if $I = (m)$ is the principal ideal generated by $m \in R$, then $I = (m)$ is exactly the set of all necessary algebraic consequences of setting $m = 0$, because if we set $m = 0$, multiplying both sides by r, we see that all multiples rm ($r \in R$) of m must also be set to 0. We can therefore think of $R/(m)$ as "R setting $m = 0$", a generalization of the $m = 0$ Principle of Section 3.1.

The tricky part of Definition 7.2.6 is determining whether the operations in a quotient ring are *well-defined*, or in other words, unambiguous. To understand the issue here, it's important to ask the following question.

Ask Yourself 7.2.8. What could possibly be ambiguous about Definition 7.2.6? Specifically, is there more than one way you could possibly interpret (7.2.8) and (7.2.9)?

Ask Yourself 7.2.9. Seriously, ask yourself: What could possibly be fishy about Definition 7.2.6?

Before we answer Ask Yourself 7.2.8, let's consider one specific, and actually familiar, example.

Example 7.2.10. Taking $R = \mathbf{Z}$, consider the ideal $I = (2)$ (the even integers) of $\mathbf{Z}$. The quotient ring $\mathbf{Z}/(2)$ has two elements, namely, the two cosets of I, which are $0 + I = I$ (the set of even numbers) and $1 + I$ (the set of odd numbers). Applying (7.2.8), we see, for example, that
$$(1 + I) + (1 + I) = 2 + I = I, \tag{7.2.10}$$
since 2 is in the set of even numbers $0 + I = I$ (by Theorem 7.2.4).

Similarly, consider the ideal $I = (3)$ of $\mathbf{Z}$ (the integers divisible by 3). The quotient ring $\mathbf{Z}/(3)$ has three elements, I, $1 + I$, and $2 + I$, and applying (7.2.9), we see, for example, that
$$(2 + I)(2 + I) = 4 + I = 1 + I, \tag{7.2.11}$$

since $4 \in 1+I$. Since $-1+I = 2+I$, we could equivalently see that

$$(2+I)(-1+I) = -2+I = 1+I, \tag{7.2.12}$$

since, again, $-2 \in 1+I$, as -2 and 1 are congruent mod I.

More generally, let $I = (m)$ for some positive integer m. If m is understood, by saying that we are working in $\mathbf{Z}/(m)$, and we regard integers a, b, etc. "in $\mathbf{Z}/(m)$" as abbreviations for the cosets $a+I$, $b+I$, etc., then we see that Definition 3.1.3 is just a special case of Definition 7.2.6. (In particular, if $I = (2)$ or (3), then we get exactly the operations described in Example 3.1.6.) Furthermore, the Congruent Subsitution Principle of Section 3.1 holds precisely because we are choosing different representatives for the same coset.

We are now ready to show that Definition 7.2.6 genuinely defines a ring R/I. Again, the main issue is whether the ring operations (7.2.8) and (7.2.9) are well-defined.

Theorem 7.2.11. *Let R be a ring and let I be an ideal of R. Then the operations (7.2.8) and (7.2.9) in R/I are well-defined and give R/I the structure of a ring with zero element I and one element $1+I$.*

Proof. Again, the main issue is whether the operations (7.2.8) and (7.2.9) are well-defined. To finally give an answer to Ask Yourself 7.2.8, the problem with (7.2.8) and (7.2.9) is that the sum $(r+I)+(s+I)$ and product $(r+I)(s+I)$ seem to depend on the choice of coset representatives r and s. So what we need to show is that if we choose different coset representatives, we get the same result. (See, for example, (7.2.11) and (7.2.12).)

So suppose $r, r' \in R$ are representatives for the same coset, as are $s, s' \in R$. By Theorem 7.2.4, we see that $r' = r+a$ and $s' = s+b$ for some $a, b \in I$. Therefore, applying (7.2.8), we see that

$$(r'+I)+(s'+I) = (r'+s')+I = (r+s+(a+b))+I. \tag{7.2.13}$$

However, since I is closed under addition, $a+b \in I$, and so

$$(r'+I)+(s'+I) = (r+s+(a+b))+I = (r+s)+I = (r+I)+(s+I). \tag{7.2.14}$$

It follows that the definition of addition in R/I is independent of choices of coset representatives, and therefore, well-defined (unambiguous).

Similarly, applying (7.2.9), we see that

$$(r'+I)(s'+I) = (r's')+I = ((r+a)(s+b))+I = (rs+rb+sa+ab)+I. \tag{7.2.15}$$

Here is precisely the moment that we use the fact that I is not just closed under multiplication by other elements of I, but also by *arbitrary elements of* R (Definition 7.1.1). Because of that property, rb, sa, and ab are all elements of I, and because I is closed under addition, so is $rb+sa+ab$. It follows that

$$(r'+I)(s'+I) = (rs+rb+sa+ab)+I = rs+I = (r+I)(s+I), \tag{7.2.16}$$

and therefore, that multiplication in R/I is well-defined.

It remains to check that the axioms of a ring (Definition 4.2.2) all hold. However, this essentially follows directly from the fact that the ring axioms hold in R. For example, for arbitrary elements $r+I, s+I, t+I \in R/I$, we check associativity of multiplication as follows:

$$\begin{aligned} ((r+I)(s+I))(t+I) &= (rs+I)(t+I) \\ &= ((rs)t+I) \\ &= (r(st)+I) \\ &= (r+I)(st+I) \\ &= (r+I)((s+I)(t+I)), \end{aligned} \tag{7.2.17}$$

where the third equality follows from associtivity of multiplication in R, and the others follow from the definition of multiplication in R/I. The other axioms can be checked similarly (Problem 7.2.4), and the theorem follows. □

Whew! That was a lot of definitions for what may seem like not much reward: In terms of the examples we have seen so far, all we have done is confirm that Section 3.1 is not somehow secretly inconsistent. However, in keeping with our fundamental dogma:

$$\text{Abstraction} \Rightarrow \text{Simplification} \Rightarrow \text{Generalization} \Rightarrow \text{Power}$$

Let me point out that all of the pure abstraction above is in the end probably less work than directly proving that the operations in $\mathbf{Z}/(m)$ are well-defined, without using ideals. Furthermore, as we'll see momentarily, the full abstract definition of R/I allows us to consider new examples of quotient rings, most notably, *finite fields*.

Problems

7.2.1. In the ring $\mathbf{Z}$, let (4) be the principal ideal generated by 4.

(a) Write out the elements of (4). (There are infinitely many, but they fall into a pattern you should indicate by $\ldots$.)

(b) Similarly, write out the elements of each of the four cosets of (4).

7.2.2. (*Proves Theorem 7.2.4*) Let R be a ring, let I be an ideal of R, and let $r, s \in R$. Suppose $r \in s+I$. Prove that $r-s \in I$. (Suggestion: What does it mean to say that $r \in s+I$?)

7.2.3. (*Proves Theorem 7.2.4*) Let R be a ring, let I be an ideal of R, and let $r, s \in R$. The goal of this problem is to prove that if $r = s+a$ for some $a \in I$, then $r+I = s+I$ (i.e., each set is contained in the other). Suppose $r = s+a$ for some $a \in I$.

(a) Prove that $r+I \subseteq s+I$ by proving that if $t \in r+I$, then $t \in s+I$. (Suggestion: What properties of the definition of ideal do we use here?)

(b) Prove that $s = r+b$ for some $b \in I$. (Suggestion: What properties of ideals do we use here?)

(c) Prove that $s + I \subseteq r + I$ by proving that if $t \in s + I$, then $t \in r + I$.

7.2.4. (*Proves Theorem 7.2.11*) Let R be a ring and let I be an ideal of R. This problem goes through the formalities of checking that the ring axioms hold for the quotient ring R/I.

(a) What do three arbitrary elements of R/I look like? (Suggestion: See the end of the proof of Theorem 7.2.11.)

(b) Prove that addition is associative in R/I.

(c) Prove that I is the additive identity in R/I.

(d) Prove that multiplication in R/I is commutative.

(e) Prove that $1 + I$ is the multiplicative identity of R/I.

(f) Prove that the distributive law holds in R/I.

7.3 Computation in $F[x]/(m(x))$

We come to what turns out to be one of the single most useful topics in this book.

Motivating Problem 7.3.1. Let F be a field, let $m(x)$ be a polynomial in $F[x]$, and let $\overline{R} = F[x]/(m(x))$ (the ring of polynomials mod $m(x)$). How do we compute the ring operations of addition and multiplication in $\overline{R}$? How can we tell if $f(x) + (m(x))$ is a unit in $\overline{R}$, and if it is, how do we compute its (multiplicative) inverse?

First, let's review how to solve these same problems in $\mathbf{Z}/(m)$. Let m be a positive integer, and let $I = (m)$.

- By the Division Theorem 2.3.1, every $a \in \mathbf{Z}$ is congruent to exactly one element of $\{0, \dots, m-1\}$ mod I, so by Theorem 7.2.4), I has exactly m cosets in $\mathbf{Z}$, namely, $0 + I, \dots, (m-1) + I$. To avoid having to write $+I$ all the time, we abbreviate $a + I$ as just a, with the understanding that we are working in $\mathbf{Z}/(m)$. To standardize coset representatives, we also call the representative of a given coset $r + I$ that lies in the range $0, \dots, m-1$ the *reduced representative* of $r + I$; again, because of the Division Theorem, this is precisely the remainder obtained when dividing r by m. Note that while it's sometimes helpful to use non-reduced representatives, reduced representatives are perfect for the purpose of figuring out, say, when two elements are equal.

- Definition 7.2.6 tells us that to calculate $a+b$ in $\mathbf{Z}/(m)$, we can calculate $a+b$ in $\mathbf{Z}$ and then use the Division Theorem to find the reduced represenative of $a+b$. We calculate ab similarly. Note that this is precisely the definition of addition and multiplication from Definition 3.1.3.

- Note that inverting a in $\mathbf{Z}/(m)$ is the same as solving $ax = 1$ in $\mathbf{Z}/(m)$, which, by Theorem 7.2.4, is the same as solving $ax + my = 1$ in $\mathbf{Z}$. Corollary 3.2.2 then tells us that a solution is possible exactly when $\gcd(a, m) = 1$, in which case we can invert a by Euclidean Reduction.

In a nutshell, computation in $F[x]/(m(x))$ works exactly the same way! More precisely, for the rest of this section, let F be a field, let $m(x)$ be a polynomial in $F[x]$, let $I = (m(x))$ be the principal ideal generated by $m(x)$ (Definition 7.1.6), and let $\overline{R} = F[x]/m(x)$. We begin by defining reduced representatives for cosets of I in $\overline{R}$.

Theorem 7.3.2. *Every coset $f(x)+I$ has a unique representative $r(x)$ such that $\deg r(x) < \deg m(x)$, computed by finding the remainder $r(x)$ upon dividing $f(x)$ by $m(x)$ (Theorem 3.4.4).*

Proof. Problem 7.3.1. □

Definition 7.3.3. For $f(x) \in F[x]$ we define the *reduced representive of $f(x) + I$* to be the unique representative of $f(x) + I$ such that $\deg r(x) < \deg m(x)$, as per Theorem 7.3.2.

Definition 7.3.4. To avoid having to write $+I$ or mod I all of the time, we abbreviate the coset $x + I$ as $\alpha \in \overline{R}$. Note that by the definition of the ring operations of $\overline{R}$, we can also abbreviate $f(x) + I$ as $f(\alpha)$; in other words, we indicate reducing a polynomial $f(x)$ mod I by replacing x with α. Note also that since $m(x) \in I$, by the definition of quotient ring, $m(\alpha) = 0$. We therefore sometimes describe α as a *root of* $m(x)$.

Notation 7.3.5. Combining Definitions 7.3.3 and 7.3.4, we see that just as we can represent the elements of $\mathbf{Z}/(m)$ as $0, \dots, m-1$, we can represent the elements of $\overline{R} = F[x]/(m(x))$ as polynomials in α of degree up to $(\deg m(x))-1$. Since every element of $\overline{R} = F[x]/(m(x))$ can thereby be written as a polynomial in α, we also sometimes describe $\overline{R}$ as "$F[\alpha]$, where α is a root of $m(x)$." In this context, $F[\alpha]$ is pronounced "F adjoin α."

To summarize, combining Theorem 7.3.2, Definitions 7.3.3 and 7.3.4, and Notation 7.3.5, we see that:

> **The elements of $F[x]/(m(x))$:** Suppose F is a field and $m(x) \in F[x]$ has degree e. If $\overline{R} = F[\alpha]$, where α is a root of $m(x)$, then each element of $\overline{R}$ can be represented *uniquely* as a polynomial of degree $\leq e-1$ in α. In other words, we can take the elements of $F[x]/(m(x)) = F[\alpha]$ to be the set of all polynomials in α of degree at most $e-1$.

In particular, if $F = \mathbf{F}_p$, then $\overline{R}$ has p^e elements, as there are p^e polynomials of degree $\leq e-1$ (see Problem 3.3.5).

Now, as with $\mathbf{Z}/(m)$, there are times when it might be more convenient to use representatives other than the reduced representatives for elements of $F[x]/(m(x))$. However, reduced representatives do at least give definite algorithms for ring operations in $F[x]/(m(x))$, which we can see as follows. Let $\overline{R} = F[\alpha]$, where α is a root of $m(x)$ (Notation 7.3.5), and let $d = \deg m(x)$.

- Since any element of $F[\alpha]$ can be expressed uniquely as a polynomial in α of degree at most $d-1$, and the sum of two such polynomials is a polynomial of degree at most $d-1$, addition in $F[\alpha]$ is just ordinary polynomial addition with coefficients in F.
- For $f(\alpha), g(\alpha) \in F[\alpha]$, to find the product $f(\alpha)g(\alpha)$, we can compute $f(\alpha)g(\alpha)$ as a polynomial, and then reduce it using long division by $m(\alpha)$ (since $m(\alpha) = 0$), with the remainder being the reduced representative of the product.

When computing products in practice, long division is often not the most efficient method, as we can see in the following example.

Example 7.3.6. Let $\overline{R} = \mathbf{F}_2[\alpha]$, where α is a root of $m(x) = x^4 + x + 1$. To compute the product of $\alpha^3 + \alpha + 1$ and $\alpha^3 + \alpha^2$, we first compute:

$$(\alpha^3 + \alpha + 1)(\alpha^3 + \alpha^2) = \alpha^6 + \alpha^5 + \alpha^4 + \alpha^2. \tag{7.3.1}$$

(Check that calculation yourself, remembering that $2\alpha^3 = 0$, since the field of coefficients is $\mathbf{F}_2$.)

However, we're not done yet, as we need to find a reduced representative for this product. Here, instead of doing long division by $\alpha^4+\alpha+1$, it's often better for both hand and machine computation to use the fact that $\alpha^4 + \alpha + 1 = 0$, yielding the following consequences:

$$\begin{aligned} \alpha^4 &= -\alpha - 1 = \alpha + 1 \\ \alpha^5 &= \alpha^2 + \alpha \\ \alpha^6 &= \alpha^3 + \alpha^2. \end{aligned} \tag{7.3.2}$$

The point of those facts is that we can use them to reduce higher powers to lower powers until we get a sum of powers of α less than 4. In any case, our final answer therefore becomes

$$\alpha^6 + \alpha^5 + \alpha^4 + \alpha^2 = (\alpha^3 + \alpha^2) + (\alpha^2 + \alpha) + (\alpha + 1) + \alpha^2 = \alpha^3 + \alpha^2 + 1, \tag{7.3.3}$$

using the fact that $2\alpha^2 = 0$ and $2\alpha = 0$.

As for inverses, they work almost exactly the same as they do in $\mathbf{Z}/(m)$, in that the solution comes as a direct consequence of Bezout's identity (in this case, Corollary 3.6.4).

Corollary 7.3.7. *Let $\overline{R} = F[\alpha]$, where α is a root of $m(x) \in F[x]$. For $b(x) \in F[x]$, the element $b(\alpha) \in \overline{R}$ has an inverse in $\overline{R}$ if and only if* $\gcd(b(x), m(x)) = 1$, *in which case the inverse $g(\alpha)$ of $b(\alpha)$ can be computed by solving*

$$f(x)m(x) + g(x)b(x) = 1 \tag{7.3.4}$$

in $F[x]$, using Euclidean Reduction for polynomials. □

In particular, we come to the following new construction for fields, which is of greatest interest to us when F is finite (especially $F = \mathbf{F}_2$), in which case we get a *finite field.*

Corollary 7.3.8. *Let $\overline{R} = F[x]/(m(x))$. Then $\overline{R}$ is a field if and only if $m(x)$ is irreducible.*

Proof. On the one hand, if $m(x)$ has no factors other than units and associates, then for any $b(x)$ with $\deg b(x) < \deg m(x)$, we have that $\gcd(m(x), b(x)) = 1$, which means that $b(\alpha)$ is a unit, by Corollary 7.3.7. On the other hand, if $m(x) = f(x)g(x)$, Problem 7.3.5 shows that $\overline{R}$ is not an integral domain, and therefore not a field (Theorem 4.2.12). □

We'll return to the topic of finite fields in Section 7.6. For now, we turn to an example of finiding inverses in a quotient ring $F[x]/(m(x))$.

Example 7.3.9. Let $m(x) = x^4 + x + 1 \in \mathbf{F}_2[x]$, and let $\overline{R} = \mathbf{F}_2[\alpha]$, where α is a root of $m(x)$. (In other words, let $\overline{R} = \mathbf{F}_2[x]/(m(x))$.) By Corollary 7.3.7, to look for the (multiplicative) inverse of $\beta = b(\alpha) = \alpha^2 + 1$ in $\overline{R}$, we solve

$$f(x)m(x) + g(x)b(x) = 1 \tag{7.3.5}$$

in $\mathbf{F}_2[x]$. Applying the Euclidean Algorithm, we get

$$\begin{aligned} m(x) &= (x^2+1)b(x) + x \\ b(x) &= (x)(x) + 1. \end{aligned} \tag{7.3.6}$$

So $\gcd(m(x), b(x)) = 1$, which means that β actually has an inverse in $\overline{R}$. To calculate β^{-1}, we use Euclidean rewriting:

$$\begin{aligned} x &= m(x) + (x^2+1)b(x) \\ 1 &= b(x) + (x)(x) \\ &= b(x) + x(m(x) + (x^2+1)b(x)) \\ &= xm(x) + (x^3+x+1)b(x). \end{aligned} \tag{7.3.7}$$

It follows that $\beta^{-1} = \alpha^3 + \alpha + 1$. (But to make sure, go check that $\beta(\alpha^3 + \alpha + 1) = 1$ in $\overline{R}$ yourself!)

Note that we can simplify the bookkeeping in (7.3.7) by working "mod $m(x)$", or in other words, not bothering to keep track of the multiples of $m(x)$ in your rewriting. (This is analogous to what we did in Example 3.2.3 while solving $50x = 2$ in $\mathbf{Z}/(68)$.) That is, if we work mod $m(x)$, (7.3.7) becomes:

$$\begin{aligned} x &= (x^2+1)b(x) \\ 1 &= b(x) + (x)(x) \\ &= b(x) + x(x^2+1)b(x) \\ &= (x^3+x+1)b(x). \end{aligned} \tag{7.3.8}$$

Definitely simpler, don't you think?

We conclude this section with a complete summary of how to compute in $F[x]/(m(x))$.

How to compute in $F[x]/(m(x))$

Let $m(x) = x^e + c_{e-1}x^{e-1} + \cdots + c_1 x + c_0$ be a polynomial of degree e in $F[x]$, and let α be a root of $m(x)$. All elements of $\overline{R} = F[x]/(m(x))$ are expressed as polynomials in α.

Reduction: We can use the equation

$$\alpha^e = -c_{e-1}\alpha^{e-1} - \cdots - c_1\alpha - c_0 \tag{7.3.9}$$

to reduce any polynomial β of degree $\geq e$ in α to a polynomial of degree $< e$ in α, giving the (unique) *reduced form* of β. Note that in our most common frequent setting, where our polynomial coefficients are in $\mathbf{F}_2$, we have that $+1 = -1$ and the minus signs in (7.3.9) can be ignored.

Now suppose β, γ are elements of $\overline{R}$ in reduced form.

Addition: To compute $\beta + \gamma$, use ordinary polynomial addition.

Multiplication: To compute $\beta\gamma$, compute the product $\beta\gamma$ as an ordinary polynomial and then reduce to a polynomial of degree $< e$. (See reduction, above.)

Division: To compute β^{-1}, where $\beta = b(\alpha)$:

1. Use the Euclidean algorithm to compute $\gcd(m(\alpha), b(\alpha))$ as polynomials in α. If the GCD is not 1, then β is not invertible.

2. Otherwise, Euclidean Reduction gives $f(\alpha), g(\alpha)$ such that

   $$f(\alpha)m(\alpha) + g(\alpha)b(\alpha) = 1. \tag{7.3.10}$$

   But then, since $m(\alpha) = 0$, $g(\alpha)b(\alpha) = 1$, or in other words, $\beta^{-1} = g(\alpha)$. Note that for efficiency, you can apply $m(\alpha) = 0$ from the very beginning of Euclidean Reduction.

Field vs. zero divisors: By Corollary 7.3.8, if $m(x)$ is irreducible, then $\overline{R}$ is a field; and if $m(x)$ is reducible, then for any nonconstant divisor $b(x)$ of $m(x)$, $b(\alpha)$ is a zero divisor in $\overline{R}$.

Problems

7.3.1. (*Proves Theorem 7.3.2*) Let F be a field, and for a nonzero $m(x) \in F[x]$, let $I = (m(x))$ (the principal ideal generated by $m(x)$, see Definition 7.1.6). Suppose $f(x) \in F[x]$.

(a) Prove that the coset $f(x) + I$ contains some $r(x)$ such that $\deg(r(x)) < \deg(m(x))$. (Suggestion: What is the analogous statement for the ideal (m) of $\mathbf{Z}$?)

(b) Prove that if $r_1(x), r_2(x) \in f(x) + I$ and both $r_1(x)$ and $r_2(x)$ have degree strictly less than $\deg(m(x))$, then $r_1(x) = r_2(x)$. (Suggestion: What can you say about $r_2(x) - r_1(x)$?)

7.3.2. Find reduced representatives for the following products $\beta\gamma$ in $\overline{R} = \mathbf{F}_2[\alpha]$, where α is a root of a particular $m(x) \in \mathbf{F}_2[x]$. (In other words, $\overline{R} = \mathbf{F}_2[x]/(m(x))$.)

(a) $m(x) = x^4 + 1$, $\beta = \alpha^3 + \alpha^2$, $\gamma = \alpha^2 + \alpha + 1$.

(b) $m(x) = x^4 + x^3 + 1$, $\beta = \alpha^3 + 1$, $\gamma = \alpha^2 + \alpha + 1$.

(c) $m(x) = x^5 + x + 1$, $\beta = \alpha^4 + \alpha + 1$, $\gamma = \alpha^3 + \alpha + 1$.

(d) $m(x) = x^5 + x^3 + x^2 + 1$, $\beta = \alpha^4 + \alpha^2 + \alpha + 1$, $\gamma = \alpha^4 + \alpha^3 + \alpha^2$.

7.3.3. For each of the following $m(x) \in \mathbf{F}_2[x]$, let $\overline{R} = \mathbf{F}_2[\alpha]$, where α is a root of $m(x)$. (In other words, $\overline{R} = \mathbf{F}_2[x]/(m(x))$.) For each $\beta \in \overline{R}$, either find the (multiplicative) inverse of β in $\overline{R}$ or explain why no such inverse exists.

(a) $m(x) = x^4 + 1$, $\beta = \alpha^3 + \alpha^2$.

(b) $m(x) = x^4 + x^3 + 1$, $\beta = \alpha^3 + 1$.

(c) $m(x) = x^5 + x^2 + 1$, $\beta = \alpha^4 + \alpha + 1$.

(d) $m(x) = x^5 + x^3 + x^2 + 1$, $\beta = \alpha^4 + \alpha^3 + \alpha^2 + \alpha + 1$.

7.3.4. The goal of this problem is to compare and contrast two different rings $\overline{R}_1$ and $\overline{R}_2$.

(a) Suppose $\overline{R} = \mathbf{F}_2[x]/(m(x))$, where $m(x)$ is a polynomial of degree 2 in $\mathbf{F}_2[x]$. How many elements does $\overline{R}$ have? Briefly explain. (See Theorem 7.3.2.)

(b) Now let $\overline{R}_1 = \mathbf{F}_2[\alpha_1]$, where α_1 is a root of $x^2 + 1$, and let $\overline{R}_2 = \mathbf{F}_2[\alpha_2]$, where α_2 is a root of $x^2 + x + 1$. Make multiplication tables for each of $\overline{R}_1$ and $\overline{R}_2$, using reduced representations for all elements.

(c) For $i = 1, 2$, answer the following questions: Does $\overline{R}_i$ have zero divisors (Definition 4.2.6)? How many units are there in $\overline{R}_i$?

7.3.5. Prove that if F is a field, $m(x), f(x), g(x) \in F[x]$, $m(x) = f(x)g(x)$, $\deg(f(x)) < \deg(m(x))$, and $\deg(g(x)) < \deg(m(x))$, then $\overline{R} = F[x]/(m(x))$ has zero divisors (Definition 4.2.6). (For notation, let $\alpha \in \overline{R}$ be a root of $m(x)$, as usual.)

7.4 Principal ideal domains

Returning to abstraction, you may have noticed that the only examples of ideals of $F[x]$ that we have considered are principal ideals, that is, the set of all $F[x]$-multiples of some $m(x) \in F[x]$. There's a good reason for that: We'll momentarily show that *every* ideal of $F[x]$ is a principal. This turns out to be such a useful property that we give it a name.

Definition 7.4.1. To say that a ring R is a *principal ideal domain*, or *PID*, means that R is an integral domain (Definition 4.2.6) and that every ideal of R is principal. In other words, the second condition says that if I is an ideal of R, then $I = (a)$ (the set of all R-multiples of a) for some $a \in I$.

As you might hope, or at least suspect, our favorite rings are all PIDs.

Theorem 7.4.2. *Let R be either $\mathbf{Z}$ or $F[x]$ (F a field), or more generally, let R be a Euclidean domain. Then R is a PID.*

Proof. For the case $R = \mathbf{Z}$, let I be an ideal of R. If $I = \{0\}$, then $I = (0)$, and I is principal. Otherwise, I contains nonzero elements of $\mathbf{Z}$; in fact, since I is closed under multiplication by $-1 \in \mathbf{Z}$, I must contain positive elements.

So let d be the smallest positive element of I. For any other $a \in I$, by the Division Algorithm, we have that

$$a = qd + r \qquad\qquad \text{where } 0 \leq r < d. \tag{7.4.1}$$

However, since $-q \in R$, $d \in I$, and I is an ideal, $-qd \in I$; and since $a \in I$, $-qd \in I$, and I is an ideal, $a - qd \in I$. Therefore, $r = a - qd \in I$, and since d is the smallest positive element of I, we must have $r = 0$. It follows that $I = (d)$.

For a field F, the case $R = F[x]$ follows entirely analogously; see Problem 7.4.1. In fact, more or less the same proof works in any Euclidean domain; see Problem 7.4.2. □

Alternate proof. For the case $R = \mathbf{Z}$, let I be an ideal of R. If $I = \{0\}$, then $I = (0)$, and I is principal; otherwise, I contains nonzero elements of $\mathbf{Z}$.

In that case, let d be a nonzero element of I where $|d|$ is as small as possible among nonzero elements of I. For any other $a \in I$, by the Signed Division Theorem 2.3.4, we have that, for some $q, r \in \mathbf{Z}$,

$$a = dq + r \qquad\qquad \text{with } |r| \leq \frac{|d|}{2}. \tag{7.4.2}$$

However, since $-q \in R$, $d \in I$, and I is an ideal, $-qd \in I$; and since $a \in I$, $-qd \in I$, and I is an ideal, $a - qd \in I$. Therefore, $r = a - qd \in I$, and since d is a smallest nonzero element of I and r is a smaller element of I, we must have $r = 0$. It follows that $I = (d)$. □

The fact that every ideal in $F[x]$ is principal leads to an important definition (Definition 7.4.4), for which the following theorem provides some useful context.

Theorem 7.4.3. *Suppose R is a domain (Definition 4.2.6), and suppose that for $a, b \in R$, we have $(a) = (b)$ (i.e., the two elements a and b generate the same principal ideal). Then a and b are associates (Definition 4.3.3).*

Proof. Problem 7.4.3. □

Definition 7.4.4. Let F be a field, and let I be an ideal of $F[x]$. To say that $d(x)$ is the *minimal polynomial* of I means that $I = (d(x))$. Note that if we choose the leading term of $d(x)$ to be 1, then $d(x)$ really is unique and not just "unique up to associates."

Now, the proof of Theorem 7.4.2 is *nonconstructive*, in that it does not describe a method for computing the minimal polynomial $d(x)$ in practice, partly because we have not yet specified the data structure that defines I (e.g., some kind of list of polynomials). However, if, for example, we begin with a finite set of generators for I, we can compute $d(x)$, and you'll never guess what algorithm we use to do that. (OK, at this point in the book, maybe you can guess.)

Theorem 7.4.5. *Let F be a field, and consider the ideal $I = (a(x), b(x))$ of $F[x]$, where $a(x)$ and $b(x)$ are nonzero polynomials in $F[x]$. Then the minimal polynomial of I is $\gcd(a(x), b(x))$, which can be computed by the Euclidean algorithm.*

Proof. Let $d(x) = \gcd(a(x), b(x))$. On the one hand, since $d(x)$ divides both $a(x)$ and $b(x)$, $d(x)$ divides every element of $I = \{r(x)a(x) + s(x)b(x) \mid r(x), s(x) \in F[x]\}$, which means that I is contained in $(d(x))$. Conversely, by Bézout's identity, we know that

$$d(x) = r(x)a(x) + s(x)b(x) \tag{7.4.3}$$

for some $r(x), s(x) \in F[x]$, which means that $d(x)$ and all of its $F[x]$-multiples are contained in I; in other words, $(d(x))$ is contained in I. It follows that $I = (d(x))$, and the theorem follows. □

To conclude this section, we can now state the next step in proving unique factorization in our favorite rings $\mathbf{Z}$ and $F[x]$. (Theorem 4.3.11).

Theorem 7.4.6. *If R is a PID, then unique factorization holds in R, in the sense of Theorem 4.3.11.*

Again, while this kind of theoretical result is important, it's not the main focus of this book, so if you're interested in how that proof goes, see Section A.2.

Problems

7.4.1. (*Proves Theorem 7.4.2*) Let F be a field. The goal of this problem is to prove that every ideal of $F[x]$ is principal. To start, suppose that I is a nonzero ideal of $F[x]$.

(a) The first step is to find a "smallest" possible nonzero element of I. What does "smallest" mean here?

(b) Now suppose that $d(x)$ is a smallest possible nonzero element of I. Imitating the $\mathbf{Z}$ case of the proof of Theorem 7.4.2, prove that any $a(x) \in I$ is a multiple of $d(x)$.

7.4.2. (*Proves Theorem 7.4.2*) Let R be a Euclidean domain with size function σ (Definition 4.3.7). The goal of this problem is to prove that every ideal of R is principal. To start, suppose that I is a nonzero ideal of R.

(a) The first step is to find a "smallest" possible nonzero element of I. What does "smallest" mean here?

(b) Now suppose that d is a smallest possible nonzero element of I. Imitating the $\mathbf{Z}$ case of the proof of Theorem 7.4.2, prove that any $a \in I$ is a multiple of d.

7.4.3. (*Proves Theorem 7.4.3*) Suppose R is a domain and $a, b \in R$.

(a) Prove that if $b \in (a)$ (the principal ideal generated by a) then a divides b.

(b) Prove that if $(a) = (b)$ (i.e., the two principal ideas are the same set), then a and b are associates (Definition 4.3.3). (Suggestion: Use part ((a)) and cancellation (Theorem 4.2.8).)

7.4.4. Consider the ideal $I = (x^3 + 2x + 2, x^4 + 2x^3 + x^2 + x + 3)$ in $\mathbf{F}_5[x]$. Compute the minimal polynomial of I.

7.4.5. (Compute minimal polynomial for ideal generated by three.)

7.5 Homomorphisms

Before we can get to the big finish of this chapter, we need to establish a few more ideas centered around the main idea of a *homomorphism*. To be honest, the things you really need from this section are:

1. The idea of isomorphism and what it means for two rings to be isomorphic (Definition 7.5.9); and
2. The definition of automorphism (Defnition 7.5.13).

However, the more general idea of a homomorphism (Definition 7.5.1) doesn't require much more work, and is so fundamental, that I can't resist the temptation to include the fuller picture. So please bear with me while I indulge my theoretical side for a bit here.

Anyway, without further ado, here's the main definition.

Definition 7.5.1. Let R and R' be rings. To say that a function $\varphi : R \to R'$ is a *homomorphism* means that for all $r, s \in R$,

$$\varphi(r + s) = \varphi(r) + \varphi(s), \qquad \varphi(rs) = \varphi(r)\varphi(s). \tag{7.5.1}$$

In other words, a homomorphism is a function between rings that preserves addition and multiplication.

Example 7.5.2. Let R be a ring, and let I be an ideal of R. Define a function $\varphi : R \to R/I$ by declaring

$$\varphi(r) = r + I \tag{7.5.2}$$

for all $r \in R$. Then φ is a homomorphism called the *canonical homomorphism* from R to R/I. Note that in this case, the two conditions of Definition 7.5.1 become

$$(r + s) + I = (r + I) + (s + I) \qquad (rs) + I = (r + I)(s + I), \tag{7.5.3}$$

and those equalities hold by the definition of R/I (Definition 7.2.6). In fact, we can think of the definition of R/I as being chosen to ensure that the canonical homomorphism really is a homomorphism.

Example 7.5.3. Let F be a field, and fix some $\alpha \in F$. We define a function $\varphi : F[x] \to F$ by declaring

$$\varphi(f(x)) = f(\alpha) \tag{7.5.4}$$

for all $f(x) \in F[x]$. Then φ turns out to be a type of homomorphism known as a *substitution homomorphism*.

The proof that φ is a homomorphism is not deep, and boils down to checking that, for all $f(x), g(x) \in F[x]$, whether you compute $f(\alpha) + g(\alpha)$ and $f(\alpha)g(\alpha)$ by multiplying the polynomials first and then substituting, or by substituting first and then multiplying. The notation is annoying, though, so we'll just discuss the basic idea for the easier verification (checking $f(\alpha) + g(\alpha)$). The point is that the sum $f(x) + g(x)$ in $F[x]$ is defined by

$$\begin{array}{rrrr} & a_nx^n + \cdots + & a_1x + & a_0 \\ + & b_nx^n + \cdots + & b_1x + & b_0 \\ \hline & (a_n + b_n)x^n + \cdots + & (a_1 + b_1)x + & (a_0 + b_0). \end{array} \tag{7.5.5}$$

However, because of the commutativity and associativity of addition in F and the distributive law in F, we also have that

$$\begin{array}{rrrr} & a_n\alpha^n + \cdots + & a_1\alpha + & a_0 \\ + & b_n\alpha^n + \cdots + & b_1\alpha + & b_0 \\ \hline & (a_n + b_n)\alpha^n + \cdots + & (a_1 + b_1)\alpha + & (a_0 + b_0), \end{array} \tag{7.5.6}$$

so adding first and then substituting produces the same result as substituting and then adding. A similar, but more complicated (and did I mention, notationally annoying), argument shows the analogous equality holds for $f(\alpha)g(\alpha)$, and so φ is a homomorphism.

Example 7.5.4. Let R be a ring, and let $\varphi : R \to R$ be an automorphism of R. Define a map $\Phi : R[x] \to R[x]$ by declaring that for $f(x) = a_nx^n + \cdots + a_1x + a_0 \in R[x]$, we define the polynomial $(\Phi(f))(x)$ by

$$(\Phi(f))(x) = \varphi(a_n)x^n + \cdots + \varphi(a_1)x + \varphi(a_0). \tag{7.5.7}$$

In other words, $(\Phi(f))(x)$ is obtained by applying φ to the *coefficients* of $f(x)$. Then Φ is an automorphism of $R[x]$, called the *automorphism of* $R[x]$ *induced by* φ. (The proof is not a big deal but again notationally annoying, so we again omit it.)

Example 7.5.5. Non-examples are sometimes just as illuminating as examples, so here are two non-examples of homomorphisms. Define two maps $\varphi_+ : \mathbf{R} \to \mathbf{R}$ and $\varphi_\times : \mathbf{R} \to \mathbf{R}$ by

$$\varphi_+(x) = 7x, \quad \varphi_\times(x) = x^2. \tag{7.5.8}$$

Now, true to their names, φ_+ preserves addition and $\varphi_\times$ preserves multiplication, because

$$\varphi_+(x+y) = 7(x+y) = 7x+7y = \varphi_+(x)+\varphi_+(y) \tag{7.5.9}$$

and

$$\varphi_\times(xy) = (xy)^2 = x^2y^2 = \varphi_\times(x)\varphi_\times(y). \tag{7.5.10}$$

However, φ_+ isn't a homomorphism because it doesn't preserve multiplication. That is, if we compute both $\varphi_+(xy)$ and $\varphi_+(x)\varphi_+(y)$, we get

$$\varphi_+(xy) = 7xy, \qquad \varphi_+(x)\varphi_+(y) = (7x)(7y) = 49xy. \tag{7.5.11}$$

Those expressions certainly don't look equal! But to be truly careful, we should find specific numbers where Definition 7.5.1 fails. For example, taking $x = 2$ and $y = 3$, we get

$$\varphi_+(2\cdot 3) = \varphi_+(6) = 42, \qquad \varphi_+(2)\varphi_+(3) = (14)(21) = 294, \tag{7.5.12}$$

which is more conclusive. Similarly, if we compute $\varphi_\times(x+y)$ and $\varphi_\times(x)+\varphi_\times(y)$, we get

$$\varphi_\times(x+y) = (x+y)^2 = x^2+2xy+y^2, \qquad \varphi_\times(x)+\varphi_\times(y) = x^2+y^2. \tag{7.5.13}$$

Doesn't look promising, but again, to be careful, taking $x = -1$ and $y = 4$, we get

$$\varphi_\times(-1+4) = \varphi_\times(3) = 9, \qquad \varphi_\times(-1)+\varphi_\times(4) = 1+16 = 17, \tag{7.5.14}$$

and indeed, $\varphi_\times$ doesn't preserve addition.

Remark 7.5.6. In Example 7.5.5, it may seem excessive that you have to try actual numbers to make sure that φ_+ and $\varphi_\times$ aren't homomorphisms — aren't (7.5.11) and (7.5.13) enough? However, the fact is that in certain rings, the maps defined by φ_+ and $\varphi_\times$ actually *are* homomorphisms, even though (7.5.11) and (7.5.13) still hold! See Problem 7.5.3 and Theorem 8.4.7 for details.

Returning to abstraction, the definition of homomorphism implies the following properties in a straightforward way.

Theorem 7.5.7. *Let R and R' be rings, with additive identity elements 0 and $0'$, and let $\varphi : R \to R'$ be a homomorphism. Then $\varphi(0) = 0'$, and for all $r \in R$, $\varphi(-r) = -\varphi(r)$.*

That is to say, homomorphisms must also preserve the additive identity element and negatives (subtraction).

Proof. Problems 7.5.1 and 7.5.2. □

We also have the following less straightforward fundamental property of homomorphisms.

Theorem 7.5.8. *Let R, R', and R'' be rings, and let $\varphi : R \to R'$ and $\rho : R' \to R''$ be homomorphisms. Then $(\rho\circ\varphi) : R \to R''$ is a homomorphism.*

Proof. Problem 7.5.4. □

Our next big idea (Definition 7.5.9) lets us express the idea of two rings being "abstractly the same." (You may first want to take a quick look at functions in the abstract (Appendix **??**), especially if you've never seen that material before; basically you need to know that bijections are one-to-one correspondences, or in other words, one-to-one and onto maps, and that a function is a bijection if and only if it has an inverse.)

Definition 7.5.9. An *isomorphism* is a bijective (one-to-one and onto) homomorphism. To say that rings R and R' are *isomorphic* means that there exists some isomorphism $\varphi : R \to R'$.

The point of the existence of an isomorphism $\varphi : R \to R'$ is that, because φ is a bijective correspondence and preserves the ring operations, R and R' are fundamentally the same algebraic object, just with different names for their elements.

One nonobvious consequence of Definition 7.5.9 is that if $\varphi : R \to R'$ is an isomorphism, then the inverse function $\varphi^{-1} : R' \to R$ not only exists, by Theorem 1.3.14, but is also itself a homomorphism:

Theorem 7.5.10. *If $\varphi : R \to R'$ is an isomorphism, then the inverse function $\varphi^{-1} : R' \to R$ is also a homomorphism (and therefore, an isomorphism, since φ^{-1} is also bijective).*

Proof. Problem 7.5.5. □

Remark 7.5.11. It should be said that the main point of isomorphisms is not just the formal defintion, but the underlying idea. The point is, if rings R and R' are isomorphic, they have exactly the same identifying features in terms of any algebraic properties that can be stated abstractly. To give a few examples:

If two rings R and R' are isomorphic, then:

- R and R' have the same number of units.
- R is an integral domain if and only if R' is an integral domain.
- R is a field if and only if R' is a field.
- R is a PID if and only if R' is a PID.

Note that the converse of the above statement is *not* true; that is, it is possible that two rings share all of the same characteristics listed above (same number of units, etc.) but are not isomorphic.

Remark 7.5.12. When two rings R and R' are isomorphic, they are so much the same that an unfussy algebraist might say that R "is" R'. This is the sense in which we'll be able to describe (without proof, for now) "all" finite fields in Section 7.6.

Finally, we'll need the following idea at a critical moment in the next chapter.

Definition 7.5.13. An *automorphism* is an isomorphism $\varphi : R \to R$ from a ring to itself.

Before you object that "Yeah, I already know that R is the same ring as R," the point of automorphisms of R is that they represent symmetries of R, as illustrated by the following example.

Example 7.5.14. Let $\varphi : \mathbf{C} \to \mathbf{C}$ be the operation of complex conjugation, or in other words, let

$$\varphi(a + bi) = a - bi \tag{7.5.15}$$

for $a, b \in \mathbf{R}$. Then φ is a homomorphism (Problem 7.5.6) and φ inverts itself (i.e., $\varphi(\varphi(z)) = z$ for all $z \in \mathbf{C}$), so φ is an isomorphism, and therefore, an automorphism of $\mathbf{C}$.

The following theorem gives an application of automorphisms that generalizes something may remember from high school algebra (see Example 7.5.16). More immediately, we'll need to use Theorem 7.5.15 in Section 8.4.

Theorem 7.5.15. *Let R be a ring, let $\varphi : R \to R$ be an automorphism of R, and let*

$$f(x) = a_n x^n + \cdots + a_1 x + a_0 \tag{7.5.16}$$

be a polynomial with coefficients in R such that $\varphi(a_i) = a_i$ for $0 \le i \le n$. For $\alpha \in R$, if $f(\alpha) = 0$, then $f(\varphi(\alpha)) = 0$.

Example 7.5.16. Take $R = \mathbf{C}$, let $\varphi : \mathbf{C} \to \mathbf{C}$ be the automorphism of complex conjugation (Example 7.5.14), and suppose $f(x) = a_n x^n + \cdots + a_1 x + a_0$ is a polynomial with real-valued coefficients. In that case, since $\varphi(a_i) = a_i$ for $0 \le i \le n$, Theorem 7.5.15 says that if $a+bi \in \mathbf{C}$ is a zero of $f(x)$ (i.e., $f(a+bi) = 0$), then its complex conjugate $\varphi(a+bi) = a-bi$ is also a zero of $f(x)$ (i.e., $f(a-bi) = 0$). (You may have seen this result before, expressed as something like "The nonreal zeros of a real polynomial come in complex conjugate pairs.")

Proof. Suppose $f(x) = a_n x^n + \cdots + a_1 x + a_0$ and $f(\alpha) = 0$. Applying φ to both sides of $f(\alpha) = 0$, we get

$$\begin{aligned} \varphi(a_n\alpha^n + \cdots + a_1\alpha + a_0) &= \varphi(0) \\ \varphi(a_n\alpha^n) + \cdots + \varphi(a_1\alpha) + \varphi(a_0) &= \varphi(0) \qquad (*) \\ \varphi(a_n)\varphi(\alpha)^n + \cdots + \varphi(a_1)\varphi(\alpha) + \varphi(a_0) &= \varphi(0) \qquad (**) \end{aligned} \tag{7.5.17}$$

where (*) follows because homomorphisms preserve addition and (**) follows because homomorphisms preserve multiplication. Therefore, since $\varphi(a_i) = a_i$ for $0 \le i \le n$ and homomorphisms preserve 0, we have that

$$f(\varphi(\alpha)) = a_n\varphi(\alpha)^n + \cdots + a_1\varphi(\alpha) + a_0 = 0. \tag{7.5.18}$$

The theorem follows. □

Definition 7.5.17. kernel

Example 7.5.18. substitution kernel

Theorem 7.5.19. *first isomorphism theorem.*

Proof. Problem 7.5.12. □

Problems

7.5.1. (*Proves Theorem 7.5.7*) Let R and R' be rings, with additive identity elements 0 and $0'$, and let $\varphi : R \to R'$ be a homomorphism. Prove that $\varphi(0) = 0'$. (Suggestion: Consider $\varphi(0+0)$.)

7.5.2. (*Proves Theorem 7.5.7*) Let R and R' be rings, with additive identity elements 0 and $0'$, and let $\varphi : R \to R'$ be a homomorphism. Prove that if $r \in R$, then $\varphi(-r) = -\varphi(r)$. (Suggestion: Consider $\varphi(r) + \varphi(-r)$.)

7.5.3. Define $\varphi : \mathbf{Z}/(20) \to \mathbf{Z}/(20)$ by the formula

$$\varphi(x) = 5x. \tag{7.5.19}$$

Prove that φ is a homomorphism.

7.5.4. (*Proves Theorem 7.5.8*) Let R, R', and R'' be rings, and let $\varphi : R \to R'$ and $\rho : R' \to R''$ be homomorphisms. The goal of this problem is to show that the composite function $(\rho \circ \varphi) : R \to R''$ is also a homomorphism.

(a) Prove that if $r, s \in R$, then $(\rho \circ \varphi)(r + s) = (\rho \circ \varphi)(r) + (\rho \circ \varphi)(s)$.

(b) Prove that if $r, s \in R$, then $(\rho \circ \varphi)(rs) = (\rho \circ \varphi)(r)(\rho \circ \varphi)(s)$.

7.5.5. (*Proves Theorem 7.5.10*) inverse of isom is isom

7.5.6. Let $\varphi : \mathbf{C} \to \mathbf{C}$ be the operation of complex conjugation, or in other words, let

$$\varphi(a + bi) = a - bi \tag{7.5.20}$$

for $a, b \in \mathbf{R}$. Prove that φ is a homomorphism. (Suggestion: Use the definition of homomorphism.)

7.5.7. Let F be a field in which $1 + 1 = 2 = 0$. (One example of such a field is $F = \mathbf{F}_2$, but it turns out that there are many other examples.) Define $\varphi : F \to F$ by

$$\varphi(x) = x^2 \tag{7.5.21}$$

for $x \in F$.

(a) Prove that for $x, y \in F$, $(x + y)^2 = x^2 + y^2$.

(b) Prove that φ is a homomorphism. (Suggestion: Use the definition of homomorphism.)

7.5.8. Continuing Problem 7.3.4, prove that the rings $\overline{R}_1$ and $\overline{R}_2$ are *not* isomorphic.

7.5.9. Let F be a field of order 8 (i.e., F is a field, and F has 8 elements). Prove that F is *not* isomorphic to $\mathbf{Z}/(8)$. (Suggestion: What abstract features are there in $\mathbf{Z}/(8)$ that cannot be present in a field?)

7.5.10. Let F be a field of order 9 (i.e., F is a field, and F has 9 elements). Prove that F is *not* isomorphic to $\mathbf{Z}/(9)$. (Suggestion: What abstract features are there in $\mathbf{Z}/(9)$ that cannot be present in a field?)

7.5.11. For which $m > 1$ is $\mathbf{Z}/(m)$ a field? Try to prove as much as you can. (Suggestions: First find examples of m for which $\mathbf{Z}/(m)$ *is* a field; cite specific results that justify your answer. Also, compare Problems 7.5.9 and 7.5.10 and look for a pattern.)

7.5.12. (*Proves Theorem 7.5.19*) Proof of first isomorphism theorem.

7.6 Finite fields

This chapter has been building towards describing the key facts about finite fields (Theorems 7.6.5, 7.6.9, 7.6.17, and 7.6.18). When the proofs are short, we give them here; when the proofs will be short later, we delay them until later; and when the proofs are substantial, we just state the theorems without proof. In any case, the really important thing is that you understand the *statements* of the theorems.

We begin with, what else, more definitions.

Definition 7.6.1. The *order* of a field F is defined to be the number of elements in F; a *finite field* is therefore the same as a field of finite order. More generally, the *order* of any algebraic object is the number of elements it contains.

Definition 7.6.2. Let R be a ring. Since R has a multiplicative identity 1, for a positive integer n, we can abbreviate

$$n \cdot 1 = \underbrace{1 + \cdots + 1}_{n \text{ times}}. \tag{7.6.1}$$

Then one of two things must happen. Either:

1. $n \cdot 1 = 0$ for some positive integer n; or
2. $n \cdot 1 \neq 0$ for all positive integers n.

In case (1), we define $\mathrm{char}(R)$, the *characteristic* of R to be the smallest positive integer n such that $n \cdot 1 = 0$; and in case (2), we define $\mathrm{char}(R) = 0$.

Example 7.6.3. For $R = \mathbf{Z}/(m)$, $m \cdot 1 = 0$, and for $0 < n < m$, $n \cdot 1 \neq 0$, so $\mathrm{char}(R) = m$.

Example 7.6.4. By reasoning similar to that of Example 7.6.3, $\mathbf{F}_p[x]$ has characteristic p, and by the same argument again, if $m(x) \in F_p[x]$ has $\deg(m(x)) \geq 1$, then $\mathbf{F}_p[x]/(m(x))$ also has characteristic p. This is important because, for example, if $R = \mathbf{F}_p[\alpha]$, where α is a root of $m(x)$, then it is still the case that $\mathrm{char}(R) = p$, which means that it is still true that $p = 0$ in R, even though R contains $p^{\deg(m(x))}$ elements.

Returning to the matter of finite fields, our first point is:

Theorem 7.6.5. *Let F be a finite field. Then* $\mathrm{char}(F) = p$ *for some prime p.*

We use the notation of Definition 7.6.2, especially (7.6.1), throughout.

Proof. First, since F is finite, by the pigeonhole principle, there must be some integers $0 < k < n$ such that $k \cdot 1 = n \cdot 1$. Cancelling additively, we see that $(n-k)\cdot 1 = 0$, and so $\operatorname{char}(F) \neq 0$.

So suppose F has nonprime characteristic, or in other words, suppose $\operatorname{char}(F) = n = ab$ for some positive integers a, b, n with $1 < a, b < n$. Then $a \cdot 1 \neq 0$ and $b \cdot 1 \neq 0$, but repeated application of the distributive law yields

$$(a \cdot 1)(b \cdot 1) = n \cdot 1 = 0. \tag{7.6.2}$$

But a field cannot contain zero divisors (Theorem 4.2.12); contradiction. It follows that $\operatorname{char}(F)$ must be prime. □

The point of Theorem 7.6.5 is that if F is a finite field, then F has a copy of $\mathbf{Z}/(p)$, or really, $\mathbf{F}_p$, sitting inside it. As we'll see, we can think of this copy of of $\mathbf{F}_p$ as a base on which F can be constructed.

Our next key fact about finite fields has to do with their multiplicative structure. As usual, we begin with a definition.

Definition 7.6.6. Let F be a field. We use $F^\times$ to denote the set of all nonzero elements of F, all of which are units (since F is a field). We therefore call $F^\times$ the *multiplicative group* of F.

You may have noticed the appearance of a new word, *group*, in Definition 7.6.6. In Chapter 10, we'll give an axiomatic definition of groups, much as we did earlier for rings and fields. For now, however, you can just think of $F^\times$ as a set, and worry about any other properties it might have later.

Definition 7.6.7. Let $F^\times$ be the multiplicative group of the field F, and suppose $\alpha \in F^\times$. We define the *cyclic subgroup generated by* α to be the set

$$\langle \alpha \rangle = \{\alpha^n \mid n \in \mathbf{Z}\}, \tag{7.6.3}$$

or in other words, the set of all powers of α, positive, negative, or zero. (Note that negative powers of α make sense, since α is a unit, and therefore, α^{-1} exists.)

We introduce cyclic subgroups here for use in the following definition.

Definition 7.6.8. Let $F^\times$ be the multiplicative group of the field F. To say that $F^\times$ is *cyclic* means that there exists some $\alpha \in F^\times$ such that $F^\times = \langle \alpha \rangle$, or in other words, such that every element of $F^\times$ is some power of α. If $F^\times = \langle \alpha \rangle$, we say that α is a *primitive* element of F.

We can now state our next fundamental fact about finite fields.

Theorem 7.6.9. *If F is a finite field, then its multiplicative group $F^\times$ is cyclic. In other words, every finite field contains a primitve element.*

See Section B.4 for a proof. Note that even though Theorem 7.6.9 guarantees the existence of a primitive element, it doesn't say how to find such an element; in fact, as of 2023, this is still a wide-open problem (refs?). (For example, recall that back in Section 3.1, we discussed how the question of whether 2 is primitive in $\mathbf{F}_p$ for infinitely many p is still unresolved.)

For now, we introduce a few ideas that make finding primitive elements (slightly) more efficient.

Definition 7.6.10. Let $F^\times$ be the multiplicative group of the field F, and suppose $\alpha \in F^\times$. If $\alpha^n = 1$ for some positive integer n, we define the *order* of α to be the *smallest* possible n such that $\alpha^n = 1$. Otherwise, if $\alpha^n \neq 1$ for all positive integers n, we say that α has *infinite order.*

We have the following facts about the order of an element of $F^\times$. They follow from more general facts proven in Section B.4, so we delay their proofs until then.

Theorem 7.6.11. *Let F be a field of order n, let $F^\times$ be the multiplicative group of F, and suppose $\alpha \in F^\times$. Then:*

1. *(Max order means primitive) The order of α is equal to the order of (number of elements in) $\langle\alpha\rangle$. It follows that α is primitive if and only if the order of α is equal to $n-1$, the order of $F^\times$.*
2. *(Order of a Power Formula) If k is the order of α, then the order of α^m is $\dfrac{k}{\gcd(k,m)}$.*
3. *(Lagrange's Theorem) If k is the order of α, then k divides $n-1$ (the order of $F^\times$).*

Because of Theorem 7.6.11, to find primitive elements in a field F with n elements, we just need to find elements of order $n-1$. In fact, because the order of some $\alpha \in F^\times$ must be a divisor of $n-1$, it suffices to show that $\alpha^d \neq 1$ for all proper divisors of $n-1$, cutting down the computation of powers of α by at least roughly half.

Example 7.6.12. If you think about it, you may find Theorem 7.6.9 surprising even when $F = \mathbf{F}_p$. For example, take $\mathbf{F}_7 = \mathbf{Z}/(7)$. Because $\mathbf{F}_7^\times = \{1, 2, 3, -1, -2, -3\}$ has 6 elements, Lagrange's Theorem (Theorem 7.6.11, part (3)) says that the order of any element of $\mathbf{F}_7^\times$ must divide 6, which means that any element of order > 3 must have order 6, and must therefore be primitive (Theorem 7.6.11, part (1)).

So let's find the orders of each of the elements of $\mathbf{F}_7^\times$. The elements ± 1 are straightforward: $1^1 = 1$, so 1 has order 1, and no other element has order 1; and $(-1)^2 = 1$, so -1 has order 2. The other elements take a little more work:

$$2^2 = -3 \qquad 2^3 = 1 \tag{7.6.4}$$
$$3^2 = 2 \qquad 3^3 = -1 \tag{7.6.5}$$
$$(-2)^2 = -3 \qquad (-2)^3 = -1 \tag{7.6.6}$$
$$(-3)^2 = 2 \qquad (-3)^3 = 1. \tag{7.6.7}$$

We therefore see that 2^3 and $(-3)^3$ are equal to 1, with no smaller power equal to 1, so 2 and (-3) each have order 3. On the other hand, $(-2)^k$ and 3^k are not equal to 1 for $1 \leq k \leq 3$, so their orders are greater than 3; as mentioned above, each of those elements has order 6, and is therefore primitive. So, as predicted by Theorem 7.6.9, $\mathbf{F}_7^\times$ is cyclic, generated by either -2 or 3. (Compare Example 3.1.10.)

Example 7.6.13. Once we find a primitive element, we can also use the Order of a Power Formula (Theorem 7.6.11, part (2)) to find the orders of the elements of $\mathbf{F}_7^\times$. For example, starting with the primitive element 3, we write each element of $\mathbf{F}_7^\times$ as a power of 3:

$$\begin{array}{lll} 3^1 = 3, & 3^2 = 2 & 3^3 = -1 \\ 3^4 = -3 & 3^5 = -2 & 3^6 = 1. \end{array} \tag{7.6.8}$$

Then applying the Order of a Power Formula, we see that:

- The order of $2 = 3^2$ is $\dfrac{6}{\gcd(2,6)} = 3$;
- The order of $-1 = 3^3$ is $\dfrac{6}{\gcd(3,6)} = 2$;
- The order of $-3 = 3^4$ is $\dfrac{6}{\gcd(4,6)} = 3$;

And so on.

Example 7.6.14. For a more complicated example of finding and using a primitive element, let $m(x) = x^4+x^3+x^2+x+1$, which turns out to be irreducible in $\mathbf{F}_2[x]$, and let $F = \mathbf{F}_2[\alpha]$, where α is a root of $m(x)$. Because $m(x)$ has degree 4, by Notation 7.3.5 and the dicussion immediately afterwards, F is a field of order $2^4 = 16$ whose elements are the polynomials in α with coefficients in $\mathbf{F}_2$ and degree 3 or less, with reduced form given by the rule $m(\alpha) = 0$, or in other words, by the reduction rule

$$\alpha^4 = \alpha^3 + \alpha^2 + \alpha + 1 \tag{7.6.9}$$

and related reductions. (See Example 7.3.6.) By Theorem 7.6.9, we know F has a primitive element, so let's try to find that element by trial and error.

Since max order means primitive (Theorem 7.6.11, part 1), we need to find an element of order 15; and since the divisors of 15 are 1, 3, 5, and 15, by Lagrange's Theorem (Theorem 7.6.11, part 3), it's enough to find an element whose order is strictly greater than 5. However, beyond that, short of using some pretty sophisticated math (see ??), there isn't really anything better to do than to consider a random element $\beta \in F^\times$ and taking powers of β until we either find some $\beta^n = 1$ or show that the order of β is greater than 5.

In an attempt to keep things simple, let's try α first. Because α, α^2, and α^3 are all degree ≤ 3, they're all reduced, and therefore not equal to 1; it follows that the order of α is strictly greater than 3. In fact, (7.6.9) gives the reduced form of α^4, so it remains to compute α^5. However, multiplying both sides of (7.6.9) by α and reducing using (7.6.9) gives

$$\alpha^5 = \alpha^4 + \alpha^3 + \alpha^2 + \alpha = (\alpha^3 + \alpha^2 + \alpha + 1) + \alpha^3 + \alpha^2 + \alpha = 1, \tag{7.6.10}$$

so α has order 5, and is therefore not primitive. So we need to try another guess.

Trying $\beta = \alpha + 1$ next, we see that $\beta^2 = \alpha^2 + 1$ and $\beta^3 = \alpha^3 + \alpha^2 + \alpha + 1$ have degree ≤ 3, so they're both reduced (and not equal to 1). Next, we have

$$\beta^4 = \alpha^4 + 1 = (\alpha^3 + \alpha^2 + \alpha + 1) + 1 = \alpha^3 + \alpha^2 + \alpha \neq 1 \tag{7.6.11}$$

and

$$\beta^5 = \beta^4(\alpha + 1) = \alpha^4 + \alpha = \alpha^3 + \alpha^2 + \alpha + 1 + \alpha = \alpha^3 + \alpha^2 + 1 \neq 1. \tag{7.6.12}$$

It follows that the order of β is greater than 5, which, by Lagrange's Theorem, means that the order of β is 15, and $\beta = \alpha + 1$ is primitive.

Note that the Order of a Power Formula applies here as well, so:

- The order of $\beta^2 = \alpha^2 + 1$ is $\dfrac{15}{\gcd(2, 15)} = 15$;
- The order of $\beta^3 = \alpha^3 + \alpha^2 + \alpha + 1$ is $\dfrac{15}{\gcd(3, 15)} = 5$;
- The order of $\beta^4 = \alpha^3 + \alpha^2 + \alpha$ is $\dfrac{15}{\gcd(4, 15)} = 15$;
- The order of $\beta^5 = \alpha^3 + \alpha^2 + 1$ is $\dfrac{15}{\gcd(5, 15)} = 3$;

and so on.

We also have the following important theoretical consequences of Theorem 7.6.11.

Corollary 7.6.15. *Let F be a field of order q. Then every α is a root of the polynomial $x^q - x \in F[x]$, and consequently,*

$$x^q - x = \prod_{\alpha \in F} (x - \alpha). \tag{7.6.13}$$

Note that the (possibly mysterious) stuff on the right-hand side of (7.6.13) should be read as "the product of all possible monomials $(x - \alpha)$, where α ranges over all elements of F." In fact, (7.6.13) is a pretty remarkable statement even for $F = \mathbf{Z}/(p)$; for example, for $F = \mathbf{Z}/(7)$, (7.6.13) says that in $\mathbf{F}_7[x]$,

$$x^7 - x = x(x-1)(x-2)(x-3)(x-4)(x-5)(x-6). \tag{7.6.14}$$

(Try it yourself, probably with the help of a computer.)

Proof. This is certainly true for $\alpha = 0$, so suppose $\alpha \in F^\times$. By Theorem 7.6.11, $\alpha^k = 1$ for some k such that $q - 1 = kd$ (i.e., some k dividing $q - 1$), so

$$\alpha^{q-1} = (\alpha^k)^d = 1^d = 1. \tag{7.6.15}$$

Multiplying both sides by α, we see that $\alpha^q = \alpha$.

As for (7.6.13), let

$$f(x) = \prod_{\alpha \in F} (x - \alpha), \tag{7.6.16}$$

the product of all of the $(x - \alpha)$. By the Factor Theorem (Corollary 3.4.8), $(x - \alpha)$ must divide $x^q - x$ for all $\alpha \in F$, so by Unique Factorization 3.5.12, since the $(x - \alpha)$ are all distinct irreducibles, $f(x)$ must also divide $x^q - x$. However, since $f(x)$ also has degree q, we must have that $x^q - x = cf(x)$, where $c \neq 0$ is a constant polynomial; and since both $x^q - x$ and $f(x)$ have leading coefficient 1, (7.6.13), and the corollary, follow. □

Remark 7.6.16. Two points about order that you may find confusing:

1. The word "order" is used in several different ways here: it's the smallest positive n such that $\alpha^n = 1$, and it's also the number of elements in something. Fortunately those two ideas don't usually conflict (see Theorem 7.6.11), but the word is certainly overused.

2. Since $F^\times$ contains all of the elements of F except 0, if the order of F is n, the order of $F^\times$ is $n - 1$.

In any case, while the first confusion is the fault of mathematicians, and the second confusion is a fact of nature, there's not much we can do about either at this point.

The remaining facts about finite fields are deeper in nature.

Theorem 7.6.17. *Let F be a finite field of characteristic p. Then F is isomorphic to $\mathbf{F}_p[x]/(m(x))$ for some irreducible polynomial $m(x) \in \mathbf{F}_p[x]$.*

See Section B.4 for a proof, or at least most of a proof. Note that as a result, the order of a finite field must be p^e for some prime p and some positive integer e. More surprisingly, something of the reverse is true: There is a unique field of order p^e for every prime p and positive e.

Theorem 7.6.18. *Let p be a prime, and let e be a positive integer.*

1. *There exists at least one field of order p^e.*

2. *If F and K are both finite fields of order p^e, then F and K are isomorphic (Definition 7.5.9).*

While a full proof of Theorem 7.6.18 is outside the scope of this book, see Section B.4 for an idea of the proof.

Definition 7.6.19. Note that since any two fields of order p^e are isomorphic, algebraically we can think of them as being the same; in other words, there is really only *one* field of any given order p^e. For $q = p^e$ (p prime, $e \geq 1$), we may therefore define $\mathbf{F}_q$ to be "the" field of order q. This field is also sometimes known as the *Galois field of order* q, or $GF(q)$ for short. (Compare Definition 3.2.9.)

Remark 7.6.20. While we have said this already, it again bears repeating that if p is prime, $e > 1$, and $q = p^e$, then $\mathbf{F}_q$ is *not* isomorphic to $\mathbf{Z}/(q)$. (See Problem 7.5.9.) Instead, by Theorem 7.6.17, $\mathbf{F}_q = \mathbf{F}_p[x]/(m(x))$ for some irreducible polynomial $m(x) \in \mathbf{F}_p[x]$ such that $\deg m(x) = e$. Note that Theorem 7.6.17 therefore has the highly nonobvious consequence that for any prime p and positive integer e, there exists at least one irreducible polynomial $m(x) \in \mathbf{F}_p[x]$ of degree e.

Well, that was a lot of stuff going on there! So, to summarize the key takeaways about finite fields:

Five Facts for Finite Fields

1. **Prime power:** The characteristic (Definition 7.6.2) of a finite field must be a prime p, and its order must be $q = p^e$ for some $e \geq 1$.

2. **Orders of elements:** The multiplicative group of a finite field (Definition 7.6.6) is cyclic (Definition 7.6.7), or in other words, if F has q elements, $F^\times$ must contain at least one primitive element of order $q-1$. Moreover, every element of $F^\times$ must have order dividing $q-1$, and conversely, for each positive divisor d of $q-1$, $F^\times$ must contain some element of order d.

3. **Magic polynomial:** If F is a field of order q, then every $\alpha \in F$ is a root of $x^q - x$, or in other words, $\alpha^q = \alpha$ for every $\alpha \in F$. Consequently, $x^q - x$ factors as the product of all $(x - \beta)$, where β ranges over all elements of F (including $0 \in F$).

4. **Construction:** Every finite field F of characteristic p is isomorphic to $\mathbf{F}_p[x]/(m(x))$ for some irreducible polynomial $m(x)$. If $\deg m(x) = e$ and α is a root of $m(x)$, then the elements of F are the polynomials in α of degree at most $e - 1$, with multiplication given by the reduction rule $m(\alpha) = 0$.

5. **Classification:** For any prime p and $q = p^e$ ($e \geq 1$), there exists a field $\mathbf{F}_q$ of order q that is unique up to isomorphism.

Example 7.6.21. Let's see what the Five Facts tell us about a hypothetical field F of order $q = 1024$.

1. (Prime power) There exist fields of order 1024 because $1024 = 2^{10}$, i.e., $p - 2$ and $e - 10$. Note that by comparison, there are no fields of order 1023, 1025, or 1026.

2. (Orders of elements) Since $q-1 = 1023$, F must contain some primitive element α, of order 1023. Furthermore, since $1023 = 3 \cdot 11 \cdot 31$, the positive divisors of 1023 are 1; 3, 11, 31; 33, 93, 341; and 1023,* so those are exactly the orders of elements of $F^{\times}$.

3. (Magic polynomial) We have that $\beta^{1024} - \beta$ for every β in F. Therefore, if
$$F = \{0, 1, \beta_3, \dots, \beta_{1024}\}, \tag{7.6.17}$$
then
$$x^{1024} - x = \prod_{\beta \in F} (x - \beta) = x(x-1)(x-\beta_3)\cdots(x-\beta_{1024}). \tag{7.6.18}$$

4. (Construction) F is isomorphic to $\mathbf{F}_2[x]/(m(x))$, where $m(x)$ is some irreducible polynomial of degree 10. For example, we could have $m(x) = x^{10} + x^3 + 1$ or $m(x) = x^{10} + x^4 + x^3 + x^1 + 1$ (two polynomials found by consulting an online reference).

5. (Classification) F is unique up to isomorphism. For example, the two fields
$$\mathbf{F}_2[x]/(x^{10} + x^3 + 1), \qquad \mathbf{F}_2[x]/(x^{10} + x^4 + x^3 + x^1 + 1) \tag{7.6.19}$$
are isomorphic (!!).

As a look back at where we've been and a look ahead at where we'll go, note that the Five Facts draw upon much of what we've seen in Chapter 7 and also draw on many things to come. Specifically:

- **Prime power:** Theorem 7.6.5 plus the Construction fact and Theorem 7.3.2.
- **Orders of elements:** Theorem 7.6.9 and Theorem 7.6.11 (particularly Lagrange's Theorem), proofs in (later? never?).
- **Magic polynomial:** Corollary 7.6.15.
- **Construction:** The theory is found in Theorem 7.6.17, proof in (later? never?). Computational methods for working in $\mathbf{F}_p[x]/(m(x))$ are the main topic of Section 7.3.
- **Classification:** Theorem 7.6.18, proof in (later? never?).

As we said at the beginning of the chapter: All the pieces matter.

*The products of 0; 1; 2; and 3 of the primes 3, 11, 31, respectively.

Problems

7.6.1. Find the order (Definition 7.6.10) of each nonzero element of $\mathbf{F}_{11}$. Which elements are primitive?

7.6.2. Find the order (Definition 7.6.10) of each nonzero element of $\mathbf{F}_{13}$. Which elements are primitive?

7.6.3. It is a fact that $x^2 + 1 \in \mathbf{F}_3[x]$ is irreducible. Let $\mathbf{F}_9 = \mathbf{F}_3[\alpha]$, where α is a root of $x^2 + 1$.

(a) Find a primitive element of $\mathbf{F}_9$ by trial and error.

(b) Use the primitive element you found and Theorem 7.6.11 to find the orders of each nonzero element of $\mathbf{F}_9$, as in Example 7.6.13. Which elements are primitive?

7.6.4. It is a fact that $x^4 + x + 1 \in \mathbf{F}_2[x]$ is irreducible. Let $\mathbf{F}_{16} = \mathbf{F}_2[\alpha]$, where α is a root of $x^4 + x + 1$.

(a) Find a primitive element of $\mathbf{F}_{16}$ by trial and error.

(b) Use the primitive element you found and Theorem 7.6.11 to find the orders of each nonzero element of $\mathbf{F}_{16}$, as in Example 7.6.13. Which elements are primitive?

7.6.5. It is a fact that $x^4 + x^3 + 1 \in \mathbf{F}_2[x]$ is irreducible. Let $\mathbf{F}_{16} = \mathbf{F}_2[\alpha]$, where α is a root of $x^4 + x^3 + 1$.

(a) Find a primitive element of $\mathbf{F}_{16}$ by trial and error.

(b) Use the primitive element you found and Theorem 7.6.11 to find the orders of each nonzero element of $\mathbf{F}_{16}$, as in Example 7.6.13. Which elements are primitive?

7.6.6. Let $m_1(x) = x^4 + x + 1$ and $m_2(x) = x^4 + x^3 + 1$ in $\mathbf{F}_2[x]$. Write down one particular isomorphism between the field $F_1 = \mathbf{F}_2[x]/(m_1(x))$ and the field $F_2 = \mathbf{F}_2[x]/(m_2(x))$.

7.6.7. Use Theorem 7.6.11 to explain, without calculation, why *every* $\alpha \in \mathbf{F}_8^\times$, except $\alpha = 1$, must be primitive. Generalize this idea to $\mathbf{F}_q$ for other $q = 2^e$. (Suggestion: This also works for $q = 32$ and $q = 128$.)

7.6.8. By the Five Facts for Finite Fields, there is exactly one field of order $1024 = 2^{10}$, up to isomorphism. Let $F = \mathbf{F}_{1024}$ be that field.

(a) What is the largest order of any element of the multiplicative group $F^\times$? Explain.

(b) Does $F^\times$ contain an element of order 32? Explain how you know for sure, one way or another.

(c) Does $F^\times$ contain an element of order 31? Explain how you know for sure, one way or another.

7.7 Two worked examples: $\mathbf{F}_8$ and $\mathbf{F}_{16}$

Since a lot just happened in this chapter, it's certainly understandable if you find it challenging to put it all together. To help with that process, this section works out the gory details of two of the smaller examples of finite fields: the fields of order $8 = 2^3$ and $16 = 2^4$. Note that since the field of order 2^e can be constructed as $\mathbf{F}_2[x]/(m(x))$, where $m(x)$ is *any* irreducible polynomial in $\mathbf{F}_2[x]$ of degree e, we start each example by choosing a particular $m(x)$.

7.7.1 The field $\mathbf{F}_8$

Definition. Let $m(x) = x^3 + x + 1$, an irreducible polynomial in $\mathbf{F}_2[x]$. We define $\mathbf{F}_8$ to be $\mathbf{F}_2[x]/(m(x))$, and we let α be a root of $m(x)$ in $\mathbf{F}_8$. (We can also write "$\mathbf{F}_8 = \mathbf{F}_2[\alpha]$, where α is a root of $m(x)$.")

Elements. By Theorem 7.3.2 and Definition 7.3.4, the elements of $\mathbf{F}_8$ can be written as the polynomials of degree < 3 (that is, degree ≤ 2) in α. That is,

$$\mathbf{F}_8 = \left\{0, 1, \alpha, \alpha+1, \alpha^2, \alpha^2+1, \alpha^2+\alpha, \alpha^2+\alpha+1\right\}. \tag{7.7.1}$$

If you're wondering, where did that list come from, here's one way to enumerate the polynomials of degree ≤ 2. First, list all the polynomials of degree ≤ 0, i.e., the constant polynomials:

$$\{0, 1\} \tag{7.7.2}$$

If $f(\alpha)$ is a polynomial of degree ≤ 1, then either $f(\alpha)$ actually has degree ≤ 0, or $f(\alpha)$ has the form $\alpha + g(\alpha)$, where $g(\alpha)$ has degree ≤ 0. So the list of polynomials of degree ≤ 1 is (7.7.2), plus the list you get from adding α to everything in (7.7.2):

$$\{0, 1, \alpha, \alpha+1\} \tag{7.7.3}$$

To get the polynomials of degree ≤ 2, we start with (7.7.3) and then append the list of polynomials you get by adding α^2 to each of the polynomials in (7.7.3), which gives (7.7.1).

Addition. Addition in $\mathbf{F}_8$ is exactly the same as addition in $\mathbf{F}_2[x]$; just keep in mind that $2 = 0$ (i.e., $\mathbf{F}_8$ has characteristic 2). For example,

$$(\alpha^2+1) + (\alpha^2+\alpha+1) = 2\alpha^2 + \alpha + 2 = \alpha. \tag{7.7.4}$$

Reduction table. For multiplication and inverses, we list the consequences of $\alpha^3+\alpha+1 = 0$. That is, since $\alpha^3 = \alpha + 1$ (remember that $+1 = -1$!), we have:

$$\alpha^3 = \alpha + 1, \qquad \alpha^4 = \alpha^2 + \alpha. \tag{7.7.5}$$

Note that these relations allow us to reduce any polynomial of degree 3 or 4 in α to a polynomial of degree ≤ 2 in α.

Multiplication table. We can use (7.7.5) to construct the following multiplication table for $\mathbf{F}_8$. (For typographical reasons, we use the abbreviation $\beta = \alpha^2 + \alpha + 1$.)

$\cdot$	0	1	α	$\alpha+1$	α^2	α^2+1	$\alpha^2+\alpha$	β
0	0	0	0	0	0	0	0	0
1	0	1	α	$\alpha+1$	α^2	α^2+1	$\alpha^2+\alpha$	β
α	0	α	α^2	$\alpha^2+\alpha$	$\alpha+1$	1	β	α^2+1
$\alpha+1$	0	$\alpha+1$	$\alpha^2+\alpha$	α^2+1	β	α^2	1	α
α^2	0	α^2	$\alpha+1$	β	$\alpha^2+\alpha$	α	α^2+1	1
α^2+1	0	α^2+1	1	α^2	α	β	$\alpha+1$	$\alpha^2+\alpha$
$\alpha^2+\alpha$	0	$\alpha^2+\alpha$	β	1	α^2+1	$\alpha+1$	α	α^2
β	0	β	α^2+1	α	1	$\alpha^2+\alpha$	α^2	$\alpha+1$

But don't just look at that table! Choose a few entries at random and work them out yourself. For example,

$$\begin{aligned}(\alpha^2+1)\beta &= \alpha^4+\alpha^3+\alpha^2+\alpha^2+\alpha+1 \\ &= \alpha^4+\alpha^3+\alpha+1 && \text{because } 2\alpha^2 = 0 \\ &= (\alpha^2+\alpha)+(\alpha+1)+\alpha+1 && \text{by (7.7.5)} \\ &= \alpha^2+\alpha && \text{because } 3\alpha=\alpha \text{ and } 2=0.\end{aligned} \tag{7.7.6}$$

Inverses. By Corollary 7.3.7, the inverse of $b(\alpha)$ can be computed by applying Euclidean Reduction to $\gcd(b(x), m(x))$ for $m(x) = x^3 + x + 1$. For example, to find the inverse of $\beta = \alpha^2 + \alpha + 1$, we take $b(x) = x^2 + x + 1$ and compute

$$\begin{aligned}x^3+x+1 &= (x+1)(x^2+x+1)+x \\ x^2+x+1 &= (x+1)x+1.\end{aligned} \tag{7.7.7}$$

Then keeping in mind that $-1 = +1$,

$$\begin{aligned}x &= m(x)+(x+1)b(x) \\ 1 &= b(x)+(x+1)(m(x)+(x+1)b(x)) \\ &= (1+(x+1)^2)b(x)+(x+1)m(x).\end{aligned} \tag{7.7.8}$$

It follows that

$$\beta^{-1} = 1+(\alpha+1)^2 = 1+\alpha^2+1 = \alpha^2. \tag{7.7.9}$$

You should also check that $(\alpha^2+\alpha+1)\alpha^2 = 1$, both directly and in the multiplication table above.

Primitive elements and orders. By Problem 7.6.7, every element of $\mathbf{F}_8$, except 0 and 1, has order 7, and is therefore primitive.

Log-antilog tables. You may have noticed that in a finite field F, multiplication, and even more so division, are kind of complicated! So what do you do in practice if you need to do a lot of computation in F? To be honest, for small examples (e.g., size smaller than a few thousand) you work out the multiplication table ahead of time and hard-code it as a lookup table.[†] But for larger fields, e.g., order 2^{20}, where the lookup table would have around $2^{40}/2$ entries, you can use the following smaller lookup table solution.

Definition 7.7.1. Let F be a finite field and let α be a primitive element of F. The *antilog table of F with respect to α* is a list of the powers of α in reduced form, and the *log table of F with respect to α* is a list of the nonzero elements of F, expressed as powers of α.

Returning to our example, the antilog and log tables of $\mathbf{F}_8$ with respect to α are shown in Table 7.7.1.

Antilog		Log	
α^0	1	1	α^0
α^1	α	α	α^1
α^2	α^2	$\alpha+1$	α^3
α^3	$\alpha+1$	α^2	α^2
α^4	$\alpha^2+\alpha$	α^2+1	α^6
α^5	$\alpha^2+\alpha+1$	$\alpha^2+\alpha$	α^4
α^6	α^2+1	$\alpha^2+\alpha+1$	α^5

Table 7.7.1: Antilog and log tables for $\mathbf{F}_8$

The point of making a table like Table 7.7.1 is that we can use it to reduce multiplication and division in $\mathbf{F}_q$ to arithmetic mod $q-1$ (since the order of primitive element is $q-1$). For example, if we want to multiply α^2+1 and $\alpha^2+\alpha+1$, looking at the log table, we can see that

$$\alpha^2+1=\alpha^6, \qquad \alpha^2+\alpha+1=\alpha^5. \tag{7.7.10}$$

Therefore, $(\alpha^2+1)(\alpha^2+\alpha+1)=\alpha^{11}=\alpha^4$, since α has order 7, and therefore, exponents can all be considered mod 7. From the antilog table, we see that our product is equal to $\alpha^2+\alpha$.

Similarly, if we want to find the inverse of $\alpha+1$, since the log table tells us that $\alpha+1=\alpha^3$, we see that

$$(\alpha+1)^{-1}=\alpha^{-3}=\alpha^4=\alpha^2+\alpha, \tag{7.7.11}$$

where the middle equality comes from the fact that $\alpha^7=1$, and the last equality comes from the antilog table. Again, exponents of α are computed mod 7, since, as previously mentioned, the order of α is 7.

7.7.2 The field $\mathbf{F}_{16}$

Definition. Let $m(x)=x^4+x+1$, an irreducible polynomial in $\mathbf{F}_2[x]$. We define $\mathbf{F}_{16}$ to be $\mathbf{F}_2[x]/(m(x))$, and we let α be a root of $m(x)$ in $\mathbf{F}_{16}$. (We can also write "$\mathbf{F}_{16}=\mathbf{F}_2[\alpha]$, where α is a root of $m(x)$.")

[†]Always keep naive and brute force solutions in mind!

Elements. By Theorem 7.3.2 and Definition 7.3.4, the elements of $\mathbf{F}_{16}$ can be written as the polynomials of degree < 4 (that is, degree ≤ 3) in α. Looking back at (7.7.1), we can get the polynomials of degree ≤ 3 by taking the list in (7.7.1) and adjoining the list you get from adding α^3 to each polynomial in (7.7.1):

$$\begin{aligned}\mathbf{F}_{16} = \big\{&0, 1, \alpha, \alpha+1, \alpha^2, \alpha^2+1, \alpha^2+\alpha, \alpha^2+\alpha+1,\\ &\alpha^3, \alpha^3+1, \alpha^3+\alpha, \alpha^3+\alpha+1,\\ &\alpha^3+\alpha^2, \alpha^3+\alpha^2+1, \alpha^3+\alpha^2+\alpha, \alpha^3+\alpha^2+\alpha+1\big\}.\end{aligned} \tag{7.7.12}$$

Addition. As with $\mathbf{F}_8$, addition in $\mathbf{F}_{16}$ is exactly the same as addition in $\mathbf{F}_2[x]$, remembering that $2 = 0$, as $\mathbf{F}_{16}$ has characteristic 2.

Reduction table. For multiplication and inverses, we list the consequences of $\alpha^4+\alpha+1 = 0$. That is, since $\alpha^4 = \alpha+1$ (again, $+1 = -1!$), we have:

$$\alpha^4 = \alpha+1, \qquad \alpha^5 = \alpha^2+\alpha, \qquad \alpha^6 = \alpha^3+\alpha^2. \tag{7.7.13}$$

Note that these relations allow us to reduce any polynomial of degree between 4 and 6 in α to a polynomial of degree ≤ 3 in α.

Multiplication table. As with $\mathbf{F}_8$, we could use (7.7.13) to construct the multiplication table of $\mathbf{F}_{16}$, but for both typographical reasons and laziness, let's just do one small piece of it:

$\cdot$	$\alpha^3+\alpha^2+\alpha$	$\alpha^3+\alpha^2+\alpha+1$
$\alpha^2+\alpha+1$	$\alpha^3+\alpha^2$	$\alpha^3+\alpha+1$
α^3+1	$\alpha^2+\alpha+1$	$\alpha^3+\alpha^2+\alpha$

Again, please work out those examples, and other randomly chosen examples, yourself.

Inverses. By Corollary 7.3.7, the inverse of $b(\alpha)$ can be computed by applying Euclidean Reduction to $\gcd(b(x), m(x))$ for $m(x) = x^4+x+1$. For example, to find the inverse of $\beta = \alpha^3+\alpha+1$, we take $b(x) = x^3+x+1$ and compute

$$\begin{aligned} x^4+x+1 &= x(x^3+x+1)+(x^2+1)\\ x^3+x+1 &= x(x^2+1)+1.\end{aligned} \tag{7.7.14}$$

Then keeping in mind that $-1 = +1$,

$$\begin{aligned} x^2+1 &= m(x)+xb(x)\\ 1 &= b(x)+x(m(x)+xb(x))\\ &= (1+x^2)b(x)+xm(x).\end{aligned} \tag{7.7.15}$$

It follows that

$$\beta^{-1} = \alpha^2+1. \tag{7.7.16}$$

You should also check that $(\alpha^3+\alpha+1)(\alpha^2+1) = 1$ directly.

Primitive elements and orders. By Problems 7.6.4 and 7.6.5 (since all fields of order 16 are isomorphic!), $\mathbf{F}_{16}^\times$ has 1 element of order 1 (the element 1), 2 elements of order 3,

α^0	1	α^8	α^2+1
α^1	α	α^9	$\alpha^3+\alpha$
α^2	α^2	α^{10}	$\alpha^2+\alpha+1$
α^3	α^3	α^{11}	$\alpha^3+\alpha^2+\alpha$
α^4	$\alpha+1$	α^{12}	$\alpha^3+\alpha^2+\alpha+1$
α^5	$\alpha^2+\alpha$	α^{13}	$\alpha^3+\alpha^2+1$
α^6	$\alpha^3+\alpha^2$	α^{14}	α^3+1
α^7	$\alpha^3+\alpha+1$		

1	α^0	α^3+1	α^{14}
α	α^1	$\alpha^3+\alpha$	α^9
$\alpha+1$	α^4	$\alpha^3+\alpha+1$	α^7
α^2	α^2	$\alpha^3+\alpha^2$	α^6
α^2+1	α^8	$\alpha^3+\alpha^2+1$	α^{13}
$\alpha^2+\alpha$	α^5	$\alpha^3+\alpha^2+\alpha$	α^{11}
$\alpha^2+\alpha+1$	α^{10}	$\alpha^3+\alpha^2+\alpha+1$	α^{12}
α^3	α^3		

Table 7.7.2: Antilog and log tables for $\mathbf{F}_{16}$

4 elements of order 5, and 8 elements of order 15. Those last 8 elements are the primitive elements of $\mathbf{F}_{16}$.

Log-antilog tables. As with $\mathbf{F}_8$, by computing the powers of the primitive element α, we get the log and antilog tables of $\mathbf{F}_{16}$ shown in Table 7.7.2 with respect to α, which reduce multiplication and division in $\mathbf{F}_{16}$ to addition and subtraction mod 15.

Convention 7.7.2. Note that in the log table portions of Tables 7.7.1 and 7.7.2, polynomials in α are ordered by writing out the coefficients of each polynomial as binary digits and then listing them in numerical order. For example, in Table 7.7.1, the log table entries are in the order 001, 010, 011, 100, 101, 110, 111. We call this ordering *dictionary order*, and we take that as our standard ordering for log table entries.

Problems

7.7.1. Let $F = \mathbf{F}_2[\alpha]$, where α is a root of $m(x) = x^3 + x^2 + 1$ (an irreducible polynomial in $\mathbf{F}_2[x]$).

(a) Write out the elements of F as polynomials in α.

(b) Write out the multiplication table of F.

(c) Find the inverse of each nonzero element of F.

(d) Write out the antilog and log tables of F with respect to α, keeping the entries of your log table in dictionary order (Convention 7.7.2).

(e) What is the order of every element of $F^\times$ except 1? Explain.

7.7.2. Let $F = \mathbf{F}_2[\alpha]$, where α is a root of $m(x) = x^4 + x^3 + 1$ (an irreducible polynomial in $\mathbf{F}_2[x]$).

(a) Write out the elements of $F^\times$ as powers of α, i.e., construct the antilog table of F with respect to α.

(b) Write out the log table of F with respect to α, keeping the entries of your log table in dictionary order (Convention 7.7.2).

(c) Use your log and antilog tables to compute $(\alpha^2 + \alpha + 1)(\alpha^3 + \alpha^2 + 1)$ and $(\alpha^3 + \alpha^2)^{-1}$.

7.7.3. Let $F = \mathbf{F}_2[\alpha]$, where α is a root of $m(x) = x^4 + x + 1$ (an irreducible polynomial in $\mathbf{F}_2[x]$), and let $\beta = \alpha + 1$.

(a) Write out the elements of $F^\times$ as powers of β, i.e., construct the antilog table of F with respect to β. (In particular, this will show that β is a primitive element of F.)

(b) Write out the log table of F with respect to β, keeping the entries of your log table in dictionary order (Convention 7.7.2).

(c) Use your log and antilog tables (with respect to β) to compute $(\alpha^2 + 1)(\alpha^3 + \alpha + 1)$ and $(\alpha^3 + \alpha)^{-1}$.

7.7.4. Let $F = \mathbf{F}_2[\alpha]$, where α is a root of $m(x) = x^5 + x^2 + 1$ (an irreducible polynomial in $\mathbf{F}_2[x]$).

(a) Write out the elements of $F^\times$ as powers of α, i.e., construct the antilog table of F with respect to α.

(b) Write out the log table of F with respect to α, keeping the entries of your table in dictionary order (Convention 7.7.2).

(c) Use your log and antilog tables to compute $(\alpha^4+\alpha^2)(\alpha^3+\alpha+1)$ and $(\alpha^4+\alpha^3+\alpha^2+1)^{-1}$.

7.7.5. Let $F = \mathbf{F}_2[\alpha]$, where α is a root of $m(x) = x^6 + x + 1$ (an irreducible polynomial in $\mathbf{F}_2[x]$).

(a) Write out the elements of $F^\times$ as powers of α, i.e., construct the antilog table of F with respect to α.

(b) Write out the log table of F with respect to α, keeping the entries of your table in dictionary order (Convention 7.7.2).

(c) Use your log and antilog tables to compute $(\alpha^4+\alpha^2)(\alpha^3+\alpha+1)$ and $(\alpha^4+\alpha^3+\alpha^2+1)^{-1}$.

Chapter 8

Stronger: BCH codes

A man got to have a code.

— Omar Little, *The Wire*

8.1 How to build a better code

When you look at the success of the Hamming 7- and 8-codes (Chapter 6), it's natural to ask: Can we do better than that? That is, can we devise binary linear codes that can somehow correct even more errors than $\mathcal{H}_7$ and $\mathcal{H}_8$ while also requiring less extra data to be transmitted?

Before we consider that problem, let's review the statistics that tell us how good a binary linear code is. Recall that:

- An $[n, k, d]$ code $\mathcal{C}$ is a binary linear code of *length* n, *dimension* k, and *minimum distance* d. In other words, $\mathcal{C}$ is a subspace of $\mathbf{F}_2^n$, $\dim \mathcal{C} = k$ as a subspace of $\mathbf{F}_2^n$, and the smallest nubmer of 1s appearing in a nonzero codeword of $\mathcal{C}$ is d. (See Section 6.4.)

- We would like k/n to be as large as possible, because k/n represents the portion of each transmitted message that contains useful data.

- Also, since the maximum number of errors that can be corrected in a single transmitted codeword is $\left\lfloor \dfrac{d-1}{2} \right\rfloor$ (Theorem 6.4.13), we would like d to be as large as possible.

It follows that to create a good code, we need to:

Motivating Problem 8.1.1. Find $[n, k, d]$ codes where both k and d are as large as possible, given n.

These goals of large dimension and large minimum distance turn out to have a certain tension with each other. For example, you can achieve a large minimum distance inefficiently by just repeating each data bit many times in the $[n, 1, n]$ repetition code, but it would be more useful to achieve a large minimum distance while also being able to communicate more actual message data per transmitted bit. On the other hand, we have the following class of examples.

Definition 8.1.2. For an integer $r \geq 2$, let $n = 2^r - 1$, and let H_n be the $k \times n$ matrix whose ith column ($1 \leq i \leq n$) is the binary digits of the integer i, written upwards. For example, for $r = 3$ and $r = 4$, we have

$$\begin{aligned} H_7 &= \begin{bmatrix} 1&0&1&0&1&0&1 \\ 0&1&1&0&0&1&1 \\ 0&0&0&1&1&1&1 \end{bmatrix}, \\ H_{15} &= \begin{bmatrix} 1&0&1&0&1&0&1&0&1&0&1&0&1&0&1 \\ 0&1&1&0&0&1&1&0&0&1&1&0&0&1&1 \\ 0&0&0&1&1&1&1&0&0&0&0&1&1&1&1 \\ 0&0&0&0&0&0&0&1&1&1&1&1&1&1&1 \end{bmatrix}. \end{aligned} \tag{8.1.1}$$

The *Hamming n-code* $\mathcal{H}_n$ is the binary linear code whose parity check matrix is H_n.

The Hamming n-code turns out to have the following $[n, k, d]$ statistics.

Theorem 8.1.3. *For an integer $r \geq 2$ and $n = 2^r - 1$, the Hamming n-code $\mathcal{H}_n$ is an $[n, n-r, 3]$ code. It follows that we can correct one error in each n-bit transmission.*

Proof. The $[n, n-r, 3]$ statement is proved in Problems 8.1.2 and 8.1.3. The fact that we can correct one error per transmission follows from Theorem 6.4.13; alternatively, Problem 8.1.4 gives an algorithm for correcting one error per transmitted codeword. □

Big picture: As $n = 2^r - 1$ gets larger, almost all of what we're transmitting is actual data (i.e., $\dfrac{n-r}{r} \to 1$), but we can't correct very many errors (one out of every $n = 2^r - 1$ bits transmitted). Now this might be quite useful — maybe we have a very good but not perfect transmission channel, so we only need to correct errors very rarely. However, Motivating Problem 8.1.1 asks: Is it possible to do both somehow? Or at least is it possible to find some kind of balance besides lots of error correction but very low rate of data transmission (the $[n, 1, n]$ repetition code) or very high rate of data transmission but very little error correction (the Hamming code $\mathcal{H}_n$)?

The general Hamming code construction also begs the following question:

Motivating Problem 8.1.4. Find general methods for constructing good codes.

The rest of this chapter describes one such method, the *BCH construction*. It's remarkable (if mabye not surprising, given the previous chapter) that the keys to the BCH construction are two quite abstract ideas: *ideals* and *finite fields*.

Problems

8.1.1. For $r = 2$ and $n = 2^2 - 1 = 3$, what are the codewords in the Hamming 3-code $\mathcal{H}_3$? What is another name for this code that you've seen?

8.1.2. As in Definition 8.1.2, let r be a positive integer, let $n = 2^r - 1$, let H_n be the $r \times n$ matrix whose ith column $(1 \leq i \leq n)$ is the binary digits of the integer i, written upwards, and let $\mathcal{H}_n$ be the code with parity check matrix H_n.

(a) Note that H_7 and H_{15} are both already in RREF. Which columns of H_7 are pivot columns? Which columns of H_{15} are pivot columns?

(b) Now consider an arbitrary positive integer r and $n = 2^r - 1$. Explain why the matrix H_n is already in RREF.

(c) Prove that the dimension of the Hamming n-code $\mathcal{H}_n$ is $(2^r - 1) - r$.

8.1.3. Let r, n, H_n, and $\mathcal{H}_n$ be as in Problem 8.1.2.

(a) Prove that $\mathcal{H}_n$ has no words of weight 1. (Suggestion: Any word of weight 1 has the form $\mathbf{e}_i$.)

(b) Prove that $\mathcal{H}_n$ has no words of weight 2. (Suggestion: Any word of weight 2 has the form $\mathbf{e}_i + \mathbf{e}_j$ for some $i \neq j$.)

(c) Prove that $\mathcal{H}_n$ always has a word of weight 3. (Suggestion: Find three columsn of H_n that sum to 0.)

8.1.4. Let r, n, H_n, and $\mathcal{H}_n$ be as in Problem 8.1.2. Prove that the error-correction scheme from Theorem 6.3.4 also works for $\mathcal{H}_n$. (Suggestion: Imitate the proof of Theorem 6.3.4.)

8.2 Cyclic codes

The first key ingredient in the BCH construction is to consider the following class of codes.

Definition 8.2.1. Let $\mathcal{C}$ be a binary linear code of length n. To say that $\mathcal{C}$ is *cyclic* means that it is closed under cyclic permutation of coordinates. That is, to say that $\mathcal{C}$ is cyclic means that if $\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix}$ is in $\mathcal{C}$, then so are $\begin{bmatrix} c_{n-1} \\ c_0 \\ c_1 \\ \vdots \\ c_{n-2} \end{bmatrix}$, $\begin{bmatrix} c_{n-2} \\ c_{n-1} \\ c_0 \\ \vdots \\ c_{n-3} \end{bmatrix}$, and so on.

Notation 8.2.2. Later on, starting in Section 8.7, we'll look at cyclic codes over other fields (see Section 8.7 if you're curious about what that means), but for now, when we say "cyclic code," we mean "cyclic binary linear code."

Example 8.2.3. The code

$$\mathcal{C} = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right\} \tag{8.2.1}$$

is a cyclic code of length 4 and dimension 3. Permuting coordinates cyclically cycles codewords 2–5, as well as 6 and 7, and fixes codewords 1 and 8. You can also check by brute force that $\mathcal{C}$ is a subspace; in fact, $\mathcal{C}$ is precisely the parity check code of length 4.

Example 8.2.4. The code

$$\begin{aligned}\mathcal{C} = \Bigg\{ & \begin{bmatrix}0\\0\\0\\0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\0\\1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\\0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\1\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\\1\\0\\1\end{bmatrix}, \begin{bmatrix}1\\0\\0\\0\\1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\\0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\\0\\0\\0\\1\end{bmatrix}, \\ & \begin{bmatrix}0\\0\\1\\0\\1\\1\\1\end{bmatrix}, \begin{bmatrix}1\\0\\0\\1\\0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\1\\0\\0\\1\\0\\1\end{bmatrix}, \begin{bmatrix}1\\1\\1\\0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\\1\\0\\0\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\\1\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\1\\1\\1\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\\1\\1\\1\\1\end{bmatrix} \Bigg\}\end{aligned} \tag{8.2.2}$$

is a cyclic code of length 7 and dimension 4. Permuting coordinates cyclically cycles codewords 2–8, as well as vectors 9–15, and fixes codewords 1 and 16. Again, you can check by brute force that $\mathcal{C}$ is a subspace; in fact, we'll see that $\mathcal{C}$ is variation on the Hamming 7-code $\mathcal{H}_7$ (see Example 8.3.6).

The symmetry condition in Definition 8.2.1 is reasonably natural, as it is often the case that "good" codes are more symmetric than usual. However, the really useful thing about cyclic codes can be seen by writing codewords using the following notation.

Notation 8.2.5. The *polynomial notation* for vectors in $\mathbf{F}_2^n$ represents the vector $\begin{bmatrix}c_0\\ \vdots \\ c_{n-1}\end{bmatrix}$ as the polynomial

$$c_0 + c_1x + c_2x^2 + \cdots + c_{n-1}x^{n-1} \tag{8.2.3}$$

in the ring $R = \mathbf{F}_2[x]/(x^n - 1)$ (i.e., setting $x^n \equiv 1$).

Note that in $R = \mathbf{F}_2[x]/(x^n - 1)$, addition of polynomials and multiplication of a polynomial by a scalar both work exactly like they do for the corresponding vectors. What makes the polynomial notation useful is that if

$$c(x) = c_0 + c_1x + c_2x^2 + \cdots + c_{n-1}x^{n-1} \tag{8.2.4}$$

is a codeword, then since $x^n \equiv 1$ in R, we have that

$$\begin{aligned} xc(x) &= c_0x + c_1x^2 + c_2x^3 + \cdots + c_{n-2}x^{n-1} + c_{n-1}x^n \\ &= c_{n-1} + c_0x + c_1x^2 + c_2x^3 + \cdots + c_{n-2}x^{n-1}, \end{aligned} \tag{8.2.5}$$

which is precisely the (one-step) cyclic permutation of the coordinates of $c(x)$. Similarly, $x^2c(x)$ cyclically permutes the coordinates of $c(x)$ by two steps, and in general, $x^kc(x)$ cyclically permutes the coordinates of $c(x)$ by k steps. As we'll see momentarily, that means that cyclic codes can be characterized algebraically in terms of ideals. (Surprise!)

Notation 8.2.6. Note that for the ring $\mathbf{F}_2[x]/(x^n-1)$, instead of using the α notation (Notation 7.3.5) established earlier, we'll continue to refer to elements of $\mathbf{F}_2[x]/(x^n-1)$ as polynomials $f(x)$ in x, with the understanding that $x^n = 1$. The reason is that we will later need to plug elements of some $\mathbf{F}_q = \mathbf{F}_2[\alpha]$ into these polynomials $f(x)$.

Theorem 8.2.7. *Let $\mathcal{C}$ be a binary linear code of length n. In polynomial notation, $\mathcal{C}$ is cyclic if and only if it is an ideal of the ring $\mathbf{F}_2[x]/(x^n-1)$.*

Proof. On the one hand, suppose $\mathcal{C}$ is an ideal of $\mathbf{F}_2[x]/(x^n-1)$. Since $\mathcal{C}$ is closed under multiplication by $x \in \mathbf{F}_2[x]/(x^n-1)$, by the previous discussion if $c(x) \in \mathcal{C}$, so are all of its cyclic permutations $x^kc(x)$. It follows that $\mathcal{C}$ is cyclic.

For the converse, see Problem 8.2.1. □

Problems

8.2.1. Suppose $\mathcal{C}$ is a cyclic code of length n over $\mathbf{F}_2$. We continue to use the polynomial notation (Notation 8.2.5) and think of $\mathbf{F}_2^n$ as $\mathbf{F}_2[x]/(x^n-1)$. The goal of this problem is to prove that (explain why) $\mathcal{C}$ is an ideal (Definition 7.1.1) of $\mathbf{F}_2[x]/(x^n-1)$.

(a) Explain why $\mathcal{C}$ contains zero, is closed under addition, and is closed under scalar multiplication by elements of $\mathbf{F}_2$.

(b) Suppose $c(x) \in \mathcal{C}$. Explain why $xc(x) \in \mathcal{C}$.

(c) Suppose $c(x) \in \mathcal{C}$. For $a, b \in \mathbf{F}_2$, explain why $(ax + bx^2)c(x) \in \mathcal{C}$.

(d) Suppose $c(x) \in \mathcal{C}$ and
$$f(x) = a_0 + a_1x^1 + \cdots + a_kx^k \tag{8.2.6}$$
is an element of $\mathbf{F}_2[x]$. Explain why $f(x)c(x) \in \mathcal{C}$.

8.3 Cyclic codes and generator polynomials

Returning to cyclic codes, we first have the following variation on Theorem 7.4.2.

Theorem 8.3.1. *Fix a positive integer n, and let $\mathcal{C}$ be a nonzero cyclic code of length n, i.e., let $\mathcal{C}$ be a nonzero ideal of $\overline{R} = \mathbf{F}_2[x]/(x^n-1)$. Then $\mathcal{C}$ is principal, or in other words, $\mathcal{C} = (g(x))$ for some $g(x) \in \mathbf{F}_2[x]$. Moreover, we can choose $g(x)$ so that $g(x)$ divides x^n-1.*

Again, when we say that $\mathcal{C} = (g(x))$, we're using $g(x)$ as an abbreviation for $g(x) + I_0$, where $I_0 = (x^n-1)$.

Proof. Let $I_0 = (x^n - 1)$, so elements of $\overline{R}$ have the form $f(x) + I_0$ for some $f(x) \in \mathbf{F}_2[x]$; and let

$$I = \{f(x) \in \mathbf{F}_2[x] \mid f(x) + I \in \mathcal{C}\}. \tag{8.3.1}$$

Then we can use the definition of $\overline{R} = \mathbf{F}_2[x]/I_0$ and the fact that $\mathcal{C}$ is an ideal of $\overline{R}$ to show that I is an ideal of $\mathbf{F}_2[x]$ containing I_0; see Problem 8.3.1 for details.

In any case, since I is an ideal of the PID $\mathbf{F}_2[x]$, $I = (g(x))$ for some $g(x) \in \mathbf{F}_2[x]$, and reducing mod I_0, we see that $\mathcal{C} = (g(x) + I_0)$, which we abbreviate as $\mathcal{C} = (g(x))$. Furthermore, since $x^n - 1 \in I_0 \subseteq I$, $g(x)$ divides $x^n - 1$, and the theorem follows. □

Definition 8.3.2. Let $\mathcal{C}$ be a cyclic code of length n. We define the *generator polynomial* of $\mathcal{C}$ to be the minimal polynomial $g(x)$ of $\mathcal{C}$, as described in Theorem 8.3.1.

Therefore, if we want to study the cyclic codes of length n, it suffices to consider each possible factor $g(x)$ of $x^n - 1$ over $\mathbf{F}_2$ and study the corresponding principal ideals $(g(x))$. Still easier said than done! But at least this considerably limits the kind of example we need to look at.

Before we turn to one particular method of making good cyclic codes (Sections 8.4–8.8), we first establish some facts that hold for cyclic codes in general, beginning with the dimension of a cyclic code.

Theorem 8.3.3. *Let $\mathcal{C}$ be a cyclic code of length n generated by the divisor $g(x) \in \mathbf{F}_2[x]$ of $x^n - 1$. If $\deg g(x) = r$, then the set*

$$\mathcal{B} = \left\{g(x), xg(x), \dots, x^{(n-1)-r}g(x)\right\} \tag{8.3.2}$$

is a basis for $\mathcal{C}$. Consequently, the dimension of $\mathcal{C}$ is $k = n - r$.

Proof. We first show that $\mathcal{B}$ is linearly independent. Let

$$g(x) = c_0 + c_1 x + \cdots + c_{r-1}x^{r-1} + x^r, \tag{8.3.3}$$

where the coefficient of x^r must be 1 because $\deg g(x) = r$. Then if G is the matrix whose columns are $\mathcal{B}$ in ordinary vector form, we have

$$G = \begin{bmatrix} c_0 & 0 & 0 & \cdots & 0 \\ c_1 & c_0 & 0 & \cdots & 0 \\ c_2 & c_1 & c_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & c_{r-1} & c_{r-2} & \cdots & 0 \\ 0 & 1 & c_{r-1} & \cdots & c_0 \\ 0 & 0 & 1 & \cdots & c_1 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}, \tag{8.3.4}$$

where the lower right-hand entry corresponds to the term x^{n-1} in $x^{(n-1)-r}g(x)$. We see that G is a matrix in REF in which every column is a leading column, and it follows that $G\mathbf{x} = \mathbf{0}$ has $\mathbf{x} = \mathbf{0}$ as its only solution, or in other words, that $\mathcal{B}$ is linearly independent.

To see why $\mathcal{B}$ spans $\mathcal{C}$, see Problem 8.3.3. □

Encoding and reading an error-free transmission are then straightforward:

- To encode a message $\mathbf{m} = \begin{bmatrix} m_0 \\ \vdots \\ m_{n-r-1} \end{bmatrix}$, we write $\mathbf{m}$ in polynomial notation as

$$m(x) = m_0 + m_1 x + \cdots + m_{n-r-1}x^{n-r-1} \tag{8.3.5}$$

  and transmit the codeword

$$m(x)g(x) = m_0 g(x) + m_1 x g(x) + \cdots + m_{n-r-1}x^{n-r-1}g(x). \tag{8.3.6}$$

- Conversely, to read a received codeword $y(x)$ that has had no errors in transmission, we use polynomial division to recover

$$m(x) = y(x)/g(x). \tag{8.3.7}$$

  Note that if $g(x)$ does not divide $y(x)$, then $y(x)$ is not an element of $\mathcal{C} = (g(x))$, and an error must have occurred in transmission.

Of course, as is always the case with error-correcting codes (compare Section 6.2), actually *correcting* errors is more difficult, and the error-correcting procedures we use very much depend on the details of the different examples we consider.

We end this section by looking at a few examples of cyclic codes; see Problems 8.3.6–8.3.7 for some more examples.

Example 8.3.4 (Parity check as cyclic)**.** For $n \in \mathbf{N}$, note that

$$x^n - 1 = (x-1)(1 + x + \cdots + x^{n-1}) \tag{8.3.8}$$

in $\mathbf{F}_2[x]$. Let $\mathcal{C}$ be the cyclic code of length n generated by the divisor $1 + x$ (which is the same as $x - 1$ with coefficients (mod 2)). Then $\mathcal{C}$ is precisely the parity check code of length n from Example 6.2.8 (see Problem 8.3.4). In particular, when $n = 4$, the polynomial multiples of $x + 1$ of degree ≤ 3 are precisely

$$\begin{aligned} 0(x+1) &= 0 & 1(x+1) &= x+1 \\ x(x+1) &= x^2 + x & (x+1)(x+1) &= x^2 + 1 \\ x^2(x+1) &= x^3 + x^2 & (x^2+1)(x+1) &= x^3 + x^2 + x + 1 \\ (x^2+x)(x+1) &= x^3 + x & (x^2+x+1)(x+1) &= x^3 + 1. \end{aligned} \tag{8.3.9}$$

If you compare Example 8.2.3, you'll see that the polynomials appearing on the right-hand sides of (8.3.9) are precisely the same as the codewords in (8.2.1), but in polynomial notation.

Example 8.3.5 (Repetition as cyclic)**.** Let $\mathcal{C}$ be the cyclic code of length n generated by $(1+x+\cdots+x^{n-1})$. Then $\mathcal{C}$ is precisely the repetition code of length n from Example 6.2.9; see Problem 8.3.5.

Example 8.3.6 ($\mathcal{H}_7$ as cyclic)**.** Let $\mathcal{C}$ be the cyclic code of length 7 over $\mathbf{F}_2$ generated by $1+x+x^3$. We now show that, after a change of coordinates, $\mathcal{C}$ is the Hamming 7-code $\mathcal{H}_7$.

First, following the proof of Theorem 8.3.3, we see that

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8.3.10}$$

is a generator matrix for $\mathcal{C}$. See Example 8.2.4 for a list of the actual codewords of $\mathcal{C}$; note that the basis for $\mathcal{C}$ given in (8.3.10) appears as vectors 2–5 in (8.2.2).

Next, we claim that

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{8.3.11}$$

is a parity check matrix for $\mathcal{C}$. While it may be unclear how H was derived, the fact that HG is the zero matrix (check this for yourself!) and $\operatorname{rank} H = 3 = 7-4$ means that the columns of G are a basis for $\operatorname{Null}(H)$, which must therefore be $\mathcal{C}$.

We then observe that the columns of H are the binary digits (written backwards) of $3, 6, 7, 5, 1, 2, 4$, respectively. It follows that $\mathcal{C}$ is the Hamming 7-code, but with bits $x_1, \dots, x_7$ moved to positions $5, 6, 1, 7, 4, 2, 3$, respectively.

Example 8.3.7 ($\mathcal{H}_n$ as cyclic)**.** Similarly, for an integer $r \geq 2$ and $n = 2^r - 1$, it turns out that if $g(x) \in F_2[x]$ is an irreducible polynomial of degree r, then the cyclic code generated by $g(x)$ can be renumbered to be the Hamming code $\mathcal{H}_n$. See Problem 8.3.8 for a proof.

Example 8.3.8. The cyclic code of length 23 generated by

$$g(x) = x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1 \tag{8.3.12}$$

turns out to be a $[23, 12, 7]$ code called the *Golay 23-code*. For an explanation of where $g(x)$ comes from, see Example 8.5.9; or for much more about the Golay 23-code, MacWilliams and Sloane Ch. 16 and 20.

Examples 8.3.6 and 8.3.8 certainly show that it is possible to find good cyclic codes. The rest of this chapter is therefore devoted to a solution to the following problem:

Motivating Problem 8.3.9. How can we choose $n \in \mathbf{N}$ and $g(x) \in \mathbf{F}_2[x]$ dividing $x^n - 1$ so that the cyclic code $\mathcal{C}$ generated by $g(x)$ has both a relatively large dimension and a large minimum distance?

Problems

8.3.1. (*Proves Theorem 8.3.1*)

Let R be a ring, let I_0 be an ideal of R, let J be an ideal of R/I_0, and let

$$I = \{r \in R \mid r + I_0 \in J\}. \tag{8.3.13}$$

Prove that I is an ideal of R containing I_0, as follows.

(a) Explain why $0 \in I$.

(b) Suppose $a \in I_0$. Explain why $a \in I$.

(c) Suppose $a, b \in I$. Explain why $a + b \in I$.

(d) Suppose $a \in I$ and $r \in R$. Explain why $ra \in I$.

(Suggestion for all parts: Since J is an ideal of R/I_0, how do the three axiomatic properties of ideals translate into properties of J inside R/I_0?)

8.3.2. Let $\mathcal{C}$ be the cyclic code of length 9 generated by x^5+x^4+1, or in other words, let $\mathcal{C}$ be the principal ideal (x^5+x^4+1). Find the generating polynomial (minimal polynomial) $g(x)$ of $\mathcal{C}$. (Suggestion: Note that by Theorem 8.3.1, $g(x)$ must divide both x^5+x^4+1 and x^9-1.)

8.3.3. (*Proves Theorem 8.3.3*) Let $\mathcal{C}$ be a cyclic code of length n generated by the divisor $g(x) \in \mathbf{F}_2[x]$ of $x^n - 1$, and suppose that $\deg g(x) = r$. Let

$$\mathcal{B} = \left\{ g(x), xg(x), \dots, x^{(n-1)-r} g(x) \right\}, \tag{8.3.14}$$

and suppose $f(x) \in \mathcal{C}$. Since we are working mod (x^n-1), we may assume that $\deg f \leq n-1$.

(a) Explain why it must be the case that $f(x) = q(x)g(x)$ for some polynomial $q(x)$ of degree at most $(n-1)-r$.

(b) Prove that $f(x)$ is a linear combination of the elements of $\mathcal{B}$.

8.3.4. Let $\mathcal{C}$ be the cyclic code of length n generated by $1+x$.

(a) Write out the generator matrix of $\mathcal{C}$, as in the proof of Theorem 8.3.3. What is the dimension of $\mathcal{C}$?

(b) Prove that each basis element from part (a) is a codeword of the parity check code of length n (Example 6.2.8). (Since the two codes have the same dimension, it follows that they must be equal.)

8.3.5. Let $\mathcal{C}$ be the cyclic code of length n generated by $1 + x + \cdots + x^{n-1}$.

(a) Write out the generator matrix of $\mathcal{C}$, as in the proof of Theorem 8.3.3. What is the dimension of $\mathcal{C}$?

(b) Prove that each basis element from part (a) is a codeword of the repetition code of length n (Example 6.2.9). (Since the two codes have the same dimension, it follows that they must be equal.)

8.3.6. Let $\mathcal{C}$ be the cyclic code of length 8 generated by $1 + x + x^4 + x^5$.

(a) Write out the generator matrix of $\mathcal{C}$, as in the proof of Theorem 8.3.3. What is the dimension of $\mathcal{C}$?

(b) Find the minimum distance of $\mathcal{C}$, i.e., the smallest number of 1s in any nonzero codeword of $\mathcal{C}$. How many errors does $\mathcal{C}$ correct? How many does it detect? Briefly explain.

8.3.7. Let $\mathcal{C}$ be the cyclic code of length 9 generated by $1 + x^3 + x^6$.

(a) Write out the generator matrix of $\mathcal{C}$, as in the proof of Theorem 8.3.3. What is the dimension of $\mathcal{C}$?

(b) Find the minimum distance of $\mathcal{C}$, i.e., the smallest number of 1s in any nonzero codeword of $\mathcal{C}$. How many errors does $\mathcal{C}$ correct? How many does it detect? Briefly explain.

8.3.8. (proof using finite fields).

8.4 Minimal polynomials

To be able to state, let alone explain, our theorem that will give one solution to Motivating Problem 8.3.9, we need the following definition.

Definition 8.4.1. An *extension field*, or if the context is clear, simply an *extension*, of a field F is a field E that contains F as a subfield (i.e., a subset of E that is itself a field, using the same operations of $+$ and $\cdot$). In particular, a *finite extension* of a finite field $\mathbf{F}_q$ is an extension E of $\mathbf{F}_q$ such that E itself is a finite field.

For a silly example, every field is an extension of itself. For some less silly examples, the complex numbers $\mathbf{C}$ are an extension of the real numbers $\mathbf{R}$, and by the Five Facts for Finite Fields, every field of characteristic 2 is an extension of $\mathbf{F}_2$.

Definition 8.4.2. Let E be an extension field of the field F, and suppose $f(x) \in F[x]$. Note that $f(x)$ is also a polynomial with coefficients in E, which means that $f(x)$ can be factored either in $F[x]$ or in $E[x]$. To distinguish between these ideas, we use the term *factoring over* F to mean $f(x) = g(x)h(x)$ with $g(x), h(x) \in F[x]$, and *factoring over* E to mean $f(x) = g(x)h(x)$ with $g(x), h(x) \in E[x]$. The terms *irreducible over* F and *irreducible over* E are defined similarly.

Example 8.4.3. As you may remember from precalculus, the polynomial $x^2 + 1$ is irreducible over $\mathbf{R}$, but factors as $x^2 + 1 = (x + i)(x - i)$ over $\mathbf{C}$.

Example 8.4.4. The polynomial $x^3 + x + 1$ is irreducible over $\mathbf{F}_2$, but if α is a root of $x^3 + x + 1$ in $\mathbf{F}_8$, then

$$x^3 + x + 1 = (x - \alpha)(x - \alpha^2)(x - \alpha^4) \tag{8.4.1}$$

over $\mathbf{F}_8$. (Try expanding (8.4.1) yourself; see Problem 8.4.1.) In a minute, we'll see where the form of this factorization comes from.

Returning to our main thread, we may now state the theorem that makes the subsequent material in this section useful.

Theorem 8.4.5. *Let $\mathcal{C}$ be a cyclic code of length n generated by the divisor $g(x) \in \mathbf{F}_2[x]$ of $x^n - 1$. Suppose E is an extension of $\mathbf{F}_2$ such that for some $\delta \in \mathbf{N}$ and some $\alpha \in E$ with the order of α exactly equal to n, we have that*

$$0 = g(\alpha) = g(\alpha^2) = g(\alpha^3) = \cdots = g(\alpha^{\delta-1}). \tag{8.4.2}$$

Then the minimum distance d of $\mathcal{C}$ is at least δ, i.e., $d \geq \delta$.

In other words, if we can find a generator polynomial $g(x)$ for a cyclic code $\mathcal{C}$ of length n that has roots of the form $\alpha, \alpha^2, \alpha^3, \ldots$, then $\mathcal{C}$ has a large minimum distance, and can therefore correct more errors (see Theorem 6.4.13). On the other hand, by Theorem 8.3.3, the smaller the degree of $g(x)$, the larger the dimension of $\mathcal{C}$, and the more data bits that can be transmitted inside each codeword of length n (see Motivating Problem 8.1.1).

We are therefore motivated to solve the following problem.

Motivating Problem 8.4.6. Let E be an extension of $\mathbf{F}_2$, and suppose $\alpha \in E^\times$ has order n. Find $g(x) \in \mathbf{F}_2[x]$ of smallest possible degree such that $g(x)$ divides $x^n - 1$ and

$$0 = g(\alpha) = g(\alpha^2) = \cdots = g(\alpha^{\delta-1}) \tag{8.4.3}$$

for as large a value of δ as possible.

To solve Motivating Problem 8.4.6, we'll need a few more facts about finite fields beyond those from Section 7.6. (You should also review the Five Facts for Finite Fields.)

Theorem 8.4.7. *Let E be a finite extension of $\mathbf{F}_2$, and define a function $\rho : E \to E$ by the formula*

$$\rho(\beta) = \beta^2. \tag{8.4.4}$$

1. *If E is a finite extension of $\mathbf{F}_2$, then $\beta \in E$ is a root of $x^2 - x$ if and only if $\beta \in \mathbf{F}_2$.*

2. *The map ρ is an automorphism of E. Furthermore, ρ fixes exactly the subfield $\mathbf{F}_2$; in other words, for $\beta \in E$, $\rho(\beta) = \beta$ if and only if $\beta \in \mathbf{F}_2$.*

Proof. On the one hand, each of 0 and 1 is a root of $x^2 - x$, and on the other hand, $x^2 - x$ can have at most two roots, so those roots must be precisely the subfield $\mathbf{F}_2 = \{0, 1\}$ of E.

For (2), by Theorem 7.6.17 and the discussion immediately afterwards, we know that E must have order 2^e for some $e \in \mathbf{N}$, which means that for any $\beta \in E$, $\beta^{2^e} = \beta$ (Corollary 7.6.15). Therefore, for any $\beta \in E$,

$$\rho^e(\beta) = \beta^{2^e} = \beta, \tag{8.4.5}$$

and so ρ^e is the identity function, implying that ρ is a bijection. Furthermore, the statement that ρ fixes exactly $\mathbf{F}_2$ follows directly from part (1) of the theorem.

So it remains to show that ρ is a homomorphism. Certainly $(\beta_1\beta_2)^2 = \beta_1^2\beta_2^2$, so we just have to show that $(\beta_1 + \beta_2)^2 = \beta_1^2 + \beta_2^2$. However, since $\mathbf{F}_2$ is a subfield of E, the characteristic of E is 2, so

$$(\beta_1 + \beta_2)^2 = \beta_1^2 + 2\beta_1\beta_2 + \beta_2^2 = \beta_1^2 + \beta_2^2. \tag{8.4.6}$$

The theorem follows. □

Definition 8.4.8. The function $\rho(\beta) = \beta^2$ defined in Theorem 8.4.7 is called a *Frobenius automorphism.*[*]

Turning to the problem of finding a "smallest" $g(x)$ such that $g(\beta) = 0$, the following theorem makes the idea of "smallest" precise.

Theorem 8.4.9. *Let E be an extension of $\mathbf{F}_2$, fix some $\beta \in E$, and let*

$$I = \{f(x) \in \mathbf{F}_2[x] \mid f(\beta) = 0\}. \tag{8.4.7}$$

Then I is an ideal of $\mathbf{F}_2[x]$, and consequently, $I = (m(x))$ for some $m(x) \in \mathbf{F}_2[x]$.

In other words, $f(\beta) = 0$ exactly when (if and only if) $f(x)$ is a multiple of $m(x)$.

Proof. Problem 8.4.3 shows that I is an ideal, and the last assertion follows because every idea of $\mathbf{F}_2[x]$ is principal (Theorem 7.4.2). □

Definition 8.4.10. We call the polynomial $m(x) \in \mathbf{F}_2[x]$ in Theorem 8.4.9 the *minimal polynomial of β over $\mathbf{F}_2$.*

By Theorem 8.3.3, to maximize the dimension of a cyclic code, we want a generator polynomial of as small a degree as possible, which means that we'll want to choose a generator that is a least common multiple of minimal polynomials. It will therefore be useful to establish the following notation for minimal polynomials.

Notation 8.4.11. If E is an extension of $\mathbf{F}_2$, and we fix an element $\alpha \in E$, then $m_i(x)$ denotes the minimal polynomial of α^i over $\mathbf{F}_2$.

[*]You might be wondering, why make up this separate fancy name, when we could just call this "squaring"? The answer is that ρ generalizes to other fields, and specifically, other extensions of fields; see (blah).

We can compute $m_i(x)$ efficiently using the following idea.

Definition 8.4.12. Let E be an extension of $\mathbf{F}_2$, let β be an element of E, and define

$$\beta_n = \rho^n(\beta). \tag{8.4.8}$$

In other words, let β_n be the result of applying ρ to β n times, e.g., $\beta_3 = \rho(\rho(\rho(\beta)))$. The *Frobenius orbit of* β is the set

$$\{\beta_0 = \beta, \beta_1, \beta_2, \dots\}. \tag{8.4.9}$$

Note that since some finite power of ρ is the identity, every Frobenius orbit is finite.

Theorem 8.4.13 (Orbit Theorem)**.** *Let E be an extension of $\mathbf{F}_2$, let β be in $E^\times$, and suppose the Frobenius orbit of β is $\{\beta_0, \dots, \beta_{s-1}\}$, where $\beta_k = \rho^k(\beta)$ and $\rho^s(\beta) = \beta$. Then the minimal polynomial of β over $\mathbf{F}_2$ is*

$$m(x) = (x - \beta_0)(x - \beta_1)\dots(x - \beta_{s-1}). \tag{8.4.10}$$

Furthermore, if β has order n, then $m(x)$ divides $x^n - 1$.

The "symmetrizing" idea of Theorem 8.4.13 is closely related to the fact that, for example, the complex roots of a real polynomial come in conjugate pairs; see Example 7.5.16.

Proof. First, we need to show that the polynomial $m(x)$, which looks like it might have coefficients in the bigger field E, is actually in $\mathbf{F}_2[x]$. Let $\Phi : E[x] \to E[x]$ be the automorphism induced by applying the automorphism $\rho : E \to E$ to the coefficients of each $f(x) \in E[x]$ (Example 7.5.4). Then

$$\begin{aligned}\Phi(m(x)) &= \Phi((x - \beta_0)(x - \beta_1)\dots(x - \beta_{s-2})(x - \beta_{s-1}))\\ &= (x - \rho(\beta_0)(x - \rho(\beta_1))\dots(x - \rho(\beta_{s-2}))(x - \rho(\beta_{s-1}))\\ &= (x - \beta_1)(x - \beta_2)\dots(x - \beta_{s-1})(x - \beta_0)\\ &= m(x).\end{aligned} \tag{8.4.11}$$

Therefore, by Theorem 8.4.7(2), the coefficients of $m(x)$ are actually in $\mathbf{F}_2$, and not just in E; in other words, $m(x) \in \mathbf{F}_2[x]$.

So now, as in Theorem 8.4.9, let

$$I = \{f(x) \in \mathbf{F}_2[x] \mid f(\beta) = 0\}. \tag{8.4.12}$$

On the one hand, since $m(\beta) = m(\beta_0) = 0$, we see that $m(x) \in I$, which means that the ideal $(m(x))$ is contained in I, as I is closed under taking $\mathbf{F}_2[x]$-scalar multiples.

On the other hand, suppose $f(x) \in I$, or in other words, suppose that

$$f(x) = a_0 + a_1 x + \dots + a_k x^k \in \mathbf{F}_2[x] \tag{8.4.13}$$

and $f(\beta) = 0$. Since ρ fixes $\mathbf{F}_2$, by Theorem 7.5.15,

$$f(\beta_1) = f(\rho(\beta)) = 0. \tag{8.4.14}$$

Repeating this process, we see that $f(\beta_i) = 0$ for $0 \leq i \leq s-1$, which means that $m(x)$ divides $f(x)$, by the Root Theorem. It follows that I is contained in $(m(x))$.

Finally, since $\beta^n - 1 = 0$, $x^n - 1$ is in I, which means that $x^n - 1$ is a multiple of the minimal polynmomial $m(x)$. The theorem follows. □

Remark 8.4.14. Note that if elements $\alpha^i, \alpha^j \in E$ are in the same Frobenius orbit, then the corresponding products of the form in (8.4.10) will contain the same factors, just in a different cyclic order. It follows from Theorem 8.4.13 that:

> Elements α^i, α^j in the same Frobenius orbit have the same minimal polynomials $m_i(x) = m_j(x)$.

Example 8.4.15. Consider the extension $E = \mathbf{F}_8$ of $\mathbf{F}_2$, and let α be a primitive root of E. To compute the Frobenius orbit of α^i, we start with α^i and square what we have until we get back to α^i, keeping in mind that $\alpha^7 = 1$. So the Frobenius orbit of α^1 is

$$O_1 = \left\{\alpha^1, \alpha^2, \alpha^4\right\},$$

because when we get to α^8, we return to $\alpha^8 = \alpha^1$. Note that O_1 is also the Frobenius orbit of α^2 and α^4. Putting that together with Theorem 8.4.13 and Remark 8.4.14, we see that

$$m_1(x) = m_2(x) = m_4(x) = (x-\alpha)(x-\alpha^2)(x-\alpha^4). \tag{8.4.15}$$

See Problem 8.4.1 for what $m_1(x)$ looks like as a polynomial in $\mathbf{F}_2[x]$.

Similarly, the Frobenius orbit of α^3 is

$$O_1 = \left\{\alpha^3, \alpha^6, \alpha^5\right\}, \tag{8.4.16}$$

and

$$m_3(x) = m_5(x) = m_6(x) = (x-\alpha^3)(x-\alpha^5)(x-\alpha^6). \tag{8.4.17}$$

Example 8.4.16. For a bigger example, let $q = 2^{12} = 4096$, consider the extension $E = \mathbf{F}_q$ of $\mathbf{F}_2$, let β be a primitive root of E (of order 4095), and let $\alpha = \beta^{105}$. By the Order of a Power Formula (Theorem 7.6.11), the order of α is $4095/105 = 39$, which is all we need to know to compute the Frobenius orbits of α^i.

First, using the facts that $\alpha^{64} = \alpha^{25}$, $\alpha^{44} = \alpha^5$, and $\alpha^{40} = \alpha$ (i.e., exponents are computed mod 39), we have

$$O_1 = \left\{\alpha^1, \alpha^2, \alpha^4, \alpha^8, \alpha^{16}, \alpha^{32}, \alpha^{25}, \alpha^{11}, \alpha^{22}, \alpha^5, \alpha^{10}, \alpha^{20}\right\}. \tag{8.4.18}$$

To avoid writing α over and over, we can just write the exponents and omit α, and rewrite (8.4.18) as

$$O_1 = \{1, 2, 4, 8, 16, 32, 25, 11, 22, 5, 10, 20\}. \tag{8.4.19}$$

In fact, in this notation, we can think of computing Frobenius orbits as "doubling mod the order of α." In the same notation, we also have

$$\begin{aligned} O_3 &= \{3, 6, 12, 24, 9, 18, 36, 33, 27, 15, 30, 21\}, \\ O_7 &= \{7, 14, 28, 17, 34, 29, 19, 38, 37, 35, 31, 23\}, \\ O_{13} &= \{13, 26\}. \end{aligned} \tag{8.4.20}$$

Applying Theorem 8.4.13, we see that

$$\begin{aligned} m_1(x) = {} & (x-\alpha^1)(x-\alpha^2)(x-\alpha^4)(x-\alpha^8)(x-\alpha^{16})(x-\alpha^{32}) \\ & (x-\alpha^{25})(x-\alpha^{11})(x-\alpha^{22})(x-\alpha^5)(x-\alpha^{10})(x-\alpha^{20}), \end{aligned} \tag{8.4.21}$$

and $m_3(x), m_7(x), m_{13}(x)$ can be obtained similarly. To express $m_1(x)$ explicitly as a polynomial in $\mathbf{F}_2[x]$, though, we need to do two things: One, choose a particular irreducible polynomial of degree 12 to construct $\mathbf{F}_q$; and two, resort to machine calculation, because who wants to use pencil and paper to expand a polynomial of degree 12 with coefficients in a finite field?

So let $\mathbf{F}_q = \mathbf{F}_2[\beta]$, where β is a root of $m(x) = x^{12} + x^6 + x^4 + x + 1$. By (original source??) (i.e., you can look it up on the internet), $m(x)$ is irreducible and β is a primitive element of $\mathbf{F}_q$, so we can indeed take $\alpha = \beta^{105}$. A computation in the SageMath system, similar to the computations in Problem 8.4.1 but a whole lot longer, shows that

$$m_1(x) = x^{12} + x^{10} + x^9 + x^8 + x^7 + x^3 + x^2 + x + 1. \tag{8.4.22}$$

Alternatively, we could let $\mathbf{F}_q = \mathbf{F}_2[\beta]$, where β is a root of $m(x) = x^{12} + x^{11} + x^{10} + x^8 + x^6 + x^4 + x^3 + x^1 + 1$, and let $\alpha = \beta^{105}$. Again, $m(x)$ is irreducible and β is primitive, but this time, we get (thanks again SageMath!)

$$m_1(x) = x^{12} + x^{11} + x^{10} + x^9 + x^5 + x^4 + x^3 + x^2 + 1. \tag{8.4.23}$$

So the expression of $m_1(x)$ in $\mathbf{F}_2[x]$ depends on how we choose to define $\mathbf{F}_q$. However, let's not lose sight of the remarkable thing about both (8.4.22) and (8.4.23): As predicted by the Orbit Theorem 8.4.13, somehow all of those factors of $\alpha = \beta^{105}$ in (8.4.21) cancel each other out, leaving only coefficients of 1 and 0. Huh!

Remark 8.4.17. Note that on the one hand, Examples 8.4.15 and 8.4.16 show that to find the minimal polynomial $m_i(x)$ of α^i in terms of α, you only need to know the order of α; see also Problems 8.4.6 and 8.4.7. On the other hand, those same examples show that to find $m_i(x)$ as an element of $\mathbf{F}_2[x]$, you need a specific construction of $\mathbf{F}_q$ and a choice of an element of the appropriate order inside $\mathbf{F}_q$; see also Problem 8.4.1.

Finally, if you think you noticed a pattern in the sizes of the orbits in Examples 8.4.15 and 8.4.16, you're right. To be precise, we have the following result, which may save you some time in computing Frobenius orbits, or at least prevent you from going too far down the rabbit hole after making a computational error.

Theorem 8.4.18. *Let $q = 2^e$, and let α be a nonzero element of $\mathbf{F}_q$. Then the size of (number of elements in) the Frobenius orbit of α divides e.*

Theorem 8.4.18 will be proven later, in Section ??.

Problems

8.4.1. Recall that any nonzero element α of $\mathbf{F}_8$ has order 7 (Problem 7.6.7). This problem shows that even though Example 8.4.15 determines the minimal polynomial $m_1(x)$ in terms of α, you can get actually different versions of $m_1(x)$ as a polynomial in $\mathbf{F}_2[x]$.

(a) Let α be an element of $\mathbf{F}_8$ such that $\alpha^3+\alpha+1=0$. Verify by direct computation that

$$(x-\alpha)(x-\alpha^2)(x-\alpha^4)=x^3+x+1. \tag{8.4.24}$$

(b) Let β be an element of $\mathbf{F}_8$ such that $\beta^3+\beta^2+1=0$. Verify by direct computation that

$$(x-\beta)(x-\beta^2)(x-\beta^4)=x^3+x^2+1. \tag{8.4.25}$$

8.4.2. prove Frobenius map is a homomorphism for q.

8.4.3. Let E be an extension of $\mathbf{F}_2$, fix some $\beta \in E$, and let

$$I=\{f(x)\in \mathbf{F}_2[x] \mid f(\beta)=0\}. \tag{8.4.26}$$

This problem proves that I is an ideal in $\mathbf{F}_2[x]$. Note that $0 \in I$ because plugging anything into the zero polynomial gives 0. As for the other parts of the definition of ideal:

(a) Prove that I is closed under addition. (Suggestion: What does it mean to say that $f_1(x), f_2(x) \in I$?)

(b) Prove that I is closed under multiplication by $h(x) \in \mathbf{F}_2[x]$.

8.4.4. Consider the extension $E=\mathbf{F}_{256}$ of $\mathbf{F}_2$, and let α be a primitive root of E. Find the product formulas for $m_1(x),\dots,m_5(x)$, as in Example 8.4.15.

8.4.5. Consider the extension $E=\mathbf{F}_{512}$ of $\mathbf{F}_2$, and let α be a primitive root of E. Find the product formulas for $m_1(x),\dots,m_5(x)$, as in Example 8.4.15.

8.4.6. Let $q=2^e$ for some $e\geq 1$, and let α be an element of $\mathbf{F}_q^\times$ of order 47. Find the product formulas for $m_1(x),\dots,m_5(x)$, as in Example 8.4.15.

8.4.7. Let $q=2^e$ for some $e\geq 1$, and let α be an element of $\mathbf{F}_q^\times$ of order 55. Find the product formulas for $m_1(x),\dots,m_5(x)$, as in Example 8.4.15.

8.5 BCH codes

The BCH Theorem, named after its discoverers Bose and Chaudhuri [citation?], and independently, Hocquenghem [citation?], provides a method for constructing a code with a guaranteed desired minimum distance.

Theorem 8.5.1 (BCH Theorem). *Let $\mathcal{C}$ be a cyclic code of length n generated by the divisor $g(x) \in \mathbf{F}_2[x]$ of $x^n - 1$. Suppose E is an extension of $\mathbf{F}_2$ such that for some $\delta \in \mathbf{N}$ and some $\alpha \in E$ with the order of α exactly equal to n, we have that*

$$0 = g(\alpha) = g(\alpha^2) = g(\alpha^3) = \cdots = g(\alpha^{\delta-1}). \tag{8.5.1}$$

Then the minimum distance d of $\mathcal{C}$ is at least δ, i.e., $d \geq \delta$.

Definition 8.5.2. A code of the form described by Theorem 8.5.1 is called a *BCH code*. If $\mathcal{C}$ is a BCH code, we call the quantity δ in Theorem 8.5.1 the *designed distance* of $\mathcal{C}$, because while the theorem assures us that $d \geq \delta$, it may actually be the fact that $d > \delta$, i.e., our code actually turns out to be better than intended (see Example 8.5.6 for an example).

We can now describe the following recipe for building BCH codes.

Algorithm 8.5.3 (Constructing a BCH code). The following recipe constructs a BCH code $\mathcal{C}$ over $\mathbf{F}_2$.

1. Choose an extension E of $\mathbf{F}_2$, and let 2^e be the order of E.
2. Choose $\alpha \in E$, and let n be the order of α. (Note that $n = 2^e - 1$ exactly when α is a primitive root of E.) Our code will have length n.
3. Choose a designed distance $\delta \in \mathbf{N}$.
4. Let $g(x)$ be the least common multiple of $m_1(x), \dots, m_{\delta-1}(x)$, i.e., remove repetitions of minimal polynomials and take the resulting product.

Definition 8.5.4. The cyclic code $\mathcal{C}$ obtained by a particular choice of E, α, and δ in Algorithm 8.5.3 is called the *BCH code given by E, α, and δ.*

It's worth noting the following strange feature of BCH codes, and actually, of cyclic codes in general: The lengths and dimensions of a good cyclic code can't be chosen randomly; instead, good cyclic codes are more like objects found in nature. Specifically, to find a good BCH code $\mathcal{C}$, you look for a generator $g(x)$ that has many roots α^i for consecutive values of i, which gives $\mathcal{C}$ its error-correcting power by Theorem 8.5.1, but also has relatively low $\deg g(x)$, which gives $\mathcal{C}$ a high dimension by Theorem 8.3.3.

Example 8.5.5. Let $E = \mathbf{F}_{128}$, the field of order $128 = 2^7$ (i.e., $e = 7$), and let α be a primitive root of E (i.e., α has order 127). We begin by constructing the corresponding BCH code $\mathcal{C}$ of designed distance $\delta = 13$.

To construct this code, we need to compute $m_i(x)$ by computing Frobenius orbits. The first such orbit is

$$\left\{\alpha^1, \alpha^2, \alpha^4, \alpha^8, \alpha^{16}, \alpha^{32}, \alpha^{64}\right\}, \tag{8.5.2}$$

and we know this orbit is complete because $\alpha^{128} = \alpha^1$. Using the notation of Example 8.4.16, we abbreviate this orbit as $\{1, 2, 4, 8, 16, 32, 64\}$.

Next, since we need $g(\alpha^i) = 0$ for $1 \le i \le 12$, the next power of α not yet accounted for is α^3, so we compute the orbit (again writing only the exponents and omitting α)

$$\{3, 6, 12, 24, 48, 96, 65\}\,. \tag{8.5.3}$$

We again finish because $\alpha^{130} = \alpha^3$, and we again see that the next power not yet accounted for is α^5. Continuing this process until we account for all i from 1 to 12, in total, we get:

$$\begin{aligned} &\{1, 2, 4, 8, 16, 32, 64\}\,, && \{3, 6, 12, 24, 48, 96, 65\}\,, \\ &\{5, 10, 20, 40, 80, 33, 66\}\,, && \{7, 14, 28, 56, 112, 97, 67\}\,, \\ &\{9, 18, 36, 72, 17, 34, 68\}\,, && \{11, 22, 44, 88, 49, 98, 69\}\,. \end{aligned} \tag{8.5.4}$$

It follows that $\mathcal{C}$ has generator polynomial

$$g(x) = m_1(x)m_3(x)m_5(x)m_7(x)m_9(x)m_{11}(x), \tag{8.5.5}$$

where, for example,

$$m_7(x) = (x - \alpha^7)(x - \alpha^{14})(x - \alpha^{28})(x - \alpha^{56})(x - \alpha^{112})(x - \alpha^{97})(x - \alpha^{67}). \tag{8.5.6}$$

We similarly see that each $m_i(x)$ ($i = 1, 3, 5, 7, 9, 11$) has degree 7, so in total, $\deg g(x) = 42$.

By Theorem 8.3.3, we see that $\dim \mathcal{C} = 127 - 42 = 85$, and so $\mathcal{C}$ is a $[127, 85, d]$ code, where $d \ge 13$. Note that since $13 = 2(6) + 1$, $\mathcal{C}$ corrects 6 errors per codeword, at least in the abstract.

Example 8.5.6. Continuing Example 8.5.5, suppose we now try to construct the corresponding BCH code of designed distance $\delta = 16$. As before, we get orbits

$$\{13, 26, 52, 104, 81, 35, 70\}\,, \qquad \{15, 30, 60, 120, 113, 99, 71\}\,. \tag{8.5.7}$$

So if

$$g(x) = m_1(x)m_3(x)m_5(x)m_7(x)m_9(x)m_{11}(x)m_{13}(x)m_{15}(x), \tag{8.5.8}$$

we see that $g(\alpha^i) = 0$ for $1 \le i \le 15$, as desired. However, looking back at (8.5.4), we see that 16, 17, and 18 have previously appeared in our orbits, which means that

$$g(\alpha^{16}) = g(\alpha^{17}) = g(\alpha^{18}) = 0 \tag{8.5.9}$$

as well. It follows that the corresponding code $\mathcal{C}$ actually has minimum distance at least 19, providing an example of a code where the actual minimum distance is greater than the designed distance $\delta = 16$.

In any case, since $\deg g(x) = 56$, $\dim \mathcal{C} = 127 - 56 = 71$, and $\mathcal{C}$ is a $[127, 71, d]$ code, where $d \ge 19$. Again, because $d = 2(9) + 1$, $\mathcal{C}$ corrects 9 errors per codeword.

Let's also look at an example where α is not a primitive element.

Example 8.5.7. Let $q = 2^{12} = 4096$, let $E = \mathbf{F}_q$ (an extension of $\mathbf{F}_2$), let β be a primitive root of E (of order 4095), and let $\alpha = \beta^{105}$. As we saw in Example 8.4.16, the order of α is $4095/105 = 39$, and the Frobenius orbits of α^i are, in abbreviated notation,

$$\begin{aligned} O_1 &= \{1, 2, 4, 8, 16, 32, 25, 11, 22, 5, 10, 20\}, \\ O_3 &= \{3, 6, 12, 24, 9, 18, 36, 33, 27, 15, 30, 21\}, \\ O_7 &= \{7, 14, 28, 17, 34, 29, 19, 38, 37, 35, 31, 23\}, \\ O_{13} &= \{13, 26\}. \end{aligned} \tag{8.5.10}$$

For completeness, let's look at *all* of the BCH codes given by E and α for odd δ between 3 and 13. Note that since the order of α is 39, the length of each such BCH code is 39, and *not* 4095 (a point that newcomers to this subject often find confusing).

- For $\delta = 3$, we have, as we always do for $\delta = 3$,

$$\begin{aligned} g(x) &= m_1(x) \\ &= (x - \alpha^1)(x - \alpha^2)(x - \alpha^4)(x - \alpha^8)(x - \alpha^{16})(x - \alpha^{32}) \\ &\quad (x - \alpha^{25})(x - \alpha^{11})(x - \alpha^{22})(x - \alpha^5)(x - \alpha^{10})(x - \alpha^{20}), \end{aligned} \tag{8.5.11}$$

  where the expansion of $m_1(x)$ comes from O_1, as we saw before in (8.4.21). For brevity, we omit the product expansions for the other $m_i(x)$ in the rest of this example, but you can obtain those expansions from the other O_i in a simiar manner.

  In any case, since O_1 is size 12, $\deg g(x) = 12$, and $\dim \mathcal{C} = 39 - 12 = 27$, which means that $\mathcal{C}$ is a $[39, 27, d]$ code for $d \geq 3$.

- For $\delta = 5$, we need to add on O_3 to ensure α^4 is in the roots of $g(x)$. However, when we do that, we see that we get α^5 for free, which means that $g(x)$ has roots α^i for $1 \leq i \leq 6$. In other words, $g(x) = m_1(x)m_3(x)$ is actually also the generator for the BCH code given by E, α, and $\delta = 7$. Furthermore, $\dim \mathcal{C} = 39 - 24 = 13$, and $\mathcal{C}$ is a $[39, 13, d]$ code for $d \geq 7$.

- To get $\delta = 9$, we add on O_7. However, when we do that, we see that $g(x)$ has roots α^i for $1 \leq i \leq 12$, which means that $g(x) = m_1(x)m_3(x)m_7(x)$ is also the generator for $\delta = 11$ and $\delta = 13$. We also see that $\dim \mathcal{C} = 39 - 36 = 3$, and $\mathcal{C}$ is a $[39, 3, d]$ code for $d \geq 13$.

- Finally, if we add on O_{13} to get

$$g(x) = m_1(x)m_3(x)m_7(x)m_{13}(x), \tag{8.5.12}$$

  we see that $g(x)$ has roots α^i for $1 \leq i \leq 38$, and the minimum distance $d \geq 39$ — yay, lots of error correction! Alas, the reason we can achieve so much error correction is that $\dim \mathcal{C} = 39 - 38 = 1$, and $\mathcal{C}$ is the $[39, 1, 39]$ code we already know: the repetition code of length 39 (Example 6.2.9). Yay?

It also happens to be the case that the "best" codes we've seen so far, $\mathcal{H}_7$ and $\mathcal{G}_{23}$, happen to be examples of BCH codes. (To be fair, that's probably just because $\mathcal{H}_7$ and $\mathcal{G}_{23}$ are among the smaller examples of many useful constructions of codes.)

Example 8.5.8. Let $E = \mathbf{F}_8$, the field of order $8 = 2^3$ (i.e., $e = 3$), and let α be a primitive root of E (i.e., α has order 7). As seen in Example 8.4.15, $m_1(x) = m_2(x)$, so for a designed distance $\delta = 3$, we take $g(x) = m_1(x) = m_2(x)$. Since $\deg g(x) = 3$, $\dim \mathcal{C} = 7 - 4 = 3$, and $\mathcal{C}$ is a $[7, 4, d]$ code, with $d \geq \delta = 3$. In fact, as in Example 8.3.6, $\mathcal{C} = \mathcal{H}_7$.

Example 8.5.9. Let $E = \mathbf{F}_{2048}$, the field of order $2048 = 2^{12}$ (i.e., $e = 12$), let β be a primitive root of E (so β has order $2047 = 23 \cdot 89$), and let $\alpha = \beta^{89}$. By Theorem 7.6.11), α has order $2047/89 = 23$, so a BCH code obtained by applying Algorithm 8.5.3 to E and α will have length 23. Computing the Frobenius orbit of α^1, and keeping in mind that the order of α is 23, in the notation of Example 8.5.5, we get

$$\{1, 2, 4, 8, 16, 9, 18, 13, 3, 6, 12\}\,. \tag{8.5.13}$$

Therefore, if we take $g(x) = m_1(x) = m_2(x) = m_3(x) = m_4(x)$, since $\deg g(x) = 11$, $\dim \mathcal{C} = 23 - 11 = 12$, and $\mathcal{C}$ is a $[23, 12, d]$ code, with $d \geq \delta = 5$ (i.e., correcting $\dfrac{5-1}{2} = 2$ errors per codeword). In fact, it can be shown that we can choose β, and therefore α, such that

$$g(x) = x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1, \tag{8.5.14}$$

as promised in Example 8.3.8. Moreover, it can be shown that the minimum distance of $\mathcal{C}$ is actually 7, making $\mathcal{C}$ a $[23, 12, 7]$, correcting $\dfrac{7-1}{2} = 3$ errors per codeword. For a proof of minimum distance 7 and much more about the Golay 23-code, see MacWilliams and Sloane Chs. 7, 16, and 20.

Remark 8.5.10. We do not discuss explicit error-correction methods here, but efficient error-correction methods do exist; see ??.

Before we can finally prove the BCH Theorem, we need the following fact from linear algebra.

Lemma 8.5.11. *Let F be a field, let $\alpha_1, \dots, \alpha_k$ be pairwise distinct elements of F (i.e., $\alpha_i \neq \alpha_j$ for $i \neq j$), and let $c_1, \dots, c_k \in F$ be nonzero elements of F. Then the columns of the matrix*

$$V = \begin{bmatrix} c_1 & c_2 & c_3 & \cdots & c_{k-1} & c_k \\ c_1\alpha_1 & c_2\alpha_2 & c_3\alpha_3 & \cdots & c_{k-1}\alpha_{k-1} & c_k\alpha_k \\ c_1\alpha_1^2 & c_2\alpha_2^2 & c_3\alpha_3^2 & \cdots & c_{k-1}\alpha_{k-1}^2 & c_k\alpha_k^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ c_1\alpha_1^{k-2} & c_2\alpha_2^{k-2} & c_3\alpha_3^{k-2} & \cdots & c_{k-1}\alpha_{k-1}^{k-2} & c_k\alpha_k^{k-2} \\ c_1\alpha_1^{k-1} & c_2\alpha_2^{k-1} & c_3\alpha_3^{k-1} & \cdots & c_{k-1}\alpha_{k-1}^{k-1} & c_k\alpha_k^{k-1} \end{bmatrix} \tag{8.5.15}$$

are linearly independent.

When all of the c_i are 1, a matrix V of the form in (8.5.15), or the transpose of such a matrix, is called a *Vandermonde matrix.*

Proof. By Corollary 5.7.5, it's enough to show that the matrix

$$V^t = \begin{bmatrix} c_1 & c_1\alpha_1 & c_1\alpha_1^2 & \cdots & c_1\alpha_1^{k-2} & c_1\alpha_1^{k-1} \\ c_2 & c_2\alpha_2 & c_2\alpha_2^2 & \cdots & c_2\alpha_2^{k-2} & c_2\alpha_2^{k-1} \\ c_3 & c_3\alpha_3 & c_3\alpha_3^2 & \cdots & c_3\alpha_3^{k-2} & c_3\alpha_3^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{k-1} & c_{k-1}\alpha_{k-1} & c_{k-1}\alpha_{k-1}^2 & \cdots & c_{k-1}\alpha_{k-1}^{k-2} & c_k\alpha_{k-1}^{k-1} \\ c_k & c_k\alpha_k & c_k\alpha_k^2 & \cdots & c_k\alpha_k^{k-2} & c_k\alpha_k^{k-1} \end{bmatrix} \tag{8.5.16}$$

satisfies $\mathrm{Null}(V^t) = \{\mathbf{0}\}$.

So suppose $\mathbf{b} = \begin{bmatrix} b_0 \\ \vdots \\ b_{k-1} \end{bmatrix} \in \mathrm{Null}(V^t)$. In that case, since the ith entry of $V^t\mathbf{b}$ is 0, we see that

$$c_ib_0 + c_ib_1\alpha_i + c_ib_2\alpha_i^2 + \cdots + c_ib_{k-2}\alpha_i^{k-2} + c_ib_{k-1}\alpha_i^{k-1} = 0. \tag{8.5.17}$$

Therefore, if we let $f(x) = b_0 + b_1x + \cdots + b_{k-1}x^{k-1}$, we have that $c_if(\alpha_i) = 0$. In fact, since $c_i \neq 0$, we have that $f(\alpha_i) = 0$ for $1 \leq i \leq k$, or in other words, f is a polynomial of degree at most $k-1$ with k distinct roots $\alpha_1, \ldots, \alpha_k$. It follows from Corollary 3.4.9 that $f(x)$ is the zero polynomial, or in other words, $\mathbf{b} = \mathbf{0}$. The lemma follows. □

Proof of BCH Theorem 8.5.1. Suppose $c(x) = c_0 + \cdots + c_{n-1}x^{n-1} \in \mathcal{C}$. Because $g(x)$ generates $\mathcal{C}$, $c(x) = q(x)g(x)$, so $c(\alpha^i) = 0$ for $1 \leq i \leq \delta - 1$. In other words,

$$c_0 + c_1\alpha^i + c_2\alpha^{2i} \cdots + c_{n-1}\alpha^{(n-1)i} = 0 \tag{8.5.18}$$

for $1 \leq i \leq \delta - 1$.

In vector form, (8.5.18) becomes

$$[1 \ \alpha^i \ \alpha^{2i} \ \ldots \ \alpha^{(n-1)i}] \begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \end{bmatrix} = 0. \tag{8.5.19}$$

Therefore, for any fixed $\mathbf{c} = \begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \end{bmatrix} \in \mathcal{C}$, (8.5.19) holds for $0 \leq i \leq \delta - 1$, so if we define a $(\delta - 1) \times n$ matrix H by

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^{2(2)} & \cdots & \alpha^{(n-1)2} \\ 1 & \alpha^3 & \alpha^{2(3)} & \cdots & \alpha^{(n-1)3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{\delta-1} & \alpha^{2(\delta-1)} & \cdots & \alpha^{(n-1)(\delta-1)} \end{bmatrix}, \tag{8.5.20}$$

then for any $\mathbf{c} \in \mathcal{C}$, we have that $H\mathbf{c} = \mathbf{0}$; in other words, $\mathcal{C} \subseteq \mathrm{Null}(H)$.

So now, let $\mathbf{c} \in \mathbf{F}_2^n$ be a vector of (Hamming) weight at most $\delta - 1$, or in other words, suppose $c_i = 0$ except possibly for $\delta - 1$ coordinates $c_{i_1}, \dots, c_{i_{\delta-1}}$. The corresponding columns $i_1, \dots, i_{\delta-1}$ of H then form a $(\delta - 1) \times (\delta - 1)$ matrix

$$H_{\mathbf{c}} = \begin{bmatrix} \alpha^{i_1} & \alpha^{i_2} & \alpha^{i_3} & \cdots & \alpha^{i_{\delta-1}} \\ \alpha^{2i_1} & \alpha^{2i_2} & \alpha^{2i_3} & \cdots & \alpha^{2i_{\delta-1}} \\ \alpha^{3i_1} & \alpha^{3i_2} & \alpha^{3i_3} & \cdots & \alpha^{3i_{\delta-1}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha^{(\delta-1)i_1} & \alpha^{(\delta-1)i_2} & \alpha^{(\delta-1)i_3} & \cdots & \alpha^{(\delta-1)i_{\delta-1}} \end{bmatrix} \tag{8.5.21}$$

such that $H_{\mathbf{c}}\mathbf{c} = \mathbf{0}$.

However, since α has order n, all powers of α between 0 and $n-1$ are different, which means that V has the form specified in Lemma 8.5.11 (with $c_j = \alpha^{i_j}$ and $\alpha_j = \alpha^{i_j}$). It follows that the columns of $H_{\mathbf{c}}$ are linearly independent, and therefore, since $H_{\mathbf{c}}\mathbf{c} = \mathbf{0}$, we must have $\mathbf{c} = \mathbf{0}$. Consequently, $\mathcal{C}$ has no nonzero codewords $\mathbf{c}$ of weight at most $\delta - 1$. The theorem follows. □

Remark 8.5.12. Fact: long BCH codes are bad.

Problems

8.5.1. Let $E = \mathbf{F}_{16}$, and let α be a primitive element of E (i.e., α has order $n = 15$).

(a) Let $\delta = 3$, and let $\mathcal{C}$ be the corresponding BCH code obtained by Algorithm 8.5.3. Find the generating polynomial $g(x)$ of $\mathcal{C}$ and $k = \dim \mathcal{C}$. You do not need to multiply $g(x)$ out over $\mathbf{F}_2$; just leave it in product form.

(b) Same, but for $\delta = 5$.

(c) Give an example of a BCH code where the actual minimal distance is greater than the designed minimal distance.

8.5.2. Let $E = \mathbf{F}_{32}$, and let α be a primitive element of E (i.e., α has order $n = 31$).

(a) Let $\delta = 5$, and let $\mathcal{C}$ be the corresponding BCH code obtained by Algorithm 8.5.3. Find the generating polynomial $g(x)$ of $\mathcal{C}$ and $k = \dim \mathcal{C}$. You do not need to multiply $g(x)$ out over $\mathbf{F}_2$; just leave it in product form.

(b) Same, but for $\delta = 7$.

8.5.3. Let $E = \mathbf{F}_{64}$, and let α be a primitive element of E (i.e., α has order $n = 63$).

(a) Let $\delta = 5$, and let $\mathcal{C}$ be the corresponding BCH code obtained by Algorithm 8.5.3. Find the generating polynomial $g(x)$ of $\mathcal{C}$ and $k = \dim \mathcal{C}$. You do not need to multiply $g(x)$ out over $\mathbf{F}_2$; just leave it in product form.

(b) Same, but for $\delta = 7$.

(c) Same, but for $\delta = 9$.

8.5.4. Let $E = \mathbf{F}_{64}$, let β be a primitive element of E, and let $\alpha = \beta^7$.

(a) What is the order of α? Explain.

(b) Let $\delta = 3$, and let $\mathcal{C}$ be the corresponding BCH code obtained by Algorithm 8.5.3 applied to E, α, and δ. Find the generating polynomial $g(x)$ of $\mathcal{C}$ and $k = \dim \mathcal{C}$. You do not need to multiply $g(x)$ out over $\mathbf{F}_2$; just leave it in product form.

(c) Same, but for $\delta = 5$. What is another name for the resulting code? Explain your answer.

8.5.5. Let $E = \mathbf{F}_{256}$, let β be a primitive element of E, and let $\alpha = \beta^{17}$.

(a) What is the order of α? Explain.

(b) Let $\delta = 3$, and let $\mathcal{C}$ be the corresponding BCH code obtained by Algorithm 8.5.3 applied to E, α, and δ. Find the generating polynomial $g(x)$ of $\mathcal{C}$ and $k = \dim \mathcal{C}$. You do not need to multiply $g(x)$ out over $\mathbf{F}_2$; just leave it in product form.

(c) Same, but for $\delta = 5$.

(d) Same, but for $\delta = 7$.

8.5.6. Let $E = \mathbf{F}_{256}$, let β be a primitive element of E, and let $\alpha = \beta^{15}$.

(a) What is the order of α? Explain.

(b) Let $\delta = 3$, and let $\mathcal{C}$ be the corresponding BCH code obtained by Algorithm 8.5.3 applied to E, α, and δ. Find the generating polynomial $g(x)$ of $\mathcal{C}$ and $k = \dim \mathcal{C}$. You do not need to multiply $g(x)$ out over $\mathbf{F}_2$; just leave it in product form.

(c) Same, but for $\delta = 5$. What is another name for the resulting code? Explain your answer.

8.5.7. Let $E = \mathbf{F}_{256}$, let β be a primitive element of E, and let $\alpha = \beta^5$.

(a) What is the order of α? Explain.

(b) Let $\delta = 5$, and let $\mathcal{C}$ be the corresponding BCH code obtained by Algorithm 8.5.3 applied to E, α, and δ. Find the generating polynomial $g(x)$ of $\mathcal{C}$ and $k = \dim \mathcal{C}$. You do not need to multiply $g(x)$ out over $\mathbf{F}_2$; just leave it in product form.

(c) Same, but for $\delta = 7$.

(d) Same, but for $\delta = 9$.

8.5.8. Let $E = \mathbf{F}_{256}$, let β be a primitive element of E, and let $\alpha = \beta^3$.

(a) What is the order of α? Explain.

(b) Let $\delta = 5$, and let $\mathcal{C}$ be the corresponding BCH code obtained by Algorithm 8.5.3 applied to E, α, and δ. Find the generating polynomial $g(x)$ of $\mathcal{C}$ and $k = \dim \mathcal{C}$. You do not need to multiply $g(x)$ out over $\mathbf{F}_2$; just leave it in product form.

(c) Same, but for $\delta = 7$.

(d) Same, but for $\delta = 9$.

8.5.9. Let $E = \mathbf{F}_{2048}$, let β be a primitive element of E, and let $\alpha = \beta^{23}$.

(a) What is the order of α? Explain.

(b) Let $\delta = 5$, and let $\mathcal{C}$ be the corresponding BCH code obtained by Algorithm 8.5.3 applied to E, α, and δ. Find the generating polynomial $g(x)$ of $\mathcal{C}$ and $k = \dim \mathcal{C}$. You do not need to multiply $g(x)$ out over $\mathbf{F}_2$; just leave it in product form.

(c) Same, but for $\delta = 7$.

(d) Same, but for $\delta = 9$.

8.6 Better codes and the burst error problem

EVERYTHING AFTER THIS POINT IN CHAPTER 8 NEEDS TO BE REWRITTEN — NOT YET READY FOR USE

(and won't be used in Spring 2023)

Going back to codes, remember that because of Theorem 6.4.13, we know that a code with a larger minimum distance corrects more errors. We can achieve a large minimum distance inefficiently by just repeating each data bit many times, but it would be more useful to achieve a large minimum distance while also being able to transmit more data bits. That is, using $[n, k, d]$ from Notation 6.4.1, the first motivating problem of this chapter can be stated as:

Motivating Problem 8.6.1. Find $[n, k, d]$ codes where both k and d are as large as possible, given n.

For our second motivating problem, suppose we have a communications channel where errors aren't evenly spread, but instead clumped together in bunches, or *bursts*. This happens with many real-life communications situations, such as the following.

- *Transmissions over the internet:* Instead of "static" or "noise" errors, you might instead see errors coming from some kind of disconnection that would brief by the human sense of time, but would nevertheless totally mangle a long string of bits.

- *Data on a hard drive:* As of 2024, data is still often stored on rotating disk at high density. A scratch on such a disk might again mangle a long string of bits.

- *DVDs:* As of 2024, people still buy or rent movies on plastic discs called DVDs, which have the same vulnerability to scratches that hard drives do.[†]

[†]I realize I sound like an alien being describing everyday human life, but this is here as a hedge for the future. Even 10 years ago, I would have used CDs for this example, but by now, in 2024, I suspect some readers will have never encountered a CD.

As always, we should try to look at some "duh" solutions to this problem, for comparison. More to the point, here are some problems any solution to the burst error problem must overcome.

- First off, we need to make our codewords very long, or else a burst error will randomize an entire codeword, making error-correction impossible.
- More subtly, in the terms of Notation 6.4.1, as a code gets very long ($n \to \infty$), it becomes very difficult to keep both the *information rate* k/n and the *error-correction rate* d/n high.

So it's probably the case that something has to give. Therefore, instead of looking for a code that can do it all, let's look at the following problem, which is not quite as difficult. (After all, this is still a math book, which means we still have the option of making the problem easier until we can actually solve it.)

Motivating Problem 8.6.2. Create an error-correcting code $\mathcal{C}$ that will correct relatively long burst errors. We allow for the possibility that $\mathcal{C}$ is not so great at correcting randomly scattered errors (i.e., has a low d/n).

Problems

8.6.1. Consider a binary linear error-correcting code $\mathcal{C}$. In this problem, we look at how burst errors interact with not just one codeword, but a *sequence* of codewords.

(a) Suppose $\mathcal{C}$ has length 10. What is the length b of the smallest burst of bit errors (consecutive string of bit errors) that can affect (create errors within) two consecutively transmitted codewords? Draw a picture to explain your answer.

(b) Now suppose $\mathcal{C}$ has length 13. What is the length b of the smallest burst of bit errors that can affect three consecutively transmitted codewords? Again, explain with a picture.

(c) Finally, suppose $\mathcal{C}$ has some arbitrary length n, and $m \geq 2$. In terms of n and m, what is the length b of the smallest burst of bit errors that can affect m consecutively transmitted codewords? Explain with a picture.

8.7 Cyclic codes over arbitrary fields

To construct the class of codes that solves Motivating Problem 8.6.2, we will first need to generalize the idea of a binary linear code (Definition 6.2.3).

Definition 8.7.1. Let $q = p^r$ be a prime power, and let $\mathbf{F}_q$ be the (unique) field of order q. A *linear code* $\mathcal{C}$ *of length* n *over* $\mathbf{F}_q$ is a subspace $\mathcal{C}$ of $\mathbf{F}_q^n$. Elements (vectors) of a linear code are still called *codewords*.

Remark 8.7.2. Since all of the codes we discuss will be linear, we will sometimes just call them codes. In case you were wondering, though, there are such things as nonlinear codes, which are just defined to be sub*sets* of $\mathbf{F}_q^n$, and in fact, you can construct nonlinear codes that have excellent communications statistics (i.e., the nonlinear equivalent of a high k/n and d/n). However, nonlinear codes are more difficult to use because we can no longer think of them in terms of linear algebra. For more on nonlinear codes, see ???.

definition of cyclic code (Definition 8.2.1) generalizes.

Definition 8.7.3. Let $\mathcal{C}$ be a code of length n over $\mathbf{F}_q$. To say that $\mathcal{C}$ is *cyclic* means that it is closed under cyclic permutation of coordinates. That is, to say that $\mathcal{C}$ is cyclic means that if $\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix}$ is in $\mathcal{C}$, then so are $\begin{bmatrix} c_{n-1} \\ c_0 \\ c_1 \\ \vdots \\ c_{n-2} \end{bmatrix}$, $\begin{bmatrix} c_{n-2} \\ c_{n-1} \\ c_0 \\ \vdots \\ c_{n-3} \end{bmatrix}$, and so on.

Notation 8.7.4. The *polynomial notation* for vectors in $\mathbf{F}_q^n$ represents the vector $\begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \end{bmatrix}$ as the polynomial

$$c_0 + c_1x + c_2x^2 + \cdots + c_{n-1}x^{n-1} \tag{8.7.1}$$

in the ring $R = \mathbf{F}_q[x]/(x^n - 1)$ (i.e., setting $x^n \equiv 1$).

Note that for the ring $\mathbf{F}_2[x]/(x^n - 1)$, instead of using the α notation (Notation 7.3.5) established earlier, we'll continue to refer to elements of $\mathbf{F}_2[x]/(x^n - 1)$ as polynomials in x, with the understanding that $x^n = 1$. The reason is that $\mathbf{F}_q$

Theorem 8.7.5. *Let $\mathcal{C}$ be a linear code of length n over $\mathbf{F}_q$. In polynomial notation, $\mathcal{C}$ is cyclic if and only if it is an ideal of the ring $\mathbf{F}_q[x]/(x^n - 1)$.*

Theorem 8.7.6. *Fix a positive integer n, and let $\mathcal{C}$ be a nonzero cyclic code of length n over $\mathbf{F}_q$, i.e., let $\mathcal{C}$ be an ideal of $\overline{R} = \mathbf{F}_q[x]/(x^n - 1)$. Then $\mathcal{C}$ is principal, or in other words, $\mathcal{C} = (g(x))$ for some $g(x) \in \mathbf{F}_q[x]$. Moreover, we can choose $g(x)$ so that $g(x)$ divides $x^n - 1$.*

8.8 Reed-Solomon codes

Reed-Solomon codes are constructed by taking the special case of BCH codes when $E = \mathbf{F}_q$ ($q = 2^e$) and thinking of each "byte" in $\mathbf{F}_q$ as a string of e "bits" in $\mathbf{F}_2$. As we will soon see, Reed-Solomon codes are not particularly good codes if errors are randomly scattered in a transmission, but they provide an effective and practical solution to the burst error problem (Motivating Problem 8.6.2).

We now give the recipe for constructing a Reed-Solomon code. First, however, recall that for $q = 2^e$, $\mathbf{F}_q = \mathbf{F}_2[x]/(m(x))$, where $m(x)$ is a degree e polynomial that is irreducible over $\mathbf{F}_2$. In practice, that means that elements of $\mathbf{F}_q$ can be represented as polynomials

$$a_0 + a_1x + a_2x^2 + \cdots + a_{e-1}x^{e-1} \in \mathbf{F}_2[x], \tag{8.8.1}$$

which in turn can be represented as a "byte" $(a_0, \dots, a_{e-1})$ of e bits. Therefore, when we construct a Reed-Solomon code $\mathcal{C}$, we will need to distinguish between $\mathbf{F}_q$-lengths and dimensions, which we denote by the capital letters N and K, and $\mathbf{F}_2$-lengths and dimensions, which we denote by lower-case letters n and k. Note that since each "byte" (element of $\mathbf{F}_q$) is made from e "bits" (elements of $\mathbf{F}_2$), we have that $n = Ne$ and $k = Ke$. We similarly use D and d to denote the minimum $\mathbf{F}_q$-distance and $\mathbf{F}_2$-distance of $\mathcal{C}$.

Algorithm 8.8.1 (Constructing a Reed-Solomon code)**.** The following recipe constructs a Reed-Solomon code $\mathcal{C}$ over a field $\mathbf{F}_q$.

1. Choose $q = 2^e$ for some $e \in \mathbf{N}$, and let $E = \mathbf{F}_q$.
2. Choose $\alpha \in E$ with order N. (If $N = q - 1$, then α is also a primitive element of E.)
3. Choose a designed distance $\Delta \in \mathbf{N}$.
4. Since the elements α^i are all in $\mathbf{F}_q$, $m_i(x) = (x - \alpha^i)$ is the minimal polymomial of α^i over $\mathbf{F}_q$. Therefore, let

$$g(x) = (x - \alpha^1)(x - \alpha^2)(x - \alpha^3)\dots(x - \alpha^{\Delta-1}). \tag{8.8.2}$$

Then $\mathcal{C}$ is the cyclic code generated by $g(x)$.

First, the bad news: If $\mathcal{C}$ is a Reed-Solomon code, and we look at $\mathcal{C}$ as a code over $\mathbf{F}_2$ with randomly scattered bit errors, then $\mathcal{C}$ will generally not be so great as a code. The problem is that the bit length $n = Ne$, or e times the byte length, but each randomly scattered bit error can potentially cause a byte error, which effectively makes $d = D$. The correction rate is therefore effectively

$$\frac{d}{n} = \frac{D}{Ne} = \frac{1}{e}\left(\frac{D}{N}\right), \tag{8.8.3}$$

or in other words, the random-bit-error-correction rate is reduced by a factor of e from the random-byte-error-correction rate.

The good news is that Reed-Solomon codes are very good at correcting burst bit errors, as shown by the following example.

Example 8.8.2. Let $E = \mathbf{F}_{512}$, the field of order $512 = 2^9$ (i.e., $e = 9$), and let α be a primitive root of E (so in particular, $\alpha^{511} = 1$). Let $\mathcal{C}$ be the Reed-Solomon code of designed distance $\Delta = 201$. We see that

$$g(x) = \prod_{i=1}^{200}(x - \alpha^i), \tag{8.8.4}$$

a polynomial of degree 200, and since $N = 511$, the "byte" dimension of $\mathbf{C}$ is $K = 311$. Since $e = 9$, we also see that $n = 9(511) = 4599$ and $k = 9(311) = 2799$. Futhermore, since $\Delta = 201 = 2(100) + 1$, we can correct 100 randomly occurring"byte" errors in any codeword, but as mentioned above, we are also only guaranteed to be able to correct 100 randomly occurring"bit" errors, which is not a great ratio when compared to the bit length of 4599.

However, suppose that our bit errors occur not randomly, but instead, in a single burst of length b. We see from Figure 8.8.1, below, that if $b \leq 99(9)+1$ (but not if $b = 99(9)+2$), then that burst will be contained within 100 "bytes", and can therefore be corrected. It follows that within any codeword of length 4599, we can correct any single burst of length at most $99(9) + 1 = 892$.

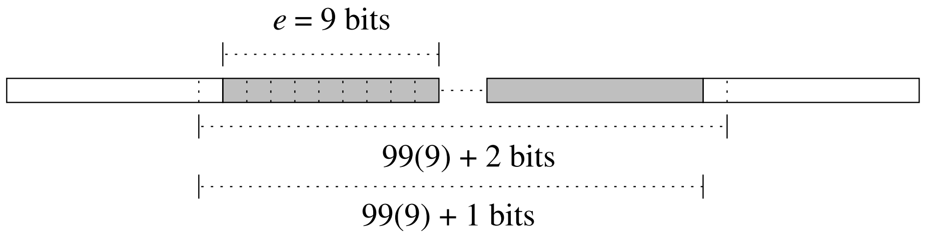

Figure 8.8.1: The longest correctible burst error

There's nothing special about the parameters of Example 8.8.2, so exactly the same argument results in the following theorem.

Theorem 8.8.3 (Reed-Solomon burst correction). *Let $\mathcal{C}$ be a Reed-Solomon code constructed over $\mathbf{F}_q$ ("bytes") with $q = 2^e$ and designed $\mathbf{F}_q$-distance $\Delta = 2T + 1$. Then any burst ("bit") error of length at most $b = (T - 1)e + 1$ within a single codeword can be corrected.*

Proof. Problem 8.8.2. □

Problems

8.8.1. Let $E = \mathbf{F}_{256}$, and let α be a primitive root of unity of E (i.e., $N = 255$).

(a) For the Reed-Solomon code $\mathcal{C}$ with designed $\mathbf{F}_q$-distance $\Delta = 15$, find the generator polynomial $g(x)$, the $\mathbf{F}_q$-dimension K, the $\mathbf{F}_2$-dimension k, and the maximum burst error correction length b.

(b) Same, but for $\Delta = 19$.

(c) Same, but for $\Delta = 33$.

8.8.2. longest burst that can be corrected.

8.9 Error correction in BCH codes

To be written later: how to correct errors in BCH codes

Chapter 9

The Discrete Fourier Transform

9.1 Digital signal processing

motivating problem

not necessary, but just in case you're interested.

continuous solution: Fourier series with sines and cosines

discrete solution: Fourier series with $e^{2\pi i n x}$.

9.2 Complex numbers and roots of unity

We have referred to complex numbers occasionally so far, but in this chapter, we'll need to understand them in much greater detail. One way to think of complex numbers is that they consist of all expressions of the form $a + bi$, where a and b are real numbers, and i is a new (to-be-defined) symbol:

$$\mathbf{C} = \{a + bi \mid a, b \in \mathbf{R}\}. \tag{9.2.1}$$

Addition and multiplication of elements of $\mathbf{C}$ is then defined to be multiplication of "polynomials in i", with the additional rule that $i^2 = -1$. In other words, $\mathbf{C}$ is precisely the quotient ring $\mathbf{R}[x]/(x^2 + 1)$! For example:

$$\begin{aligned}(3 + 7i)(4 - 5i) &= 3(4) - 3(5i) + (7i)(4) - (7i)(5i) \\ &= 12 - 15i + 28i - 35(i^2) \\ &= 12 - 15i + 28i + 35 \\ &= 47 + 13i.\end{aligned} \tag{9.2.2}$$

Another way to look at complex numbers that will be very useful to us is to draw them in the *complex plane*. As shown in Figure 9.2.1, the idea is that we draw $a + bi$ as the point (a, b) in the xy-plane. This ties naturally into two other operations one can do on a complex number $a + bi$, namely, the *modulus*, or *absolute value*, of $a + bi$:

$$|a + bi| = \sqrt{a^2 + b^2}; \tag{9.2.3}$$

and the (complex) *conjugate* of $a + bi$:

$$\overline{a + bi} = a - bi. \tag{9.2.4}$$

Note that if $a \in \mathbf{R}$, then the modulus $|a + 0i| = \sqrt{a^2}$ is the usual real absolute value of a; in other words, the modulus is a generalization of the real absolute value of a number.

The modulus and the conjugate have the following algebraic properties.

Theorem 9.2.1. *For $z, w \in \mathbf{C}$, we have that*

1. $|zw| = |z|\,|w|$;
2. $z\overline{z} = |z|^2$;
3. $|z| \geq 0$, *and* $|z| = 0$ *if and only if* $z = 0$; *and*
4. *If* $z \neq 0$, *then*
$$z\left(\frac{\overline{z}}{|z|^2}\right) = 1. \tag{9.2.5}$$

We'll see that the above property that is most useful to us is (9.2.5).

Proof. Problem 9.2.1. □

The modulus and the complex conjugate also have natural geometric interpretations in the complex plane, as shown in Figure 9.2.1: The modulus of $a + bi$ can be pictured as the distance from the point $a + bi$ to the origin, and the conjugate can be pictured as the mirror image of $a + bi$ in the real axis (x-axis).

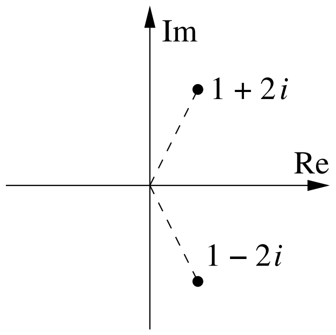

Figure 9.2.1: Two conjugate complex numbers

Taking things up a notch, we will take it as a fact[*] that for any $z \in \mathbf{C}$, the *complex exponential* e^z can be defined starting with *Euler's formula*

$$e^{ix} = \cos x + i \sin x, \tag{9.2.6}$$

[*]A fact proven in analysis (the theory of calculus), maybe even in a second course. Did I mention that analysis is difficult?

where x is any real number, and cos and sin are calculated in radians (as any good calculus student knows). As special cases of (9.2.6), we have

$$e^{\pi i} = -1, \qquad\qquad e^{2\pi i} = 1. \tag{9.2.7}$$

(The part of (9.2.7) is sometimes known as *Euler's identity.*) Since we will also take it as fact that the usual exponential law

$$e^{z+w} = e^z e^w \tag{9.2.8}$$

holds for all $z, w \in \mathbf{C}$, we can extend Euler's formula (9.2.6) to give

$$e^{x+yi} = e^x e^{iy} = e^x \cos y + (e^x \sin y)i \tag{9.2.9}$$

for any $x + yi \in \mathbf{C}$, where e^x is the usual real exponential function.

You may remember the identity known as *de Moivre's theorem* from trigonometry or precalculus. From our point of view, de Moivre's theorem is a special case of applying the usual exponential laws to complex exponentials, because for $n \in \mathbf{N}$ and $x \in \mathbf{R}$,

$$(\cos x + i \sin x)^n = (e^{ix})^n = e^{inx} = \cos(nx) + i \sin(nx). \tag{9.2.10}$$

Again, we'll take it as given that this all works; I hope you find it plausible, or at least internally consistent.

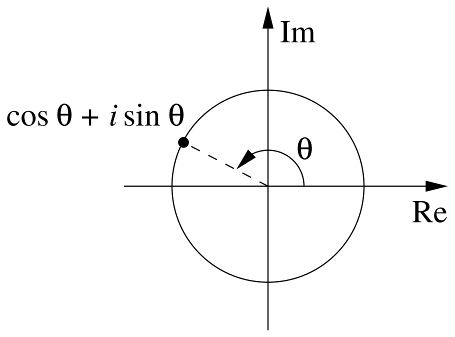

Figure 9.2.2: Complex exponentials on the unit circle

Complex exponentials also have the following geometric interpretation. Drawing (9.2.6) in the complex plane, we see that the complex numbers of the form $e^{i\theta}$, $\theta \in \mathbf{R}$, are precisely the points on the unit circle, with θ being the angle of $\cos\theta + i \sin\theta$ in the usual sense; see Figure 9.2.2. More generally, for an arbitrary complex number $a + bi \neq 0$, if $r = |a + bi|$ is the modulus of $a + bi$, then $\dfrac{a+bi}{r}$ is a point $e^{i\theta}$ on the unit circle, which means that

$$a + bi = re^{i\theta}. \tag{9.2.11}$$

Here θ is called the *argument* of $a + bi$, and (9.2.11) is called the *modulus-argument* representation of $a+bi$. The argument θ can be interpreted as the angle (again, in radians) going from the positive x-axis to the ray from the origin through $a+bi$; again, see Figure 9.2.2. Note that much like, for example, angles in polar coordinates, changing the argument of a complex number z from θ to $\theta+2\pi$ does not change z at all; in other words, the argument of z is defined only up to adding multiples of 2π.

In any case, we can now explain the reason we need complex numbers for the Discrete Fourier Transform.

Definition 9.2.2. For a positive integer N, we define the *natural primitive Nth root of unity* in $\mathbf{C}$ to be

$$\omega_N = e^{2\pi i/N}. \tag{9.2.12}$$

When N is fixed, or the context is otherwise clear, we abbreviate ω_N as ω.

The complex number ω_N is called an Nth root of unity because the rules of exponents (which, again, really do apply to complex exponentials) imply that

$$\omega_N^N = \left(e^{2\pi i/N}\right)^N = e^{2\pi i} = 1. \tag{9.2.13}$$

The "primitive" part of the name comes from the fact that there are other solutions $z \in \mathbf{C}$ for $z^N = 1$, but these other solutions are precisely the powers of ω_N. That is:

Theorem 9.2.3. *Let N be a positive integer, and let $\omega = \omega_N = e^{2\pi i/N}$. The zeros of the polynomial $z^N - 1$ (i.e., the solutions to $z^N = 1$) are precisely the powers $1, \omega, \omega^2, \dots, \omega^{N-1}$ of ω.*

Proof. Problem 9.2.2. □

Now, real talk here, if you haven't spent much time thinking about complex numbers or the complex exponential $e^{2\pi ix}$, you may be tempted to try to understand ω_N in terms of sines and cosines. Well, please don't — that approach creates a big mess and doesn't end up helping you solve problems. Just keep in mind two things about ω_N, one of which we just discussed, and the other of which we will prove in the next section.

Key properties of ω_N

Let N be a positive integer, and let $\omega = \omega_N = e^{2\pi i/N}$.

1. The solutions to $z^N = 1$ are precisely the powers $1, \omega, \omega^2, \dots, \omega^{N-1}$ (Theorem 9.2.3).

2. We have that
$$1 + \omega + \cdots + \omega^{N-1} = 0. \tag{9.2.14}$$

Later, we'll prove (9.2.14) as a particular case of Lemma 9.3.4, but you can also find a geometric justification in Figure 9.2.3. The idea there is, since the Nth roots of unity form a regular N-gon with center at $0 + 0i$, by symmetry, the average value of those points must be 0, which means that their sum must also be 0. (For a precise version of this idea, see Problem 9.2.3.)

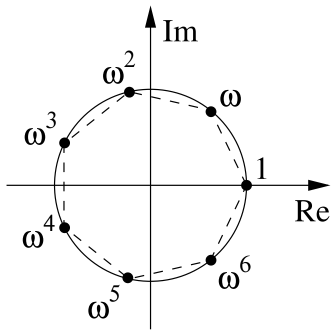

Figure 9.2.3: Nth roots of unity ($N = 7$)

In any case, for our purposes, once we have the above properties, we can pretty much treat ω as a black box and not even really think about the complex numbers again. Most notably, everything else we do with the Discrete Fourier Transform, and later, with the Fast Fourier Transform, can be done with (for example) ω a primitive element of a finite field. See, for example, Problem 9.3.3, or see (reference???).

Problems

9.2.1. Let $z = a + bi$ and $w = c + di$ be complex numbers.

(a) Prove that $|zw| = |z|\,|w|$.

(b) Prove that $|z|^2 = z\overline{z}$.

(c) Prove that $|z| \geq 0$, and $|z| = 0$ if and only if $z = 0$. (Suggestion: You can use the fact that if $a \in \mathbf{R}$, then $a^2 \geq 0$, and $a^2 = 0$ if and only if $a = 0$.)

(d) Prove that if $z \neq 0$, then $z\left(\dfrac{\overline{z}}{|z|^2}\right) = 1$.

9.2.2. The goal of this problem is to find all Nth roots of unity in $\mathbf{C}$, i.e., all solutions $z \in \mathbf{C}$ for $z^N = 1$. Let $\omega = \omega_N = e^{2\pi i/N}$ be the natural primitive Nth root of unity.

(a) Prove that for any $k \in \mathbf{Z}$, $z = \omega^k$ is a solution to $z^N = 1$.

(b) Explain why the solution set to $z^N = 1$ is precisely $\left\{\omega^k \mid k \in \mathbf{Z}\right\}$. (Suggestion: Consider $z^N - 1$ and Section 3.4.)

9.2.3. prove Nth roots sum by symmetry (multiplication by ω).

9.3 Signals

To give some context and motivation for the *Discrete Fourier Transform* (DFT), we define the following class of functions.

Definition 9.3.1. Fix $N \in \mathbf{N}$. We define a *signal* to be a function $f : \mathbf{Z}/(N) \to \mathbf{C}$, or in other words, a complex-valued function with domain $\mathbf{Z}/(N)$. Note that a signal f is defined by its N values $f(0), \dots, f(N-1) \in \mathbf{C}$, so we sometimes represent a signal f in vector form as $\begin{bmatrix} f(0) \\ \vdots \\ f(N-1) \end{bmatrix}$.

The following example of a signal on $\mathbf{Z}/(N)$ turns out to be a crucial one.

Definition 9.3.2. Fix $N \in \mathbf{N}$, and let $\omega = e^{2\pi i/N}$ be the natural primitive Nth root of unity in $\mathbf{C}$. We define the *basic trigonometric signals* $e_k : \mathbf{Z}/(N) \to \mathbf{C}$ by the formula

$$e_k(n) = \omega^{kn}. \tag{9.3.1}$$

We can also represent the basic trigonometric signals e_k in vector form as $\begin{bmatrix} 1 \\ \omega^k \\ \vdots \\ \omega^{(N-1)k} \end{bmatrix}$.

If you write out a few examples of the e_k, you'll see that when k divides N, the signal e_k has period N/k, with a slightly more complicated situation when k doesn't divide N; see Problem 9.3.1.

Remark 9.3.3. The reason we call the e_k "trigonometric" is that if you write out (9.3.1) in terms of sines and cosines, you get:

$$e_k(n) = e^{2\pi ikn/N} = \cos\left(\left(\frac{2\pi k}{N}\right) n\right) + i \sin\left(\left(\frac{2\pi k}{N}\right) n\right) \tag{9.3.2}$$

Now, I can't emphasize strongly enough, you're much better off in terms of doing algebra if you write everything in terms of ω. Just keep in mind (or in your heart?) that each function e_k is a discrete version of a complexified sine/cosine function of period dividing N.

In some sense, the property in the following lemma explains why the basic trigonometric signals are so important to us.

Lemma 9.3.4 (Orthogonality Lemma). *Fix $N \in \mathbf{N}$ and let $\omega = \omega_N = e^{2\pi i/N}$ be the natural primitive Nth root of unity in $\mathbf{C}$. For $t \in \mathbf{Z}/(N)$, we have:*

$$\sum_{k=0}^{N-1} \omega^{tk} = \begin{cases} N & \text{if } t = 0 \text{ (mod } N\text{)}, \\ 0 & \text{otherwise.} \end{cases} \tag{9.3.3}$$

Proof. See Problem 9.3.2. □

See Section 9.6 for an explanation of why we call Lemma 9.3.4 the Orthogonality Lemma. For now, suffice it to say that because of numerous applications coming from signal processing (see Section 9.1), we ask:

Motivating Problem 9.3.5. Fix $N \in \mathbf{N}$. How can we express any signal on $\mathbf{Z}/(N)$ as a linear combination of the basic trigonometric signals e_k, $0 \leq k \leq N-1$?

Problems

9.3.1. Write out the e_k in terms of vectors ($N = 6$, $k = 0, 1, 2, 3, 4, 5$).

9.3.2. (*Proves Lemma 9.3.4*) Fix $N \in \mathbf{N}$, and let $\omega = e^{2\pi i/N}$. Let $f(x) = x^N - 1$.

(a) Explain why

$$f(x) = x^N - 1 = (x-1)\left(\sum_{k=0}^{N-1} x^k\right). \tag{9.3.4}$$

(Suggestion: Try writing out the sum as $1 + x + \dots$.)

(b) Explain why for any $t \in \mathbf{Z}/(N)$, $f(\omega^t) = 0$.

(c) Prove that for any $t \in \mathbf{Z}/(N)$,

$$\sum_{k=0}^{N-1} \omega^{tk} = \begin{cases} N & \text{if } t = 0 \pmod{N}, \\ 0 & \text{if } t \neq 0 \pmod{N}. \end{cases} \tag{9.3.5}$$

Suggestion: Use the fact that $a, b \in \mathbf{C}$, if $ab = 0$, then either $a = 0$ or $b = 0$.

9.3.3. (orthogonality when ω is a primitive element of a finite field)

9.4 The Discrete Fourier Transform

Definition 9.4.1. Fix $N \in \mathbf{N}$, let $\omega = e^{2\pi i/N}$ be the natural primitive Nth root of unity in $\mathbf{C}$, and let $f : \mathbf{Z}/(N) \to \mathbf{C}$ be a signal. We define the *Discrete Fourier Transform*, or *DFT*, of f to be the function $\hat{f} : \mathbf{Z}/(N) \to \mathbf{C}$ given by

$$\hat{f}(k) = \frac{1}{N} \sum_{n=0}^{N-1} f(n)\omega^{-kn}. \tag{9.4.1}$$

Note that while the DFT of a signal f is, by definition, itself a signal, it's more helpful to think of $\hat{f}(k)$ as the *spectrum* of f, as roughly speaking, $\hat{f}(k)$ measures the strength of the part of f that has "frequency k". (See Theorem 9.4.4, below, for a justification for this idea.)

Remark 9.4.2. Looking at (9.4.1) for $k = 0, 1, 2, 3$, we get

$$\begin{aligned}
\hat{f}(0) &= \frac{1}{N}(f(0) + f(1) + f(2) + \cdots + f(N-1)),\\
\hat{f}(1) &= \frac{1}{N}(f(0) + \omega^{-1}f(1) + \omega^{-2}f(2) + \cdots + \omega^{-(N-1)}f(N-1)),\\
\hat{f}(2) &= \frac{1}{N}(f(0) + \omega^{-2}f(1) + \omega^{-2(2)}f(2) + \cdots + \omega^{-2(N-1)}f(N-1)),\\
\hat{f}(3) &= \frac{1}{N}(f(0) + \omega^{-3}f(1) + \omega^{-3(2)}f(2) + \cdots + \omega^{-3(N-1)}f(N-1)).
\end{aligned} \tag{9.4.2}$$

In fact, if we write out (9.4.1) for $k = 0, \dots, N-1$ and express the result in terms of matrix-vector multiplication, we get

$$\begin{bmatrix} \hat{f}(0) \\ \vdots \\ \hat{f}(N-1) \end{bmatrix} = \frac{1}{N} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \dots & \omega^{-(N-1)} \\ 1 & \omega^{-2} & \omega^{-2(2)} & \dots & \omega^{-2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{-(N-1)} & \omega^{-2(N-1)} & \dots & \omega^{-(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} f(0) \\ \vdots \\ f(N-1) \end{bmatrix}. \tag{9.4.3}$$

Consequently, the DFT itself is an $O(N^2)$ process, just like multiplying an $N \times N$ matrix times a column vector of length N.

Our next goal is to show that the DFT is invertible, and that its inverse is a process that is very similar to the DFT itself. We therefore optimistically make the following definition.

Definition 9.4.3. Let $\hat{f} : \mathbf{Z}/(N) \to \mathbf{C}$ be a spectrum function. The *inverse DFT* of $\hat{f}$ is defined to be

$$\sum_{k=0}^{N-1} \hat{f}(k)\omega^{kn}. \tag{9.4.4}$$

Note that this is the same transformation as the DFT (Definition 9.4.1), but with a sign change and without the factor of $\dfrac{1}{N}$.

We now prove that the inverse DFT works as advertised.

Theorem 9.4.4 (Inversion Theorem)**.** *Fix $N \in \mathbf{N}$, let $\omega = e^{2\pi i/N}$ be the natural primitive Nth root of unity in $\mathbf{C}$, and let $f : \mathbf{Z}/(N) \to \mathbf{C}$ be a signal. If $\hat{f}$ is the DFT of f, then*

$$f(n) = \sum_{k=0}^{N-1} \hat{f}(k)\omega^{kn}. \tag{9.4.5}$$

In other words, taking the inverse transform, which again is basically the DFT with a sign change, recovers our original signal f.

Proof. Considering the right-hand side of (9.4.5) with the variable x in place of the variable n, we see that

$$\begin{aligned}\sum_{k=0}^{N-1} \hat{f}(k)\omega^{kx} &= \frac{1}{N}\sum_{k=0}^{N-1}\left(\sum_{n=0}^{N-1} f(n)\omega^{-nk}\right)\omega^{kx} \\ &= \frac{1}{N}\sum_{n=0}^{N-1} f(n)\sum_{k=0}^{N-1}\omega^{k(x-n)} && (*) \\ &= \frac{1}{N}(f(x)N) && (**) \\ &= f(x),\end{aligned} \tag{9.4.6}$$

where (*) follows by switching the order of summation, and (**) follows by the Orthogonality Lemma 9.3.4. The theorem follows. □

Remark 9.4.5. Extending Remark 9.4.2, we see that Theorem 9.4.4, written in terms of matrix-vector multiplication, gives:

$$\begin{bmatrix} f(0) \\ \vdots \\ f(N-1)\end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega^1 & \omega^2 & \dots & \omega^{(N-1)} \\ 1 & \omega^2 & \omega^{2(2)} & \dots & \omega^{(N-1)2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{(N-1)} & \omega^{2(N-1)} & \dots & \omega^{(N-1)(N-1)}\end{bmatrix}\begin{bmatrix} \hat{f}(0) \\ \vdots \\ \hat{f}(N-1)\end{bmatrix} \tag{9.4.7}$$

In other words, if we form a matrix $\hat{T}$ whose kth column is the basic trigonometric signal e_k, then $\hat{T}\begin{bmatrix} \hat{f}(0) \\ \vdots \\ \hat{f}(N-1)\end{bmatrix} = \begin{bmatrix} f(0) \\ \vdots \\ f(N-1)\end{bmatrix}$. Since $\hat{T}\begin{bmatrix} \hat{f}(0) \\ \vdots \\ \hat{f}(N-1)\end{bmatrix}$ is the linear combination of the columns of $\hat{T}$ with coefficients taken from $\begin{bmatrix} \hat{f}(0) \\ \vdots \\ \hat{f}(N-1)\end{bmatrix}$, we see that the $\hat{f}$ are the coefficients that express our original signal f as a linear combination of the basic trigonometric signals e_k, solving Motivating Problem 9.3.5.

Put another way, since (9.4.7) holds for every signal f, it must actually be the case that

$$\left(\frac{1}{N}\begin{bmatrix}1 & 1 & 1 & \dots & 1\\ 1 & \omega^{-1} & \omega^{-2} & \dots & \omega^{-(N-1)}\\ 1 & \omega^{-2} & \omega^{-2(2)} & \dots & \omega^{-(N-1)2}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & \omega^{-(N-1)} & \omega^{-2(N-1)} & \dots & \omega^{-(N-1)(N-1)}\end{bmatrix}\right)^{-1} = \begin{bmatrix}1 & 1 & 1 & \dots & 1\\ 1 & \omega^{1} & \omega^{2} & \dots & \omega^{(N-1)}\\ 1 & \omega^{2} & \omega^{2(2)} & \dots & \omega^{(N-1)2}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & \omega^{(N-1)} & \omega^{2(N-1)} & \dots & \omega^{(N-1)(N-1)}\end{bmatrix}. \tag{9.4.8}$$

Problems

9.4.1. Consider the formula (9.4.1) of Definition 9.4.1.

(a) For $N = 6$, expand each of the sums $\hat{f}(0)$, $\hat{f}(1)$, $\hat{f}(2)$, $\hat{f}(3)$, $\hat{f}(4)$, and $\hat{f}(5)$, in the style of (9.4.2) in Remark 9.4.2, but write out each sum completely instead of using "dot dot dot"s. For each ω^m that appears in your answer, reduce the exponent m (mod 6) such that $-5 \le m \le 0$.

(b) Same, but $N = 8$, and expand $\hat{f}(0), \dots, \hat{f}(7)$ with ω^m such that $-7 \le m \le 0$.

(c) Same, but $N = 9$, and expand $\hat{f}(0), \dots, \hat{f}(8)$ with ω^m such that $-8 \le m \le 0$.

(d) Same, but $N = 12$, and expand $\hat{f}(0), \dots, \hat{f}(11)$ with ω^m such that $-11 \le m \le 0$.

9.4.2. Consider the formula (9.4.1) of Definition 9.4.1. For the given N:

- Write the forward DFT as a matrix, as in Remark 9.4.2, reducing each ω^m so that $-(N-1) \le m \le 0$;
- Write the inverse DFT as a matrix, as in Remark 9.4.5, reducing each ω^m so that $0 \le m \le N-1$; and
- Multiply the two matrices and verify that you get the identity.

(a) $N = 4$.

(b) $N = 6$.

(c) $N = 8$.

(d) $N = 9$.

9.5 Convolution

The DFT, like all Fourier transforms, has many remarkable properties that we won't have time to get into. The one we will look at is its interaction with the property known as *convolution.*

Definition 9.5.1. Let $f, g : \mathbf{Z}/(N) \to \mathbf{C}$ be signals. We define the *convolution* of f and g to be the signal $f * g : \mathbf{Z}/(N) \to \mathbf{C}$ defined by

$$(f * g)(n) = \frac{1}{N} \sum_{t=0}^{N-1} f(n-t)g(t). \tag{9.5.1}$$

Unless you've seen Fourier series or Fourier analysis before, you probably think Definition 9.5.1 is strange and unmotivated. However, it can be justified in terms of two important properties; the first has to do with multiplication of polynomials.

Theorem 9.5.2. *Let $f, g : \mathbf{Z}/(N) \to \mathbf{C}$ be signals. Then in the ring $\mathbf{C}[x]/(x^N - 1)$, we have that*

$$\left(\frac{1}{N}\sum_{k=0}^{N-1} f(k)x^k\right)\left(\frac{1}{N}\sum_{m=0}^{N-1} g(m)x^m\right) = \frac{1}{N}\sum_{n=0}^{N-1} (f * g)(n)x^n. \tag{9.5.2}$$

In other words, after scaling appropriately, the convolution $f * g$ gives the coefficients of the product of the polynomials whose coefficients are given by f and g, as long as we are working $\pmod{(x^N - 1)}$ (i.e., taking $x^N = 1$).

Before proving Theorem 9.5.2, we need the following lemma.

Lemma 9.5.3 (Substitution Lemma)**.** *Let $h : \mathbf{Z}/(N) \to \mathbf{C}$ be a complex-valued function on $\mathbf{Z}/(N)$. Then for any $t \in \mathbf{Z}/(N)$, we have that*

$$\sum_{k=0}^{N-1} h(k) = \sum_{n=0}^{N-1} h(n-t). \tag{9.5.3}$$

Note that we can think of (9.5.3) as justifying "summation by substitution $k = n - t$", in analogy with integration by substitution. Note also that the expression $n - t$ appearing in (9.5.3) must be computed mod N to make sense (otherwise we'd end up with negative numbers or numbers bigger than $N - 1$).

Proof. See Problem 9.5.1. □

Proof of Theorem 9.5.2. Replacing m with t in (9.5.2), we have that

$$\begin{aligned}
\left(\frac{1}{N}\sum_{k=0}^{N-1} f(k)x^k\right)\left(\frac{1}{N}\sum_{t=0}^{N-1} g(t)x^t\right) &= \frac{1}{N^2}\sum_{t=0}^{N-1}\sum_{k=0}^{N-1} f(k)g(t)x^{k+t} \\
&= \frac{1}{N^2}\sum_{t=0}^{N-1}\sum_{n=0}^{N-1} f(n-t)g(t)x^{(n-t)+t} \qquad (*) \\
&= \frac{1}{N}\sum_{n=0}^{N-1}\left(\frac{1}{N}\sum_{t=0}^{N-1} f(n-t)g(t)\right)x^n \\
&= \frac{1}{N}\sum_{n=0}^{N-1}(f*g)(n)x^n,
\end{aligned} \tag{9.5.4}$$

where we switch the order of summation twice, and we use the substitution $k = n - t$ in (*), by the Substitution Lemma 9.5.3. The theorem follows. □

The other key property of convolution is that the DFT turns convolution into pointwise multiplication. More precisely, we have the following theorem.

Theorem 9.5.4. *Let $f, g : \mathbf{Z}/(N) \to \mathbf{C}$ be signals. We have that*

$$\widehat{(f*g)}(k) = \hat{f}(k)\hat{g}(k). \tag{9.5.5}$$

Proof of Theorem 9.5.4. We have that

$$\begin{aligned}
\widehat{(f*g)}(k) &= \frac{1}{N}\sum_{n=0}^{N-1}(f*g)(n)\omega^{-nk} && (*) \\
&= \frac{1}{N}\sum_{n=0}^{N-1}\frac{1}{N}\left(\sum_{t=0}^{N-1} f(n-t)g(t)\right)\omega^{-nk} && (**) \\
&= \frac{1}{N^2}\sum_{t=0}^{N-1}\sum_{u=0}^{N-1} f(u)g(t)\omega^{-(u+t)k} && (***) \\
&= \left(\frac{1}{N}\sum_{u=0}^{N-1} f(u)\omega^{-uk}\right)\left(\frac{1}{N}\sum_{u=0}^{N-1} g(t)\omega^{-tk}\right) \\
&= \hat{f}(k)\hat{g}(k),
\end{aligned} \tag{9.5.6}$$

where (*) is the definition of the DFT of $f * g$ (Definition 9.4.1), (**) is the definition of $f * g$ (Definition 9.5.1), and (***) is the substitution $u = n - t$ (Substitution Lemma 9.5.3). The theorem follows. □

The significance of Theorem 9.5.4 is that it points the way towards a faster algorithm for multiplying two numbers. Because of carrying (!), among other complications, multiplying numbers is too hard a problem to discuss in this book, so instead, let's consider the following toy version of the same problem.

Motivating Problem 9.5.5. Compute the product of two polynomials of degree $\leq N_0$ whose coefficients are given by $f(n)$ and $g(n)$.

If $N \geq 2N_0 + 1$, multiplying polynomials of degree $\leq N_0$ reduces to multiplying polynomials in $\mathbf{C}[x]/(x^N - 1)$, so Motivating Problem 9.5.5 reduces to the following problem:

Motivating Problem 9.5.6. Compute the product of two polynomials in $\mathbf{C}[x]/(x^N - 1)$ whose coefficients are given by $f(n)$ and $g(n)$. In other words, by Theorem 9.5.2, given two signals $f(n)$ and $g(n)$, compute the convolution $(f * g)(n)$.

By our previous discussion, we have the following naive algorithm for computing $(f * g)(n)$.

Naive Algorithm 9.5.7. Suppose we have two signals $f, g : \mathbf{Z}/(N) \to \mathbf{C}$.

1. Compute the DFTs $\hat{f}(k)$ and $\hat{g}(k)$.
2. For all $k \in \mathbf{Z}/(N)$, let $\hat{h}(k) = \hat{f}(k)\hat{g}(k)$.
3. Compute the inverse DFT $h(n)$ of $\hat{h}(k)$.

Now, the good news is that computing the product/convolution $f * g(n)$ by the standard method is an $O(N^2)$ operation, whereas Step 2 of Naive Algorithm 9.5.7 is an $O(N)$ operation! However, the bad news is that Steps 1 and 3 of Naive Algorithm 9.5.7 are $O(N^2)$ operations, which would seem to make this "shortcut" not very useful. However, if we can replace the direct DFT with something that runs in less time than $O(N^2)$, then since the DFT is now the bottleneck in multiplication, we will then get a faster algorithm for multiplication. We therefore come to the following motivating problem.

Motivating Problem 9.5.8. Find an algorithm for computing the DFT that runs in less time than $O(N^2)$.

In the next few chapters, we'll see such an algorithm: the *Fast Fourier Transform*.

Problems

9.5.1. (a) Consider $\mathbf{Z}/(N) = \{0, 1, \ldots, N - 1\}$ and suppose $t \in \mathbf{Z}/(N)$. What set do we get if we add t to each element of $\mathbf{Z}/(N)$? (Suggestion: Try some random values of N and t and see what happens; then generalize.)

(b) Now let $h : \mathbf{Z}/(N) \to \mathbf{C}$ be a complex-valued function on $\mathbf{Z}/(N)$, and fix some $t \in \mathbf{Z}/(N)$. Explain why

$$\sum_{k=0}^{N-1} h(k) = \sum_{n=0}^{N-1} h(n - t). \tag{9.5.7}$$

Suggestion: Try writing out both sides of (9.5.7).

9.5.2. compare time estimates for multiplications

9.6 Inner products and orthogonality

In this section, which is something of a digression from our main discussion, we'll take another approach to Fourier inversion (Definition 9.4.3 and Theorem 9.4.4), namely, approaching inversion in terms of dot products and linear algebra. Note that since this alternative approach is not needed in the rest of this book, and since all of the proofs are left to you, this section is well-suited for self-study or a supplementary project.

As in Section 9.4, fix $N \in \mathbf{N}$ and let $\omega = e^{2\pi i/N}$ be the natural primitive Nth root of unity in $\mathbf{C}$. Note that by thinking of vectors in $\mathbf{C}^N$ as signals, we can identify the space of all signals with the vector space $\mathbf{C}^N$ (see Definition 9.3.1). That observation leads to the following definition.

Definition 9.6.1. For signals $f, g : \mathbf{Z}/(N) \to \mathbf{C}$, written in vector form $f = \begin{bmatrix} f(0) \\ \vdots \\ f(N-1) \end{bmatrix}$ and $g = \begin{bmatrix} g(0) \\ \vdots \\ g(N-1) \end{bmatrix}$, we define the *inner product* of f and g to be

$$\langle f, g \rangle = \frac{1}{N} \sum_{n=0}^{N-1} f(n)\overline{g(n)}, \tag{9.6.1}$$

where $\overline{g(n)}$ denotes the complex conjugate of $g(n)$.

The inner product on signals has properties very similar to those of the real-valued dot product with which you may be more familiar.

Theorem 9.6.2. *For signals $f, f_i, g, g_i : \mathbf{Z}/(N) \to \mathbf{C}$ $(i = 1, 2)$ and $a, b \in \mathbf{C}$, we have that*

$$\begin{aligned} \langle af_1 + bf_2, g \rangle &= a \langle f_1, g \rangle + b \langle f_2, g \rangle, \\ \langle f, ag_1 + bg_2 \rangle &= \overline{a} \langle f, g_1 \rangle + \overline{b} \langle f, g_2 \rangle, \end{aligned} \tag{9.6.2}$$

where $\overline{a}$ and $\overline{b}$ are the complex conjugates of a and b, respectively.

In other words, the inner product $\langle f, g \rangle$ is linear in the first variable and almost linear in the second variable. (Not that you really need to know this, but for the record, properly speaking, we say that $\langle f, g \rangle$ is *skew-linear* in the second variable.)

Proof. Problem 9.6.1. □

The reason we introduce the inner product on $\mathbf{C}^N$ is to be able to define the following idea.

Definition 9.6.3. Let $\{f_1, \ldots, f_k\}$ be a set of signals $f_i : \mathbf{Z}/(N) \to \mathbf{C}$. To say that $\{f_1, \ldots, f_k\}$ is an *orthonormal set* means that:

1. Whenever $i \neq j$, $\langle f_i, f_j \rangle = 0$; and
2. For all i, $\langle f_i, f_i \rangle = 1$.

We introduce Definition 9.6.3 with the following example in mind. Recall (Definition 9.3.2) that for $k \in \mathbf{Z}/(N)$, the basic trigonometric signals e_k are defined in vector form to be

$$e_k = \begin{bmatrix} 1 \\ \omega^k \\ \vdots \\ \omega^{(N-1)k} \end{bmatrix}. \tag{9.6.3}$$

We then have that:

Theorem 9.6.4. *The set $\{e_0, e_1, \ldots, e_{N-1}\}$ is an orthonormal set.*

Proof. Problem 9.6.2. □

The following is one of the key properties of orthonormal sets.

Theorem 9.6.5. *Let $\{f_1, \ldots, f_k\}$ be an orthonormal set of signals. Then $\{f_1, \ldots, f_k\}$ is linearly independent.*

Proof. Problem 9.6.3. □

Corollary 9.6.6. *The set $\{e_0, e_1, \ldots, e_{N-1}\}$ is a basis for the space $\mathbf{C}^N$ of all possible signals.*

Proof. Problem 9.6.4. □

Corollary 9.6.6 means that every signal $f : \mathbf{Z}/(N) \to \mathbf{C}$ is a linear combination of the e_k, since the e_k span the space of all possible signals. So what are the coefficients of that linear combination? Yup, you guessed it — they're precisely the Fourier coefficients $\hat{f}(k)$. That is:

Theorem 9.6.7. *Let $f : \mathbf{Z}/(N) \to \mathbf{C}$ be a signal, and suppose that*

$$f = a_0 e_0 + \ldots a_{N-1} e_{N-1}, \tag{9.6.4}$$

where $a_0, \ldots, a_{N-1} \in \mathbf{C}$. Then for $k \in \mathbf{Z}/(N)$, we have

$$a_k = \langle f, e_k \rangle = \sum_{n=0}^{N-1} f(n)\overline{e_k(n)} = \hat{f}(k). \tag{9.6.5}$$

Proof. Problem 9.6.5. □

Comparing (9.6.5) and (9.6.4) with Definition 9.4.1 and the Inversion Theorem 9.4.4, we see that Theorem 9.6.7 is precisely the Inversion Theorem written in vector notation, as promised.

Remark 9.6.8. As in earlier sections (see ??), all of the material in this section can be modified to work with the finite field $\mathbf{F}_q$ and a primitive element ω of $\mathbf{F}_q$ in place of $\mathbf{C}$ and a primitive Nth root of unity, where $N = q - 1$. The principal change is that the dot product on signals $f, g : \mathbf{Z}/(N) \to \mathbf{F}_q$ is now defined to be

$$\langle f, g\rangle = \frac{1}{N}\sum_{n=0}^{N-1} f(n)g(n)^{-1}. \tag{9.6.6}$$

That inverse on $g(n)^{-1}$ causes a number of problems! But they aren't insurmountable; see Problem 9.6.6 for details.

Problems

9.6.1. (*Proves Theorem 9.6.2*) For signals $f, g : \mathbf{Z}/(N) \to \mathbf{C}$, let

$$\langle f, g\rangle = \frac{1}{N}\sum_{n=0}^{N-1} f(n)\overline{g(n)}. \tag{9.6.7}$$

Suppose that $f, f_i, g, g_i : \mathbf{Z}/(N) \to \mathbf{C}$ $(i = 1, 2)$ are signals and $a, b \in \mathbf{C}$.

(a) Prove that $\langle af_1 + bf_2, g\rangle = a\langle f_1, g\rangle + b\langle f_2, g\rangle$.

(b) Prove that $\langle f, ag_1 + bg_2\rangle = \overline{a}\langle f, g_1\rangle + \overline{b}\langle f, g_2\rangle$.

9.6.2. (*Proves Theorem 9.6.4*) Let $\langle f, g\rangle$ be defined by (9.6.7), as in Problem 9.6.1, and for $0 \le k \le N-1$, let $e_k = \begin{bmatrix} 1 \\ \omega^k \\ \vdots \\ \omega^{(N-1)k} \end{bmatrix}$.

(a) Prove that $\langle e_k, e_k\rangle = 1$.

(b) Prove that for $j \neq k$, we have $\langle e_j, e_k\rangle = 1$. (Suggestion: Orthogonality Lemma 9.3.4.)

9.6.3. (*Proves Theorem 9.6.5*) Let $\langle f, g\rangle$ be defined by (9.6.7), as in Problem 9.6.1, and suppose that $\{e_0, e_1, \dots, e_{N-1}\}$ is an orthonormal set (but not necessarily the one from Problem 9.6.2). Prove that $\{e_0, e_1, \dots, e_{N-1}\}$ is linearly independent. (Suggestion: Use the approach from Remark 5.3.11, combined with the inner product.)

9.6.4. (*Proves Corollary 9.6.6*) From Theorem 9.6.5, we know that $\{e_0, e_1, \dots, e_{N-1}\}$ is a linearly independent subset of $\mathbf{C}^N$. Prove that $\{e_0, e_1, \dots, e_{N-1}\}$ is actually a basis for $\mathbf{C}^N$. (Suggestion: Use Section 5.6.)

9.6.5. Suppose $\{e_0, e_1, \dots, e_{N-1}\}$ is an orthonormal set (but not necessarily the one from Problem 9.6.2). Let $f : \mathbf{Z}/(N) \to \mathbf{C}$ be a signal, and suppose that

$$f = a_0e_0 + \dots a_{N-1}e_{N-1}, \tag{9.6.8}$$

where $a_0, \dots, a_{N-1} \in \mathbf{C}$. Prove that for $k \in \mathbf{Z}/(N)$, we have $\langle f, e_k\rangle = a_k$.

9.6.6. Let $\mathbf{F}_q$ be the finite field of order q (Definition 7.6.19), let $N = q - 1$, and let ω be a primitive element of $\mathbf{F}_q$ (Definition 7.6.8). (By Theorem 7.6.9, such an ω must exist and have order N.) For suitable signals $f, g : \mathbf{Z}/(N) \to \mathbf{F}_q$, let

$$\langle f, g\rangle = \frac{1}{N} \sum_{n=0}^{N-1} f(n) g(n)^{-1}. \tag{9.6.9}$$

The goal of this problem is to adapt Problems 9.6.1–9.6.1 to this new setting.

(a) When we say "suitable signals" above, we mean that not every pair of signals $f, g : \mathbf{Z}/(N) \to \mathbf{F}_q$ will actually work when plugged into (9.6.9). Exactly when is $\langle f, g\rangle$ defined? Explain.

(b) Consider Theorem 9.6.2. Only one of the properties in (9.6.2) still holds for the finite field version of $\langle f, g\rangle$ defined in (9.6.9) — which is it? Prove the property that still works.

(c) For $k \in \mathbf{Z}/(N)$, let

$$e_k = \begin{bmatrix} 1 \\ \omega^k \\ \vdots \\ \omega^{(N-1)k} \end{bmatrix}. \tag{9.6.10}$$

Prove that $\{e_0, e_1, \ldots, e_{N-1}\}$ is an orthonormal set. (Suggestion: Finite Fields Orthogonality Lemma ??.)

(d) Do the finite fields version of Problems 9.6.3 and 9.6.4. What, if anything, changes?

(e) Do the finite fields version of Problem 9.6.5. What, if anything, changes?

Chapter 10

Groups

To me, a story can be both concrete and abstract, or a concrete story can hold abstractions.

— David Lynch

10.1 Groups and subgroups

Definition 10.1.1. A *group* is a set G along with a binary operation $\cdot$, usually written as multiplication, such that the following axioms are satisfied.

1. *(Associativity)* For any $a, b, c \in G$, $(ab)c = a(bc)$.

2. *(Identity)* There exists an element $1 \in G$ such that $1a = a = a1$ for all $a \in G$.

3. *(Inverses)* For every $a \in G$, there exists some $a^{-1} \in G$ such that $aa^{-1} = 1 = a^{-1}a$.

Definition 10.1.2. Let G be a group. To say that G is *abelian* means that for all $a, b \in G$, we have that $ab = ba$.

While nonabelian groups are important, in this book, we only really care about abelian groups, and you are welcome to think of all groups as being abelian. However, all of the theory we consider works for nonabelian groups with no extra effort, so we'll just state and prove results for the general, possibly nonabelian, case.

Remark 10.1.3. If you have previous experience with groups, you may remember that abelian groups are sometimes written additively, with $a + b$ replacing $a \cdot b$, identity written as 0, and inverses written as negatives. In this notation, the first four axioms of a ring (Definition 4.2.2) say precisely that if R is a ring, then R with the operation $+$ is an abelian group. However, to avoid needlessly confusing readers unfamiliar with groups, we'll stick to multiplicative examples only.

Notation 10.1.4. We can extend the usual notation of exponents to an arbitrary group. That is, for a positive integer n, we can abbreviate

$$\begin{aligned} a^n &= \overbrace{a \cdot \dots \cdot a}^{n \text{ times}}, \\ a^{-n} &= \overbrace{a^{-1} \cdot \dots \cdot a^{-1}}^{n \text{ times}}, \\ a^0 &= 1. \end{aligned} \tag{10.1.1}$$

You can check that the usual laws of exponents for a fixed base a apply in an arbitrary group; the details are mildly tedious but straightforward, so we'll omit them.

The following class of examples is the reason that we are suddenly interested in groups.

Example 10.1.5. Let R be a ring, and let $U(R)$ be the set of all units of R. Because the product of two units is a unit (see Problem 4.2.7), the multiplication operation $\cdot$ of R defines an operation on $U(R)$. We now check that $U(R)$, with the operation $\cdot$, is a group.

1. For $a, b, c \in U(R)$, $(ab)c = a(bc)$ because multiplication is associative in all of R, and therefore in $U(R)$.

2. The multiplicative identity 1 of R is a unit, so $U(R)$ has an identity element.

3. By definition, every unit has an inverse, which is itself a unit, so every element of $U(R)$ has an inverse in $U(R)$.

Note that if F is a field, then $U(F) = F^\times$, the multiplicative group of F (Definition 7.6.6). (And now, hopefully, the use of the word "group" in that name makes a little more sense.)

More precisely, we don't so much need to study an entire group $U(R)$ or $F^\times$ as we need to study parts of it, or more precisely, certain *subgroups* of $F^\times$, which we now sort of define twice. We first consider the morally correct definition.

Definition 10.1.6. Let G be a group. A *subgroup* of G is a sub*set* of G that is itself a group, using the same operation as G.

Notation 10.1.7. We use the notation $H \leq G$ to say that H is a subgroup of a group G. The point of using $\leq$ is to distinguish between H being a sub*group* of G and H being merely a sub*set* of G ($H \subseteq G$).

On the plus side, Definition 10.1.6 is succinct and generalizes to almost every other algebraic object, in that if a *foo* is some kind of algebraic object, then a *subfoo* is a subset of a foo that is itself a foo, using the same operation(s) as the larger foo. (See: subring, subfield, and so on.) On the minus side, Definition 10.1.6 is not that helpful if you want to figure out if a particular subset of a group G is actually a subgroup of G. For that purpose, we have the following theorem, which you can think of as an equivalent definition of subgroup.

Theorem 10.1.8 (Subgroup Theorem)**.** *Let G be a group, and let S be a subset of G. Then S is actually a subgroup of G if and only if all three of the following conditions hold.*

1. *(Identity) $1 \in S$ (i.e., S contains the multiplicative identity of G).*
2. *(Multiplicative closure) S is closed under the operation of G, i.e., if $a, b \in S$, then $ab \in S$.*
3. *(Inverse closure) S is closed under taking inverses, i.e., if $a \in S$, then $a^{-1} \in S$.*

Proof. On the one hand, if S is a subgroup of G, then the above three conditions must hold in S because of the axioms of a group.

On the other hand, suppose the three conditions hold. Because of multiplicative closure, the operation of G defines an operation on S (otherwise we would have no hope of turning S into a group using this operation). Associativity holds for any $a, b, c \in S$ because it holds for any $a, b, c \in G$, and the identity and inverse axioms of a group hold in S by the identity and inverse closure conditions. $\square$

For the DFT and FFT, we are primarily interested in the following group.

Definition 10.1.9. We define C_n to be the set of all nth roots of unity in $\mathbf{C}$. In other words:

$$C_n = \{z \in \mathbf{C} \mid z^n = 1\}. \tag{10.1.2}$$

As we saw in Theorem 9.2.3, if $\omega = e^{2\pi i/n}$, then

$$C_n = \left\{1, \omega, \omega^2, \dots, \omega^{n-1}\right\}. \tag{10.1.3}$$

Theorem 10.1.10. *For $n, k \in \mathbf{N}$, we have that:*

1. *C_n is a subgroup of $\mathbf{C}^\times$, the multiplicative group of the complex numbers; and*
2. *If k divides n, then C_k is a subgroup of C_n.*

Note that statement (2) of Theorem 10.1.10 is important to us because subgroup chains like

$$C_1 \leq C_2 \leq C_4 \leq C_8 \leq C_{16} \leq \cdots \tag{10.1.4}$$

are what we'll use to describe the Fast Fourier Transform.

Proof. Problem 10.1.1 shows that:

1. C_n contains the complex number 1;
2. C_n is closed under multiplication; and
3. C_n is closed under inverses.

The first statement of the Theorem 10.1.10 then follows from the Subgroup Theorem 10.1.8. As for the second statement, if $z^k = 1$ and $n = kd$, then $z^n = z^{kd} = (z^k)^d = 1$, which means that C_k is a subset of C_n. The theorem follows. $\square$

Another way of thinking about the group C_n as a subgroup of $\mathbf{C}^\times$ is in terms of the following fundamental idea, of which we already saw a special case in Definition 7.6.7.

Definition 10.1.11. Let G be a group and a an element of G. We define the *cyclic subgroup generated by* a to be

$$\langle a \rangle = \{a^n \mid n \in \mathbf{Z}\}, \tag{10.1.5}$$

the set of all powers of a, positive, negative, and zero (see Notation 10.1.4). If $\langle a \rangle$ happens to be equal to the entirety of G, we say that G is *cyclic*, and that G is *generated by* a.

Theorem 10.1.12. *Let G be a group, and let a be an element of G. Then $\langle a \rangle$, the cyclic subgroup generated by a, is a subgroup of G.*

Proof. Problem 10.1.2 shows that:

1. $\langle a \rangle$ contains the multiplicative identity 1;
2. $\langle a \rangle$ is closed under multiplication; and
3. $\langle a \rangle$ is closed under inverses.

The theorem then follows from the Subgroup Theorem 10.1.8. □

Example 10.1.13. For a positive integer n, let $\omega_n = e^{2\pi i/n}$. Then Theorem 9.2.3 says that $C_n = \langle \omega_n \rangle$, and therefore, that C_n is cyclic.

Problems

10.1.1. Let $C_n = \{z \in \mathbf{C} \mid z^n = 1\}$.

(a) Prove that $1 \in C_n$.
(b) Prove that if $z, w \in C_n$, then $zw \in C_n$.
(c) Prove that if $z \in C_n$, then $z^{-1} \in C_n$.

10.1.2. Let G be a group, and let $\langle a \rangle = \{a^n \mid n \in \mathbf{Z}\}$.

(a) Prove that $1 \in \langle a \rangle$.
(b) Prove that if $b, c \in \langle a \rangle$, then $bc \in \langle a \rangle$.
(c) Prove that if $b \in \langle a \rangle$, then $b^{-1} \in \langle a \rangle$.

10.2 Orders of elements

Remember that in Definition 7.6.10, we saw a special case of the following definition.

Definition 10.2.1. Let G be a group and let a be an element of G. If $a^n = 1$ for some positive integer n, we define the *order* of a to be the *smallest* possible n such that $a^n = 1$. Otherwise, if $a^n \neq 1$ for all positive integers n, we say that a has *infinite order*.

We begin with an observation about orders and exponents.

Theorem 10.2.2. *Let G be a group and let a be an element of G of finite order n. Then the expression a^k depends precisely on the congruence class of k mod n. In other words, $k = \ell$ in $\mathbf{Z}/(n)$ if and only if $a^k = a^\ell$.*

As a consequence, the expression a^k is unambiguous even if k is in $\mathbf{Z}/(n)$ and not an ordinary integer.

Proof. Suppose a has order n, and suppose that $k = \ell$ in $\mathbf{Z}/(n)$, or in other words, suppose that $k = \ell + qn$ for some integer q. Then

$$a^k = a^{\ell+qn} = a^\ell a^{qn} = a^\ell (a^n)^q = a^\ell 1^q = a^\ell. \tag{10.2.1}$$

Conversely, suppose $a^k = a^\ell$. If we let $m = k - \ell$, we see that it suffices to show that if $a^m = 1$, then $m = qn$ for some $q \in \mathbf{Z}$. However, in that case, by the Division Theorem 2.3.1, let $m = qn + r$ with $0 \leq r < n$. Then

$$a^r = a^{m-qn} = a^m (a^n)^q = 1 \cdot 1^q = 1. \tag{10.2.2}$$

However, by definition, n is the *smallest* positive integer n such that $a^n = 1$. Since $0 \leq r < n$, it follows that $r = 0$, and therefore, that $m = qn$. The theorem follows. □

Since $a^0 = 1$, Theorem 10.2.2 immediately implies:

Corollary 10.2.3. *Let G be a group and let a be an element of G of finite order n. Then $a^k = 1$ if and only if and only if $k = 0$ in $\mathbf{Z}/(n)$, or in other words, if and only if k is a multiple of n.* □

In this section and the next, we justify several properties of the order of an element (see Theorem 7.6.11) that we used back in Section 7.6. Again, recall that the *order* of an algebraic object (e.g., a group) is the number of elements in that algebraic object. We begin by proving part (1) of Theorem 7.6.11.

Corollary 10.2.4. *Let G be a group and let a be an element of G of finite order n. Then the cyclic subgroup $\langle a \rangle$ contains n elements (i.e., has order n).*

Proof. By Definition 10.1.11, the elements of $\langle a \rangle$ are precisely the powers of a. By Theorem 10.2.2, the powers of a are in bijective correspondence with the elements of $\mathbf{Z}/(n)$; in particular, there are n such powers. □

The Order of a Power Formula (part (2) of Theorem 7.6.11) takes a bit more work.

Theorem 10.2.5 (Order of a Power Formula)**.** *Let G be a group and let a be an element of G of finite order n. Then the order of a^k is $\dfrac{n}{\gcd(k,n)}$.*

Proof. Let $k = dm$, where $d = \gcd(k, n)$ and $\gcd(m, n) = 1$. The theorem then follows from two separate statements:

1. If d divides n, then the order of a^d is n/d.
2. If $\gcd(m, n) = 1$, then the order of a^m is still n.

Both of these statements are proven in Problem 10.2.1. □

The remaining part of Theorem 7.6.11 is deeper than the other two, so we'll wait until the next section to prove it.

Problems

10.2.1. (*Proves Theorem 10.2.5*) Let G be a group and let a be an element of G of finite order n.

(a) Suppose $d > 0$ divides n. Explain why the order of a^d is n/d. (Suggestion: Use the definition of order.)

(b) Suppose $\gcd(m, n) = 1$. Prove that the order of a^m is still n. (Suggestion: Use the facts that $(a^m)^k = 1$ if and only if $km = 0$ in $\mathbf{Z}/(n)$, and that m is a unit in $\mathbf{Z}/(n)$ (why?).)

10.3 Cosets

We have already seen the additive version of the idea of coset back in Section 7.2, where we used cosets to define the quotient ring R/I. While that was important for ring theory, cosets turn out to be even more fundamental in group theory. As usual, we begin with the definition.

Definition 10.3.1. Let G be a group, and let H be a subgroup of G. For $a \in G$, we define the *left multiplicative coset* aH to be

$$aH = \{ah \mid h \in H\}. \tag{10.3.1}$$

If the context is clear, instead of saying "left multiplicative coset", we just say *coset*. (In particular, since we mainly care about abelian groups, we won't worry too much about left cosets vs. right cosets.)

As always, the first thing you should do when you see a new abstract definition is to make up an example and play around with it.

Example 10.3.2. Let $G = F_{13}^{\times}$, the multiplicative group of $\mathbf{F}_{13}$, and let $H = \langle 5 \rangle$, the cyclic subgroup of G generated by $5 \in G$. Since $5^2 = 12$, $5^3 = 8$, and $5^4 = 1 \pmod{13}$, we see that

$$H = \{1, 5, 12, 8\}. \tag{10.3.2}$$

Choosing (randomly) $6 \in G$, we see that

$$6H = \{6, 30, 72, 48\} = \{6, 4, 7, 9\} \pmod{13}. \tag{10.3.3}$$

Choosing (again randomly) $7 \in G$, we see that

$$7H = \{7, 35, 84, 56\} = \{7, 9, 6, 4\} \pmod{13}, \tag{10.3.4}$$

which is the same set as $6H$, just written in a different order; in other words, $7H = 6H$.

It turns out that to get a different coset, we need to choose a representative not already contained in H or $6H$, so let's try $3 \in G$:

$$3H = \{3, 15, 36, 24\} = \{3, 2, 10, 11\} \pmod{13}. \tag{10.3.5}$$

Note that $G = H \cup 6H \cup 3H$, and that all three cosets are disjoint.

In most beginning courses in abstract algebra, cosets are used to prove various fundamental results about groups, such as Corollaries 10.3.11 and 10.3.12. While we'll do that as well, our main applications of cosets stem from (a) understanding the fact that the cosets of a subgroup H of a group G *partition* G (Theorem 10.3.7) and (b) understanding the *definition* of cosets.

Theorem 10.3.3. *Let H be a subgroup of a group G, and let a be an element of G. If b is an element of aH, then $aH = bH$.*

Proof. Suppose $b \in aH$, or in other words, $b = ah_0$ for some $h_0 \in H$. Then for any $bh \in bH$, since H is closed under multiplication, $bh = a(h_0h) \in AH$; and for any $ah \in aH$, since H is closed under multiplication and inverses, $ah = b(h_0^{-1}h) \in bH$. The theorem follows. □

Theorem 10.3.3 motivates the following definition.

Definition 10.3.4. Let H be a subgroup of a group G, and let a be an element of G. A *representative* of the coset aH is an element b of aH. Note that by Theorem 10.3.3, if b is a representative of aH, then bH is an alternate name for aH.

Theorem 10.3.3 also has the following important corollary.

Corollary 10.3.5. *Let H be a subgroup of a group G, and let a and b be an element of G. Then aH and bH are either disjoint or equal.*

In other words, cosets are either disjoint are equal.

Proof. If aH and bH are not disjoint, then there exists some $g \in aH \cap bH$, and by Theorem 10.3.3, $aH = gH = bH$. □

Putting all of this nonsense together, we have the following important picture: Given a group G and a subgroup H, we can *partition* G into cosets of H, as shown in Figure 10.3.1. The point is that every element of G appears in exactly one coset, which means that, for example, we can "divide and conquer" a sum over G by summing over each coset and summing the totals from each coset, without worrying about missing an element of G or counting an element of G twice.

To be precise:

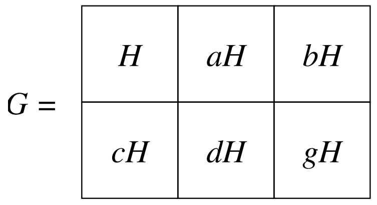

Figure 10.3.1: A group G partitioned into six cosets of a subgroup H

Definition 10.3.6. Let X be a set, and let $\{A_1, \dots, A_n\}$ be a collection of subsets of X. To say that $\{A_1, \dots, A_n\}$ *partition* X means that:

1. (Nonempty) Each $A_i \neq \emptyset$;
2. (Cover) $X = \displaystyle\bigcup_{i=1}^{n} A_i$ (i.e., X is the union of the A_i); and
3. (Pairwise disjoint) If $i \neq j$, then $A_i \cap A_j = \emptyset$.

Theorem 10.3.7. *Let G be a finite group and let H be a subgroup of G. Consider all left cosets of H, and choose one element a_i from each coset of H so that $\{a_1H, \dots, a_nH\}$ contains each coset of H exactly once. Then $\{a_1H, \dots, a_nH\}$ partitions G.*

Proof. Comparing Definition 10.3.6, we first observe that each coset a_iH is nonempty because $a_i \in a_iH$. Next, any $g \in G$ is contained in gH, which must be equal to some a_iH because of the way we chose the a_i. Finally, Corollary 10.3.5 implies that the a_iH are pairwise disjoint. □

Definition 10.3.8. Let G be a finite group, and let H be a subgroup of G. A choice of coset representatives like the set $\{a_1, \dots, a_n\}$ in the statement of Theorem 10.3.7 is called a *transversal* for H in G. In other words, to say that $\{a_1, \dots, a_n\}$ is a transversal for H in G means that

$$G = a_1H \cup \dots \cup a_nH \tag{10.3.6}$$

and that for $i \neq j$, $a_iH \cap a_jH = \emptyset$ (i.e., a_iH and a_jH are disjoint).

Example 10.3.9. Returning to the example $G = U_{13}$, $H = \langle 5 \rangle$ of Example 10.3.2, we see that $\{1, 6, 3\}$ is a transversal for H in G, as are $\{1, 7, 3\}$, $\{12, 9, 10\}$, and so on.

As promised, we finish this section by using Theorem 10.3.7 to prove a few fundamental facts about finite groups, including Lagrange's Theorem (part (3) of Theorem 7.6.11). Again, theory is not our main focus, but it takes relatively little effort, so why not, right? We begin with a small but important observation, which you'll realize is true as soon as you try any examples.

Lemma 10.3.10. *Let G be a finite group, let H be a subgroup of G, and let a be an element of G. The coset aH has the same number of elements as H does, and therefore, all cosets contain the same number of elements.*

Proof. Problem 10.3.6. □

As a consequence, we have the following corollaries (Corollaries 10.3.11 and 10.3.12), either of which could be called Lagrange's Theorem.[*]

Corollary 10.3.11. *Let G be a finite group, and let H be a subgroup of G. Then the order of H (number of elements in H) divides the order of G.*

Proof. Let k be the order of H and let n be the order of G. By Theorem 10.3.7, G can be partitioned into a finite number (say, q) pieces, each of size k. It follows that $n = qk$. □

Corollary 10.3.12. *Let G be a finite group of order N, and let a be an element of G. Then the order of a divides N.*

Proof. This follows from Corollary 10.3.11 and the fact that the order of a is equal to the order of the subgroup $\langle a \rangle$ (Theorem 10.2.4). □

Problems

10.3.1. Let $\omega = e^{2\pi i/12}$, and consider the subgroups C_2 and C_6 of C_{12}. (In fact, $C_2 \leq C_6 \leq C_{12}$.)

(a) What are the elements of C_2 in terms of $\omega = e^{2\pi i/12}$? Explain. And what are the elements of C_6 in terms of ω?

(b) Write C_{12} as a disjoint union of cosets of C_2.

(c) Find a transversal for C_2 in C_{12}. See if you can find one chosen by some orderly method.

(d) Find a transversal for C_2 in C_6.

(e) Find a transversal for C_6 in C_{12}.

10.3.2. Let $\omega = e^{2\pi i/18}$, and consider the subgroups C_3 and C_6 of C_{18}. (In fact, $C_3 \leq C_6 \leq C_{18}$.)

(a) What are the elements of C_3 in terms of $\omega = e^{2\pi i/18}$? Explain. And what are the elements of C_6 in terms of ω?

(b) Write C_{18} as a disjoint union of cosets of C_3.

(c) Find a transversal for C_3 in C_{18}. See if you can find one chosen by some orderly method.

(d) Find a transversal for C_3 in C_6.

(e) Find a transversal for C_6 in C_{18}.

10.3.3. Consider the subgroup $H = \langle 2 \rangle$ in $\mathbf{F}_{31}^{\times}$.

[*]As usual with theorem-naming, Lagrange didn't prove either of these facts as such, just an early ancestor of them; see Roth [**?**]. But hey, we still call them, and other related things, Lagrange's Theorem.

(a) List all of the elements of H (the powers of 2 (mod 31)).

(b) Write $\mathbf{F}_{31}^{\times}$ as a disjoint union of cosets of H.

(c) Find a transversal for H in $\mathbf{F}_{31}^{\times}$.

10.3.4. Let $\mathbf{F}_9 = \mathbf{F}_3[i]$, where i is a root of $m(x) = x^2 + 1$, and consider the subgroup $H = \langle i \rangle$ in $\mathbf{F}_9^{\times}$.

(a) List all of the elements of H.

(b) Write $\mathbf{F}_9^{\times}$ as a disjoint union of cosets of H.

(c) Find a transversal for H in $\mathbf{F}_9^{\times}$.

10.3.5. Let $\mathbf{F}_{16} = \mathbf{F}_2[\alpha]$, where α is a root of $m(x) = x^4 + x + 1$, and consider the subgroup $H = \langle \alpha^2 + \alpha \rangle$ in $\mathbf{F}_{16}^{\times}$.

(a) List all of the elements of H.

(b) Write $\mathbf{F}_{16}^{\times}$ as a disjoint union of cosets of H.

(c) Find a transversal for H in $\mathbf{F}_{16}^{\times}$.

10.3.6. Let G be a group, let H be a subgroup of G, and let a be an element of G. Define a function $f : H \to aH$ by

$$f(h) = ah \tag{10.3.7}$$

for all $h \in H$. To prove that H and aH have the same number of elements, it suffices to show that f is a bijection (one-to-one and onto).

(a) Explain why f is onto (surjective).

(b) Prove that f is one-to-one. (Suggestion: Suppose $f(h_1) = f(h_2)$. Why can we then conclude that $h_1 = h_2$?)

10.4 Public-key cryptography and the discrete log problem

Before we get into the algebra of *public-key cryptography*, we first need to discuss the basic setup involved.

Suppose Alice and Bob have a communicatios channel over which they want to communicate secretly. Now, if they shared some secret piece of information in common, like a password or other encryption key, we take it as given that they would be able to communicate without Eve hearing through the use of *private-key cryptography*. However, if Alice and Bob haven't previously communicated a shared private key, they face a Catch-22: To share a private key without Eve finding it out, they need some kind of encrypted communications channel, for which they need to share a private key....

So how do we get around this problem? Well, in the 1970's[†], Diffie and Hellman [**?**] devised a way for Alice and Bob to share a piece of information in full public view of Eve

[†]Diffie and Hellman's work marked the *public* origin of public-key cryptography, but very similar ideas were first devised at the British intelligence agency GCHQ a few years earlier; see blah [**?**] for an account.

but, paradoxically, still keep that piece of information hidden from Eve. Their key idea was to make use of what is known as a *one-way trapdoor function*, which is a function that is invertible in theory, but whose inverse takes an impossibly long time to compute unless you happen to know a secret key.

To be specific, their scheme, which is still in widespread use today, can be explained in group-theoretic terms as follows.

Algorithm 10.4.1 (Cyclic group Diffie-Hellman)**.** Given a cyclic group G with generator a of order n:

1. Alice chooses a secret integer x such that $\gcd(x, n) = 1$ and also chooses some z such that $xz = 1 \pmod{n}$.
2. Alice "broadcasts" (makes the world aware of, by some means) the value $b = a^x$. The group element $b \in G$ becomes a sort of public "mailbox" that the world can use to communicate securely with Alice.
3. To establish a secure channel with Alice, Bob chooses some integer y with the goal of secretly sharing the piece of information a^y with Alice.
4. Bob then calcuates $c = b^y$ and transmits c to Alice.
5. Alice then calculates c^z, which should then be the shared piece of information a^y.

Now, it might not be your first thought to use an element of a group as a shared secret key of some sort. However, if you think about it, any kind of practical implementation of computation in a group G involves representing the elements of G as sequences of bits in some way, and if n is large, then those bit sequences are just as good as any other bit sequence that you might use as a shared secret key, right? In any case, putting secrecy aside for the moment, the first order of business is to check that the algorithm really does let Alice and Bob share the piece of information a^y.

Theorem 10.4.2. *In the notation of Algorithm 10.4.1, we have $c^z = a^y$. (In other words, Diffie-Hellman works as advertised.)*

Proof. For the proof and a discussion of other details of Diffie-Hellman, see Problem 10.4.1. □

Now, to be fair, to the uninitiated, Algorithm 10.4.1 might not seem very practical: Where would you find a cyclic group just lying around the house or whatever? But you, a veteran of Chapter 7 (especially Section 7.6), know the secret: The multiplicative group of a finite field is cyclic, so we can take G to be $\mathbf{F}_p^\times$, or more generally, $\mathbf{F}_q^\times$. See Problem 10.4.2 and 10.4.3 for the details.

Perhaps more importantly, just having some strange complicated way to share a piece of information a^y isn't very useful if the eavesdropper Eve can also figure out what a^y is. However, looking at Algorithm 10.4.1 closely, we see that since Eve only overhears Alice's "mailbox" $b = a^x$ and Bob's transmission $c = a^{xy}$, Eve can't figure out a^y without more work.

Of course, if Eve knew what x was, she could figure out z just like Alice did and recover the secret information a^y. Therefore, the security of Diffie-Hellman relies on the following problem being difficult to solve efficiently.

Problem 10.4.3 (Discrete logarithm problem)**.** Let G be a cyclic group with generator a (i.e., $G = \langle a \rangle$), and suppose a has order n. The *discrete logarithm problem* for G to the base a is: Find an algorithm that, given $b = a^x$, determines the value of $x \pmod n$.

Specifically, if $G = \mathbf{F}_q^\times$ for some $q = p^e$ (p prime), and $G = F^\times$, the multiplicative group of F, then there is no known efficient algorithm for solving Problem 10.4.3 (references). (Well, outside of the use of a quantum computer, of which there is no publicly known example of a practical size, as of this writing in 2024; see (references).) For example, if $q = p$ for (say) an industrial-size prime in the hundreds of digits, the kind of log-antilog table we saw in Section 7.7 would have more entries than there are particles in the universe.

Diffie-Hellman can also be applied to an abelian group written in *additive notation*, which works as follows:

- Instead of ab, we write $a + b$.
- Instead of a^n, we write na as an abbreviation for a added to itself n times.
- Instead of a multiplicative identity 1, we have an additive identity 0.
- Instead of a^{-1}, we write $-a$.

So the associative, identity, inverse, and commutative properties look like:

$$\begin{gathered}(a+b)+c = a+(b+c)\\ a+0 = a = 0+a\\ a+(-a) = 0 = (-a)+a\\ a+b = b+a.\end{gathered} \tag{10.4.1}$$

As you can see, an additive abelian group works just like addition in a ring; in fact, you can combine the first four axioms of a ring (Definition 4.2.2) into one axiom saying that a ring is an additive abelian group under the operation $+$. In any case, if you rewrite Diffie-Hellman in additive notation, it may look a little different, but it works exactly the same; see Problem 10.4.4 for details.

So if additive Diffie-Hellman is just the same algorithm, why bother to write it out in a new notation? Well, it turns out that the solution set to an equation like

$$y^2 = x^3 + 2, \tag{10.4.2}$$

known as an *elliptic curve*, which when considered over a finite field (say, $\mathbf{F}_7$) gives an additive abelian group that is quite naturally useful for Diffie-Hellman and other kinds of cryptography schemes. Currently (2024), experts believe that Diffie-Hellman is at least as secure as other similar crytography schemes (e.g., Diffie-Hellman in $\mathbf{F}_p^\times$), and possibly more so, but for now, this hasn't been proven either way. See Koblitz [**?**] for much more on elliptic curve Diffie-Hellman and other elliptic curve encryption algorithms.

Problems

10.4.1. This problem works out some of the key details in the Diffie-Hellman algorithm. In the notation of Algorithm 10.4.1:

(a) How can Alice ensure that $\gcd(x, n) = 1$ and find z such that $xz = 1 \pmod{n}$?

(b) Why is $\alpha^{xyz} = \alpha^y$? Does Bob also need to choose y such that $\gcd(y, n) = 1$?

10.4.2. Consider the case of Algorithm 10.4.1 where $G = \mathbf{F}_p^\times$ for some prime p. Rewrite the description of Algorithm 10.4.1 completely in terms of calculations $\pmod{p}$, without reference to any group-theoretic terminology.

10.4.3. Consider the case of Algorithm 10.4.1 where $G = \mathbf{F}_q^\times$ for p prime and $q = p^e$. Rewrite the description of Algorithm 10.4.1 completely in terms of calculations in $\mathbf{F}_q$, without reference to any group-theoretic terminology.

10.4.4. Rewrite Algorithm 10.4.1 for an abelian group using additive notation. Think carefully about what terms mean in an additive context, e.g., what does it mean for a to be a generator for G?

10.4.5. Think about what makes Algorithm 10.4.1 work. Does G really have to be cyclic? Why or why not? (I.e., what happens if you remove that condition on G?) Does G really have to be abelian? Why or why not?

Chapter 11

Even faster: The Fast Fourier Transform

If you want to make any progress in life, go as far as any reasonable person would go and then keep going.

— John Horton Conway

Higher. Further. Faster. More.

— Carol Danvers, *Captain Marvel*, Kelly Sue Deconnick

11.1 Can we make multiplication faster?

The Fast Fourier Transform (FFT) has been cited as one of the ten most important algorithms of the 20th century, and its applications are almost too numerous to mention (though ?? does a pretty good job of mentioning them). However, because this is an algebra book, we'll use a motivating problem that's easy to describe: multiplying large numbers.

Motivating Problem 11.1.1. Using the standard grade-school algorithm for long multiplication, the time required to multiply two N-digit numbers is $O(N^2)$ (see Problem 11.1.1). Is there a faster algorithm for multiplying two N-digit numbers, e.g., an algorithm that's $O(N \log N)$?

If you give Problem 11.1.1 proper consideration, it should blow your mind. How could there possibly be a new and dramatically faster way to perform a calculation that people have probably been doing for thousands of years? Nonetheless, in 1971, Schönhage and Strassen (reference?) devised a practical method for multiplying two N-digit numbers that doesn't quite get the time required down to $O(N \log N)$, but is pretty close. As we'll see, the key idea is that the Discrete Fourier Transform (DFT) reduces standard multiplication to an $O(N)$ operation and so the bottleneck becomes the DFT itself, a problem solved by the FFT. We start that discussion in our next section.

New 2019 development: Integer multiplication $N \log N$ by Harvey and van der Hoven [**?**], though not yet practical.

Problems

11.1.1. Recall the usual grade-school algorithm for multiplication of two N-digit numbers.

(a) Suppose we multiply 1,234 × 5,432. Carefully count the number of single-digit multiplications and the number of additions that are required to carry out the usual grade-school algorithm on these two numbers. (Ignore carrying.)

(b) Generalize the previous part to see how long it takes to multiply two N-digit numbers. How many single-digit multiplications, as a function of N? Find an upper bound for the number of additions as well (easier than the exact number). (You can again ignore carrying, which is an $O(N)$ factor, but in perhaps a non-obvious way.)

11.2 The Fast Fourier Transform

As mentioned in the previous section, we now have the following motivating problem.

Motivating Problem 11.2.1. If we can speed up the computation of the DFT, we can speed up multiplication of polynomials of degree N, and therefore, multiplication of N-digit numbers.

The basic method of speeding up the DFT is called the *Fast Fourier Transform*, or *FFT*. The idea is due to Cooley and Tukey, though our description of it is based on Diaconis and Rockmore. Before we can get to our description, however, we need to establish some notation and a few facts about subgroup chains in C_N (the group of Nth roots of unity). As you may remember (see Section 9.2), for fixed N, C_N is a cyclic group of order N, generated by $\omega = e^{2\pi i/N}$. We have the following useful observation about subgroups of C_N.

Theorem 11.2.2. *Fix a positive integer N and let $\omega = e^{2\pi i/N}$. If $N = dq$ for positive $d, q \in \mathbf{Z}$, then C_d (the group of complex dth roots of unity) is precisely $\langle \omega^q \rangle$, the subgroup of C_N generated by ω^q.*

Proof. For $N = dq$ as stated, we have

$$\omega^q = (e^{2\pi i/N})^q = e^{2\pi i(q/N)} = e^{2\pi i/d}. \tag{11.2.1}$$

However, $e^{2\pi i/d}$ is precisely the natural complex dth root of unity that generates C_d. □

Example 11.2.3. Consider the chain of subgroups $C_1 \le C_2 \le C_4 \le C_8$, and fix $N = 8$ and $\omega = e^{2\pi i/8}$. Then by Theorem 11.2.2, we have the subgroups

$$\begin{aligned} C_1 &= \left\langle \omega^8 \right\rangle = \langle 1 \rangle = \{1\} \\ C_2 &= \left\langle \omega^4 \right\rangle = \left\{1, \omega^4\right\} \\ C_4 &= \left\langle \omega^2 \right\rangle = \left\{1, \omega^2, \omega^4, \omega^6\right\} \end{aligned} \tag{11.2.2}$$

of

$$C_8 = \langle \omega \rangle = \left\{1, \omega, \omega^2, \omega^3, \omega^4, \omega^5, \omega^6, \omega^7\right\} \tag{11.2.3}$$

Keep Example 11.2.3 and its analogues for other values of N in mind as you try to absorb how the following algorithm works.

Algorithm 11.2.4 (FFT)**.** Fix $N \in \mathbf{N}$ and $\omega = e^{2\pi i/N}$. Let

$$C_1 = H_0 \leq H_1 \leq \cdots \leq H_{n-1} \leq H_n = C_N \tag{11.2.4}$$

be a chain of subgroups of C_N that starts at the trivial subgroup $C_1 = \{1\}$ and ends at the group C_N itself. We define the *Fast Fourier Transform based on the subgroup chain* $H_0 \leq \cdots \leq H_n$ as follows.

Throughout this algorithm, let $\mathbf{x} = \begin{bmatrix} x(0) \\ \vdots \\ x(N-1) \end{bmatrix}$ represent the current state of our calculation and let $\mathbf{y} = \begin{bmatrix} y(0) \\ \vdots \\ y(N-1) \end{bmatrix}$ represent the new state of our calculation after the current step. The goal is to start with $\mathbf{x} = \begin{bmatrix} f(0) \\ \vdots \\ f(N-1) \end{bmatrix}$ and end with $\mathbf{x} = \begin{bmatrix} \hat{f}(0) \\ \vdots \\ \hat{f}(N-1) \end{bmatrix}$.

1. *Initialize.* Set $\mathbf{x} = \begin{bmatrix} f(0) \\ \vdots \\ f(N-1) \end{bmatrix}$.

2. *Main loop.* For $i = 1$ to n:

   (a) *Notation.* Suppose that $H_{i-1} = \langle \omega^m \rangle$ and $H_i = \left\langle \omega^k \right\rangle$, where m and k are chosen from Theorem 11.2.2; remember that the "big" exponent m corresponds to the old, smaller subgroup H_{i-1}, and the "small" exponent k corresponds to the new, bigger subgroup H_i. (Special case: For $i = 1$ and $H_{i-1} = H_0 = \{1\}$, we take $m = N$.) Since $H_{i-1} \leq H_i$, we have that k divides m, or $m = kd$ for some positive integer d. Use the standard transversal $1, \omega^k, \omega^{2k}, \ldots, \omega^{(d-1)k}$ for H_{i-1} in H_i. (Note that d is the number of cosets of H_{i-1} in H_i.)

   (b) *Fill entries of* $\mathbf{y}$ *corresponding to the new subgroup* H_i. For $j = 0$ to $(N/k) - 1$ (i.e., jk ranges over all exponents of ω appearing in H_i, or ω^{jk} ranges over all elements of H_i), set

$$\begin{aligned} y(jk) &= \sum_{r=0}^{d-1} x(jm + rk)\omega^{-rkj} \\ &= x(jm) + x(jm+k)\omega^{-kj} + x(jm+2k)\omega^{-2kj} + \ldots \\ &\qquad + x(jm + (d-1)k)\omega^{-(d-1)kj}. \end{aligned} \tag{11.2.5}$$

As will be clearer in examples, in the sum on the right-hand side of (11.2.5), we can think of jm as a "starting point" coming from the old subgroup H_{i-1} and rk as "offsets" stepping through the exponents in the coset representatives $1, \omega^k, \omega^{2k}, \dots, \omega^{(d-1)k}$. Note that in the expression $x(jm + rk)$ in (11.2.5), we calculate $jm + rk$ as an integer (mod N), so for example, as j ranges from 0 to N/k, jm cycles through each of the integers $0, m, 2m, \dots, N - m$ exactly d times; in other words, the "starting points" jm increase exactly d times as fast as the "coset representatives" rk.

(c) *Translate the subgroup fill to entries corresponding to the cosets of H_i in C_N.* That is, do exactly the same filling as in step 2b, but with the indices of y and x increased by $1, \dots, k-1$. To be precise, for $\ell = 1$ to $k-1$ and $j = 0$ to $(N/k)-1$, we set

$$y(jk + \ell) = \sum_{r=0}^{d-1} x(jm + rk + \ell)\omega^{-rkj}. \tag{11.2.6}$$

Note that in principle (and certainly in any code that you would actually want to run) we could actually combine this step and the previous step into one big loop, starting with for $\ell = 0$ to $k-1$, but we hope that separating the two makes our algorithm slightly more human-readable.

(d) *Set current state to new state and loop.* Set $\mathbf{x} = \mathbf{y}$ and go to the next step of the main loop.

3. *Rescale.* Divide every entry of $\mathbf{x}$ by N.

Note that if we can show that Algorithm 11.2.4 actually always computes the DFT, then essentially the same proof will show that if we change each ω^{-rkj} to ω^{rkj} and omit step 3, the same algorithm computes the inverse DFT with the same big-O complexity.

Example 11.2.5. For any $N \in \mathbf{N}$, consider the FFT based on the subgroup chain $C_1 \leq C_N$; in other words, $H_0 = C_1$ and $H_1 = C_N$. Note that since we only have one step in the subgroup chain, $n = 1$. Therefore, stepping through the algorithm, we get:

1. *Initialize.* Set $\mathbf{x} = \begin{bmatrix} f(0) \\ \vdots \\ f(N-1) \end{bmatrix}$.

2. *Main loop.* This has one step in it, with $i = n = 1$.

(a) *Notation.* Since $H_0 = \langle \omega^N \rangle$ and $H_1 = \langle \omega^1 \rangle$, we have $m = N$, $k = 1$, $d = N$, and our coset representatives for H_0 in H_1 are $1, \omega, \omega^2, \dots, \omega^{N-1}$, or in other words, the elements of $G = C_N$.

(b) *Fill entries of* $\mathbf{y}$ *corresponding to H_1.* Since $d = N$, $k = 1$, $N/k = N$, and $m = 0$ (mod N), for $j = 0$ to $N - 1$, set

$$y(j) = \sum_{r=0}^{N-1} x(r)\omega^{-rj}. \tag{11.2.7}$$

(c) *Translate to cosets.* Since the only coset of $H_1 = C_N$ in C_N is itself, nothing happens in this step.

(d) Set $\mathbf{x} = \mathbf{y}$.

3. *Rescale.* Divide every entry of $\mathbf{x}$ by N.

Comparing (11.2.7) and (9.4.1), we see that after the rescaling step, we indeed have $\mathbf{x} = \begin{bmatrix} \hat{f}(0) \\ \vdots \\ \hat{f}(N-1) \end{bmatrix}$. However, we have saved no time: Since we are doing exactly the same computation as the regular DFT, this "FFT" is just the DFT in more complicated notation, and is therefore still an $O(N^2)$ algorithm.

While Example 11.2.5 does reassure us that we haven't gone completely off the deep end here, we also see that if we don't try to cut N up into small pieces, then no time savings will occur.

We next consider a more typical, and more useful, example of an FFT.

Example 11.2.6. For $N = 6$, consider the FFT based on the subgroup chain $C_1 \leq C_2 \leq C_6$ (so $n = 2$); that is, $H_0 = C_1$, $H_1 = C_2$, and $H_2 = C_6$. Stepping through the algorithm:

1. *Initialize.* Set $\mathbf{x} = \begin{bmatrix} f(0) \\ \vdots \\ f(N-1) \end{bmatrix}$.

2. *Main loop.* This has two steps in it, with $i = 1$ and $i = 2$.

- $i = 1$.

(a) *Notation.* Since $H_0 = \langle \omega^6 \rangle$ and $H_1 = \langle \omega^3 \rangle$, we have $m = 6$, $k = 3$, $N/k = 2$, $d = 2$, with coset representatives $1, \omega^3$.

(b) *Fill entries of* $\mathbf{y}$ *corresponding to* H_1. Going from $j = 0$ to 1, we have

$$\begin{aligned} y(0) &= x(0) + x(3)\omega^{-0} = f(0) + f(3), \\ y(3) &= x(0) + x(3)\omega^{-3} = f(0) + f(3)\omega^{-3}. \end{aligned} \tag{11.2.8}$$

(c) *Translate to cosets.* Similarly, we have

$$\begin{aligned} y(1) &= x(1) + x(4)\omega^{-0} = f(1) + f(4), \\ y(4) &= x(1) + x(4)\omega^{-3} = f(1) + f(4)\omega^{-3}, \\ y(2) &= x(2) + x(5)\omega^{-0} = f(2) + f(5), \\ y(5) &= x(2) + x(5)\omega^{-3} = f(2) + f(5)\omega^{-3}. \end{aligned} \tag{11.2.9}$$

(d) Set

$$\mathbf{x} = \mathbf{y} = \begin{bmatrix} f(0)+f(3) \\ f(1)+f(4) \\ f(2)+f(5) \\ f(0)+f(3)\omega^{-3} \\ f(1)+f(4)\omega^{-3} \\ f(2)+f(5)\omega^{-3} \end{bmatrix}. \tag{11.2.10}$$

- $i = 2$.

(a) *Notation.* Since $H_1 = \langle \omega^3 \rangle$ and $H_2 = \langle \omega^1 \rangle$, we have $m = 3$, $k = 1$, $N/k = 6$, $d = 3$, with coset representatives $1, \omega, \omega^2$.

(b) *Fill entries of* $\mathbf{y}$ *corresponding to* H_2. Going from $j = 0$ to 5, we have

$$\begin{aligned} y(0) &= x(0) + x(1)\omega^{-0} + x(2)\omega^{-0} \\ &= f(0) + f(3) + f(1) + f(4) + f(2) + f(5), \\ y(1) &= x(3) + x(4)\omega^{-1} + x(5)\omega^{-2} \\ &= f(0) + f(3)\omega^{-3} + f(1)\omega^{-1} + f(4)\omega^{-4} + f(2)\omega^{-2} + f(5)\omega^{-5}, \\ y(2) &= x(0) + x(1)\omega^{-2} + x(2)\omega^{-4} \\ &= f(0) + f(3) + f(1)\omega^{-2} + f(4)\omega^{-2} + f(2)\omega^{-4} + f(5)\omega^{-4}, \\ y(3) &= x(3) + x(4)\omega^{-3} + x(5)\omega^{-6} \\ &= f(0) + f(3)\omega^{-3} + f(1)\omega^{-3} + f(4) + f(2) + f(5)\omega^{-3}, \\ y(4) &= x(0) + x(1)\omega^{-4} + x(2)\omega^{-8} \\ &= f(0) + f(3) + f(1)\omega^{-4} + f(4)\omega^{-4} + f(2)\omega^{-2} + f(5)\omega^{-2}, \\ y(5) &= x(3) + x(4)\omega^{-5} + x(5)\omega^{-10} \\ &= f(0) + f(3)\omega^{-3} + f(1)\omega^{-5} + f(4)\omega^{-2} + f(2)\omega^{-4} + f(5)\omega^{-1}. \end{aligned} \tag{11.2.11}$$

(c) *Translate to cosets.* $H_2 = C_6$, so nothing happens.

(d) Set $\mathbf{x} = \mathbf{y}$.

3. *Rescale.* Divide every entry of $\mathbf{x}$ by $N = 6$.

And lo and behold, our final answer is

$$\mathbf{x} = \frac{1}{6} \begin{bmatrix} f(0)+f(1)+f(2)+f(3)+f(4)+f(5) \\ f(0)+f(1)\omega^{-1}+f(2)\omega^{-2}+f(3)\omega^{-3}+f(4)\omega^{-4}+f(5)\omega^{-5} \\ f(0)+f(1)\omega^{-2}+f(2)\omega^{-4}+f(3)+f(4)\omega^{-2}+f(5)\omega^{-4} \\ f(0)+f(1)\omega^{-3}+f(2)+f(3)\omega^{-3}+f(4)+f(5)\omega^{-3} \\ f(0)+f(1)\omega^{-4}+f(2)\omega^{-2}+f(3)+f(4)\omega^{-4}+f(5)\omega^{-2} \\ f(0)+f(1)\omega^{-5}+f(2)\omega^{-4}+f(3)\omega^{-3}+f(4)\omega^{-2}+f(5)\omega^{-1} \end{bmatrix} = \begin{bmatrix} \hat{f}(0) \\ \vdots \\ \hat{f}(N-1) \end{bmatrix}. \tag{11.2.12}$$

In other words, it works!

Now, as the first epigraph to the chapter says, to make any progress in life, one should go as far as any reasonable person would go and then keep going. With that in mind, we have the following example.

Example 11.2.7. For $N = 16$, consider the FFT based on the subgroup chain $C_1 \leq C_2 \leq C_4 \leq C_8 \leq C_{16}$ (so $n = 4$); in other words, $H_0 = C_1$, $H_1 = C_2$, $H_2 = C_4$, $H_3 = C_8$, and $H_4 = C_{16}$. Let's step through the algorithm as in Example 11.2.7, though we will omit many details, for the sake of both typography and the limits of your patience.

1. *Initialize.* Set $\mathbf{x} = \begin{bmatrix} f(0) \\ \vdots \\ f(N-1) \end{bmatrix}$.

2. *Main loop.* This goes from $i = 1$ to 4.

   - $i = 1$.

     (a) *Notation.* Since $H_0 = \langle \omega^{16} \rangle$ and $H_1 = \langle \omega^8 \rangle$, we have $m = 16$, $k = 8$, $N/k = 2$, $d = 2$, with coset representatives $1, \omega^8$.

     (b) *Fill entries of* $\mathbf{y}$ *corresponding to* H_1. We have

$$\begin{aligned} y(0) &= x(0) + x(8)\omega^{-0} = f(0) + f(8), \\ y(8) &= x(0) + x(8)\omega^{-8} = f(0) + f(8)\omega^{-8}. \end{aligned} \tag{11.2.13}$$

     (c) *Translate and loop.* We then set, for example

$$\begin{aligned} y(1) &= x(1) + x(9)\omega^{-0} = f(1) + f(9), \\ y(2) &= x(2) + x(10)\omega^{-0} = f(2) + f(10), \\ y(9) &= x(1) + x(9)\omega^{-8} = f(1) + f(9)\omega^{-8}, \\ y(10) &= x(2) + x(10)\omega^{-8} = f(2) + f(10)\omega^{-8}, \end{aligned} \tag{11.2.14}$$

     and so on. Then set $\mathbf{x} = \mathbf{y}$, as usual.

   - $i = 2$.

     (a) *Notation.* $H_1 = \langle \omega^8 \rangle$, $H_2 = \langle \omega^4 \rangle$, $m = 8$, $k = 4$, $N/k = 4$, $d = 2$, representatives $1, \omega^4$.

     (b) *Fill entries of* $\mathbf{y}$ *corresponding to* H_2.

$$\begin{aligned} y(0) &= x(0) + x(4)\omega^{-0} = f(0) + f(8) + f(4) + f(12), \\ y(4) &= x(8) + x(12)\omega^{-4} = f(0) + f(8)\omega^{-8} + f(4)\omega^{-4} + f(12)\omega^{-12}, \\ y(8) &= x(0) + x(4)\omega^{-8} = f(0) + f(8) + f(4)\omega^{-8} + f(12)\omega^{-8}, \\ y(12) &= x(8) + x(12)\omega^{-12} = f(0) + f(8)\omega^{-8} + f(4)\omega^{-12} + f(12)\omega^{-4}. \end{aligned} \tag{11.2.15}$$

     (c) *Translate and loop.* Translate $y(0)$ to $y(1)$, $y(2)$, and $y(3)$ (details omitted) and similarly for all other empty entries; and set $\mathbf{x} = \mathbf{y}$.

- $i = 3$.
  (a) *Notation.* $H_2 = \langle \omega^4 \rangle$, $H_3 = \langle \omega^2 \rangle$, $m = 4$, $k = 2$, $N/k = 8$, $d = 2$, representatives $1, \omega^2$.
  (b) *Fill entries of* $\mathbf{y}$ *corresponding to* H_3. See Figure 11.2.1.
  (c) *Translate and loop.* The one coset translation is precisely Figure 11.2.1 with every index in $x(n)$ or $y(n)$ (but not $f(n)$) increased by 1, so we omit the details. We then set $\mathbf{x} = \mathbf{y}$ and loop.
- $i = 4$.
  (a) *Notation.* $H_3 = \langle \omega^2 \rangle$, $H_4 = \langle \omega^1 \rangle$, $m = 2$, $k = 1$, $N/k = 16$, $d = 2$, representatives $1, \omega$.
  (b) *Fill entries of* $\mathbf{y}$ *corresponding to* H_4. This now means that we fill every row, and in fact, each row is filled with the corresponding correct value of the DFT, up to rescaling. We omit the details here, due to space and typography, but you should check this yourself by trying a few entries, starting from the $i = 3$ state (i.e., Figure 11.2.1 and its translate to the odd-numbered entries).

3. *Rescale.* Divide every entry of $\mathbf{x}$ by $N = 16$.

Well, with that example, I hope it at least seems plausible that the FFT really does compute the DFT. However, once you've worked through a number of examples, there are a lot of questions that might occur to you.

Motivating Problem 11.2.8. Some questions to ponder about the FFT, as presented:

- What's the idea behind the arbitrary-looking formulas in Algorithm 11.2.4?
- Why is Algorithm 11.2.4 faster than the ordinary "matrix multiplication" version of the DFT (Remark 9.4.2)?
- What can you do faster with the FFT?
- What was the point of all of that stuff about cosets and transversals?
- How can we prove that the FFT *always* produces the right answer?

While answers to most of the above questions will have to wait until we develop some more ideas, we do have the following partial answer to the first question of Motivating Problem 11.2.8.

Remark 11.2.9. In the notation of Algorithm 11.2.4, consider step i of the main loop, with $H_{i-1} = \langle \omega^m \rangle$, $H_i = \langle \omega^k \rangle$, $d = m/k$, and

$$T_{i-1,i} = \left\{1, \omega^k, \omega^{2k}, \dots, \omega^{(d-1)k}\right\} \tag{11.2.16}$$

as the standard transversal for H_{i-1} in H_i.

One (partial) interpretation of the formula (11.2.5) is:

$$\begin{aligned}
y(0) &= x(0) + x(2)\omega^{-0} \\
&= f(0) + f(4) + f(8) + f(12) \\
&\qquad + f(2) + f(6) + f(10) + f(14) \\
y(2) &= x(4) + x(6)\omega^{-2} \\
&= f(0) + f(4)\omega^{-4} + f(8)\omega^{-8} + f(12)\omega^{-12} \\
&\qquad + f(2)\omega^{-2} + f(6)\omega^{-6} + f(10)\omega^{-10} + f(14)\omega^{-14}, \\
y(4) &= x(8) + x(10)\omega^{-4} \\
&= f(0) + f(4)\omega^{-8} + f(8) + f(12)\omega^{-8} \\
&\qquad + f(2)\omega^{-4} + f(6)\omega^{-12} + f(10)\omega^{-4} + f(14)\omega^{-12}, \\
y(6) &= x(12) + x(14)\omega^{-6} \\
&= f(0) + f(4)\omega^{-12} + f(8)\omega^{-8} + f(12)\omega^{-4} \\
&\qquad + f(2)\omega^{-6} + f(6)\omega^{-2} + f(10)\omega^{-14} + f(14)\omega^{-10}, \\
y(8) &= x(0) + x(2)\omega^{-8} \\
&= f(0) + f(4) + f(8) + f(12) \\
&\qquad + f(2)\omega^{-8} + f(6)\omega^{-8} + f(10)\omega^{-8} + f(14)\omega^{-8}, \\
y(10) &= x(4) + x(6)\omega^{-10} \\
&= f(0) + f(4)\omega^{-4} + f(8)\omega^{-8} + f(12)\omega^{-12} \\
&\qquad + f(2)\omega^{-10} + f(6)\omega^{-14} + f(10)\omega^{-2} + f(14)\omega^{-6}, \\
y(12) &= x(8) + x(10)\omega^{-12} \\
&= f(0) + f(4)\omega^{-8} + f(8) + f(12)\omega^{-8} \\
&\qquad + f(2)\omega^{-12} + f(6)\omega^{-4} + f(10)\omega^{-12} + f(14)\omega^{-4}, \\
y(14) &= x(12) + x(14)\omega^{-14} \\
&= f(0) + f(4)\omega^{-12} + f(8)\omega^{-8} + f(12)\omega^{-4} \\
&\qquad + f(2)\omega^{-14} + f(6)\omega^{-10} + f(10)\omega^{-6} + f(14)\omega^{-2}.
\end{aligned}$$

Figure 11.2.1: FFT, $N = 16$, $i = 3$, subgroup fill

To get the jth output for the "new subgroup" H_i, you take a linear combination of the jth input from the "old subgroup" H_{i-1}, with inputs"offset" by the elements of the transversal and coefficients taken to be the $(-j)$th powers of the elements of the transversal.

(Note that, again, for the inverse transform, we use jth powers instead of $(-j)$th powers.)

More precisely, suppose we want to compute $y(jk)$, the output corresponding to the jth element ω^{jk} of H_i (starting with $j = 0$).

- Start with $x(jm)$, the output corresponding to the jth element ω^{jm} of H_{i-1}, where j is counted mod the order of H_{i-1} (i.e., with wraparound).
- Take the inputs corresponding to ω^{jm} times each element of the transversal $T_{i-1,i}$ (the "offsets"), i.e.,
$$x(jm),\ x(jm+k),\ \ldots,\ x(jm+(d-1)k). \tag{11.2.17}$$
- Set $y(jk)$ to be a linear combination of those inputs, taking the coefficient for the term corresponding to the transversal element ω^{rk} to be $(\omega^{rk})^{-j} = \omega^{-rkj}$:
$$x(jm) + x(jm+k)\omega^{-kj} + \cdots + x(jm+(d-1)k)\omega^{-(d-1)kj}. \tag{11.2.18}$$

To see Remark 11.2.9 in action, we recall that step 2 of Example 11.2.6 is computed by

$$\begin{aligned} y(0) &= x(0) + x(1)\omega^{-0} + x(2)\omega^{-0} \\ y(1) &= x(3) + x(4)\omega^{-1} + x(5)\omega^{-2} \\ y(2) &= x(0) + x(1)\omega^{-2} + x(2)\omega^{-4} \\ y(3) &= x(3) + x(4)\omega^{-3} + x(5)\omega^{-6} \\ y(4) &= x(0) + x(1)\omega^{-4} + x(2)\omega^{-8} \\ y(5) &= x(3) + x(4)\omega^{-5} + x(5)\omega^{-10}. \end{aligned} \tag{11.2.19}$$

As promised, we get the outputs $y(1), \ldots, y(5)$ by repeating the inputs $x(0), x(3)$ corresponding to $H_{i-1} = \{\omega^0, \omega^3\}$ and then making adjustments to each step corresponding to the transversal $\{\omega^0, \omega^1, \omega^2\}$. Similarly, in Step 3 of Example 11.2.7, we have

$$\begin{aligned} y(0) &= x(0) + x(2)\omega^{-0}, & y(8) &= x(0) + x(2)\omega^{-8}, \\ y(2) &= x(4) + x(6)\omega^{-2}, & y(10) &= x(4) + x(6)\omega^{-10}, \\ y(4) &= x(8) + x(10)\omega^{-4}, & y(12) &= x(8) + x(10)\omega^{-12}, \\ y(6) &= x(12) + x(14)\omega^{-6}, & y(14) &= x(12) + x(14)\omega^{-14}. \end{aligned} \tag{11.2.20}$$

Again, we get the outputs $y(2), y(4), \ldots, y(14)$ by repeating the inputs $x(0), x(4), x(8), x(12)$ corresponding to $H_{i-1} = \{\omega^0, \omega^4, \omega^8, \omega^{12}\}$ and then adjusting each step as determined by the transversal $\{\omega^0, \omega^2\}$. Anyway, while the above may still not be completely transparent or natural, I hope it's least a little more memorable than (11.2.5)!

Problems

11.2.1. For $N = 8$, consider the FFT based on the subgroup chain $C_1 \leq C_2 \leq C_4 \leq C_8$ (so $n = 3$). Let $\omega = e^{2\pi i/8}$.

(a) In terms of ω, find generators for each H_i and transversals for each inclusion H_{i-1} in H_i.

(b) Compute the FFT from start to finish, and verify that the end result is the same as the result of the DFT for $N = 8$.

11.2.2. For $N = 12$, consider the FFT based on the subgroup chain $C_1 \leq C_3 \leq C_6 \leq C_{12}$ (so $n = 3$). Let $\omega = e^{2\pi i/12}$.

(a) In terms of ω, find generators for each H_i and transversals for each inclusion H_{i-1} in H_i.

(b) Compute the FFT from start to finish, and verify that the end result is the same as the result of the DFT for $N = 12$.

11.2.3. For $N = 27$, consider the FFT based on the subgroup chain $C_1 \leq C_3 \leq C_9 \leq C_{27}$ (so $n = 3$). Let $\omega = e^{2\pi i/27}$.

(a) In terms of ω, find generators for each H_i and transversals for each inclusion H_{i-1} in H_i.

(b) Compute Steps 1 and 2 of the FFT completely. You may indicate the results of the "translate" part of each step by "and so on" after demonstrating the pattern.

(c) Compute entries $y(1), y(2), y(3)$ of Step 3.

11.2.4. For $N = 24$, consider the FFT based on the subgroup chain $C_1 \leq C_2 \leq C_4 \leq C_{12} \leq C_{24}$ (so $n = 4$). Let $\omega = e^{2\pi i/24}$.

(a) In terms of ω, find generators for each H_i and transversals for each inclusion H_{i-1} in H_i.

(b) Compute Steps 1 and 2 of the FFT completely. You may indicate the results of the "translate" part of each step by "and so on" after demonstrating the pattern.

(c) Compute entries $y(2), y(4), y(6), y(8)$ of Step 3.

11.2.5. For $N = 36$, consider the FFT based on the subgroup chain $C_1 \leq C_3 \leq C_6 \leq C_{18} \leq C_{36}$ (so $n = 4$). Let $\omega = e^{2\pi i/36}$.

(a) In terms of ω, find generators for each H_i and transversals for each inclusion H_{i-1} in H_i.

(b) Compute Steps 1 and 2 of the FFT completely. You may indicate the results of the "translate" part of each step by "and so on" after demonstrating the pattern.

(c) Compute entries $y(2), y(4), y(6), y(8)$ of Step 3.

11.3 Circuit diagrams and the time complexity of the FFT

In this section, we answer the question:

- Why is Algorithm 11.2.4 faster than the ordinary "matrix multiplication" version of the DFT (Remark 9.4.2)?

We also begin to answer:

- What was the point of all of that stuff about cosets and transversals?

In addition, while Remark 11.2.9 gives some motivation for the formulas in Algorithm 11.2.4, this section provides a way to visualize those formulas that you may find to be helpful.

The idea we'll use to answer the above questions is the idea of a *circuit diagram* of an FFT. The goal of a circuit diagram for an FFT is, at each stage, to show which "inputs" (values of $x(t)$) are used to compute each "output" (value of $y(t)$). However, since circuit diagrams are, in some sense, more expository than mathematical, we'll start with examples before attempting a general definition.

Example 11.3.1. Let's return to Example 11.2.6, with $N = 6$, $\omega = \omega_6 = e^{2\pi i/6}$, and the FFT based on the subgroup chain $C_1 \leq C_2 \leq C_6$. In subgroup terms, we have

$$H_0 = C_1 = \{1\}, \qquad H_1 = C_2 = \left\langle \omega^3 \right\rangle, \qquad H_2 = C_6 = \langle \omega \rangle. \tag{11.3.1}$$

Note that $T_{0,1} = \left\{1, \omega^3\right\}$ is a transversal for H_0 in H_1 and $T_{1,2} = \left\{1, \omega, \omega^2\right\}$ is a transversal for H_1 in H_2.

The first step ($i = 1$) of Example 11.2.6 yields the following end result:

$$\begin{aligned}
y(0) &= x(0) + x(3)\omega^{-0}, \\
y(1) &= x(1) + x(4)\omega^{-0}, \\
y(2) &= x(2) + x(5)\omega^{-0}, \\
y(3) &= x(0) + x(3)\omega^{-3}, \\
y(4) &= x(1) + x(4)\omega^{-3}, \\
y(5) &= x(2) + x(5)\omega^{-3}.
\end{aligned} \tag{11.3.2}$$

Thinking in terms of inputs and outputs and not worrying about the ω^{-k} factors, we see that the outputs $y(0)$ and $y(3)$ are obtained by mixing the inputs $x(0)$ and $x(3)$, which we represent by the black graph on the left-hand side of Figure 11.3.1. Adding the red and blue translates of that diagram, we get the left-hand side of Figure 11.3.1. Note that we have also highlighted the boxes corresponding to elements of H_0, H_1, and H_2, to make it easier to identify what corresponds to those subgroups.

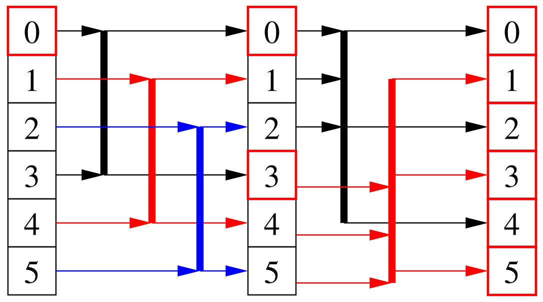

Figure 11.3.1: Circuit diagram for $C_1 \leq C_2 \leq C_6$

Turning to the $i = 2$ step, we have:

$$\begin{aligned}
y(0) &= x(0) + x(1)\omega^{-0} + x(2)\omega^{-0}, \\
y(1) &= x(3) + x(4)\omega^{-1} + x(5)\omega^{-2}, \\
y(2) &= x(0) + x(1)\omega^{-2} + x(2)\omega^{-4}, \\
y(3) &= x(3) + x(4)\omega^{-3} + x(5)\omega^{-6}, \\
y(4) &= x(0) + x(1)\omega^{-4} + x(2)\omega^{-8}, \\
y(5) &= x(3) + x(4)\omega^{-5} + x(5)\omega^{-10}.
\end{aligned} \tag{11.3.3}$$

As described in Remark 11.2.9, since $H_1 = \{\omega^0, \omega^3\}$ and $T_{1,2} = \{\omega^0, \omega^1, \omega^2\}$, the inputs going into any given output are either $x(0), x(1), x(2)$, coming from $\omega^0 \in H_1$ as adjusted by $T_{1,2}$, or $x(3), x(4), x(5)$, coming from $\omega^3 \in H_1$ as adjusted by $T_{1,2}$. The former case is represented by the black graph on the right-hand side of Figure 11.3.1, and the latter case by the red graph on the right-hand side of Figure 11.3.1.

Example 11.3.2. Next, let's return to Example 11.2.7, with $N = 16$, $\omega = \omega_6 = e^{2\pi i/16}$, and the FFT based on the subgroup chain $C_1 \leq C_2 \leq C_4 \leq C_8 \leq C_{16}$. In subgroup terms, we have

$$\begin{aligned}
&H_0 = C_1 = \{1\}, \quad H_1 = C_2 = \langle \omega^8 \rangle, \quad H_2 = C_4 = \langle \omega^4 \rangle, \\
&H_3 = C_8 = \langle \omega^2 \rangle, \quad H_4 = C_{16} = \langle \omega \rangle,
\end{aligned} \tag{11.3.4}$$

and we have the transversals

$$\begin{aligned}
&T_{0,1} = \{1, \omega^8\}, \quad T_{1,2} = \{1, \omega^4\}, \\
&T_{2,3} = \{1, \omega^2\}, \quad T_{3,4} = \{1, \omega\},
\end{aligned} \tag{11.3.5}$$

where $T_{i-1,i}$ is a transversal for H_{i-1} in H_i.

To save some space, and because the circuit diagram of step i is just the union of translates of the diagrams corresponding to elements of H_i, in each step of this FFT we will only draw the relevant *subgroup subdiagram*, that is, the portion of the circuit diagram for step i whose outputs correspond to H_i. For $i = 1$, we have

$$y(0) = x(0) + x(8)\omega^{-0}, \quad y(8) = x(0) + x(8)\omega^{-8} \tag{11.3.6}$$

and translates; we therefore have the subgroup subdiagram shown on the left-hand side of Figure 11.3.2. Similarly, for $i = 2$ we have

$$\begin{aligned} y(0) &= x(0) + x(4)\omega^{-0}, & y(8) &= x(0) + x(4)\omega^{-8}, \\ y(4) &= x(8) + x(12)\omega^{-4}, & y(12) &= x(8) + x(12)\omega^{-12} \end{aligned} \tag{11.3.7}$$

giving the subgroup subdiagram shown on the right-hand side of Figure 11.3.2. Note that, as described in Remark 11.2.9, (11.3.7) and the subgroup subdiagram both show how the output corresponding to the jth element of H_2 is connected back to the inputs starting with the input correspding to the jth element of H_1. That is, the $y(0)$ sum starts with $x(0)$, the $y(4)$ sum starts with $x(8)$, and so on.

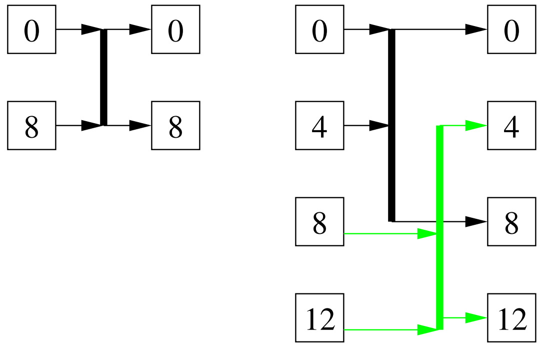

Figure 11.3.2: Partial circuit diagrams, $C_1 \leq C_2 \leq C_4 \leq C_8 \leq C_{16}$, $i = 1, 2$

For $i = 3$, we have

$$\begin{aligned} y(0) &= x(0) + x(2)\omega^{-0}, & y(8) &= x(0) + x(2)\omega^{-8}, \\ y(2) &= x(4) + x(6)\omega^{-2}, & y(10) &= x(4) + x(6)\omega^{-10}, \\ y(4) &= x(8) + x(10)\omega^{-4}, & y(12) &= x(8) + x(10)\omega^{-12}, \\ y(6) &= x(12) + x(14)\omega^{-6}, & y(14) &= x(12) + x(14)\omega^{-14} \end{aligned} \tag{11.3.8}$$

and its one translate, giving the subgroup subdiagram shown on the left-hand side of Figure 11.3.3. Finally, for $i = 4$, we have

$$\begin{aligned} y(0) &= x(0) + x(1)\omega^{-0}, & y(8) &= x(0) + x(1)\omega^{-8}, \\ y(1) &= x(2) + x(3)\omega^{-1}, & y(9) &= x(2) + x(3)\omega^{-9}, \\ y(2) &= x(4) + x(5)\omega^{-2}, & y(10) &= x(4) + x(5)\omega^{-10}, \\ y(3) &= x(6) + x(7)\omega^{-3}, & y(11) &= x(6) + x(7)\omega^{-11}, \\ y(4) &= x(8) + x(9)\omega^{-4}, & y(12) &= x(8) + x(9)\omega^{-12}, \\ y(5) &= x(10) + x(11)\omega^{-5}, & y(13) &= x(10) + x(11)\omega^{-13}, \\ y(6) &= x(12) + x(13)\omega^{-6}, & y(14) &= x(12) + x(13)\omega^{-14}, \\ y(7) &= x(14) + x(15)\omega^{-7}, & y(15) &= x(14) + x(15)\omega^{-15} \end{aligned} \tag{11.3.9}$$

with no translates (since $H_4 = G$), giving the subgroup subdiagram shown on the right-hand side of Figure 11.3.3. We again see the input-out pattern described in Remark 11.2.9, in that the output corresponding to the jth element of H_i is a linear combination of inputs corresonding to the jth element of H_{i-1} times the transversal $T_{3,4} = \{1, \omega\}$.

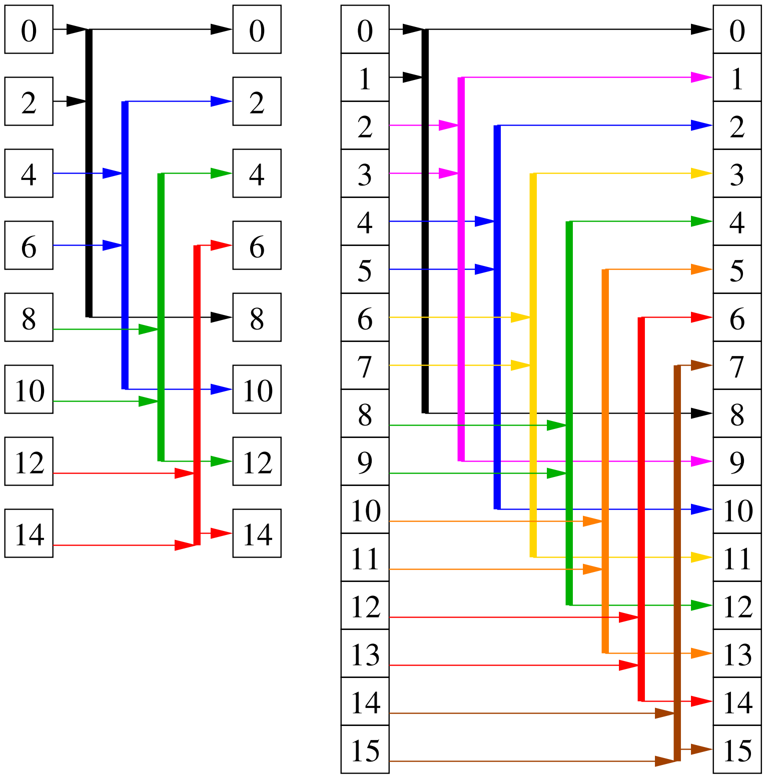

Figure 11.3.3: Partial circuit diagrams, $C_1 \leq C_2 \leq C_4 \leq C_8 \leq C_{16}$, $i = 3, 4$

For completeness, we present the full circuit diagram of this FFT in Figure 11.3.4. Note that as we did in Figure 11.3.1, we highlight the boxes that correspond to the elements of H_0, H_1, H_2, H_3, and H_4. Anyway, there's a lot going on in that picture, but the important part is that it gives some idea of the overall pattern of how this FFT works.

Armed with experience from actual examples, we can now give a more precise (though still somewhat heuristic) definiton of the thing we've been drawing.

Definition 11.3.3. The *subgroup subdiagram* for step i of a particular FFT is the diagram that connects all outputs corresponding to the subgroup H_i back to the inputs from which they are constructed, in the style of Figures 11.3.1–11.3.3. The *circuit diagram for step i of an FFT* is the union of all translates of the subgroup subdiagram for step i, and the *circuit diagram for an FFT* is the union of the circuit diagrams for all steps of that FFT. (For examples of full circuit diagrams, see Figures 11.3.1 and 11.3.4.)

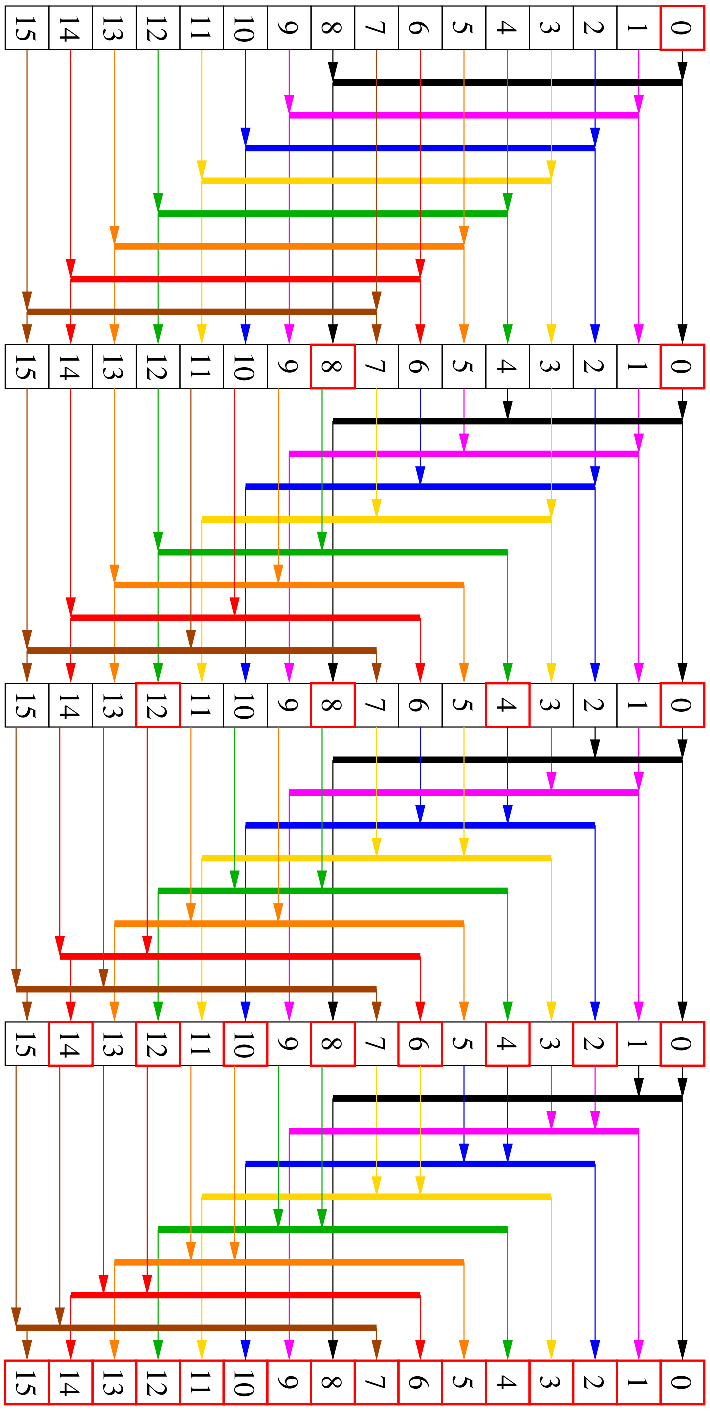

Figure 11.3.4: Circuit diagram for $C_1 \leq C_2 \leq C_4 \leq C_8 \leq C_{16}$

By Definition 11.3.3, to describe how to draw the circuit diagram for step i of an FFT, it's enough to describe how to draw the subgroup subdiagram for that step. Considering Remark 11.2.9 and looking carefully at Examples 11.3.1 and 11.3.2, we get the following recipe.

Algorithm 11.3.4. Following the conventions of the main loop of Algorithm 11.2.4, let $H_{i-1} = \langle \omega^m \rangle$ and $H_i = \langle \omega^k \rangle$, where $|H_{i-1}| = N/m$ and $|H_i| = N/k$. Let $d = m/k$, and use the standard transversal $T_{i-1,i} = \left\{1, \omega^k, \omega^{2k}, \dots, \omega^{(d-1)k}\right\}$ for H_{i-1} in H_i. The subgroup subdiagram for step i is given by two rules:

1. The output $y(jk)$, which corresponds to the jth element of H_i, is connected back to the inputs $x(jm), x(jm+k), x(jm+(d-1)k)$, which correspond to the jth element of H_{i-1} times the elements of the transversal $T_{i-1,i}$.

2. The outputs $y(j_1 k)$ and $y(j_2 k)$ are connected back to the same inputs exactly when $j_1 m = j_2 m \pmod{N}$.

Remark 11.3.5. If you look at the full circuit diagrams in Figures 11.3.1 and 11.3.4, one interesting feature is that there is exactly one path connecting each initial input to each final output. (Try a few input-output pairs and see for yourself!) While not a proof, that fact is at least consistent with each final output being a linear combination of the initial inputs with each input appearing exactly once, as we also see in the DFT. Of course, that doesn't show that those linear combinations have the cofficients we need to reconstruct the DFT, but it's at least a promising sign.

Perhaps the most notable feature of circuit diagrams is that they provide a way to visualize the time complexity of the FFT. To begin with, we need to keep in mind that even though the FFT correctly computes the DFT in terms of the original signal $f(t)$, at any given step of an actual practical computation, it only performs operations in terms of inputs $x(t)$ and outputs $y(t)$. So, for example, even though we went to some trouble in Example 11.2.6 to track what each step produces as an end result in terms of the original $f(t)$, as a plausibility test (though not yet a proof), the only computations actually performed by in the algorithm are the $x(t)$ to $y(t)$ computations we saw in Example 11.3.2.

We next establish our basic units of time. In particular, at first you might worry about the amount of time taken to compute the powers ω^ℓ that show up everywhere in the algorithm. However, since $\omega^N = 1$, we can initially create a lookup table of all values ω^ℓ for $0 \leq \ell < N$ in $O(N)$ time, after which calculating ω^{rk} (for example) boils down to calculating $rk \pmod{N}$. We may therefore take our basic arithmetic operations to be:

- Calculating one ω^{rk}, by calculating $\ell = rk \pmod{N}$ and looking up the value of ω^ℓ.
- Performing one complex addition.
- Performing one complex multiplication.

In fact, since each of these three operations takes a globally bounded amount of time, we can think of each of them as taking time 1.

Going back to the time complexity for the FFT, in the notation of Algorithm 11.3.4, since each output in step i connects back to d inputs in the circuit diagram for step i, the circuit diagram shows that each output is a linear combination of d inputs, a fact you can also verify directly either from (11.2.5) in the main loop or from Remark 11.2.9. Therefore, it takes $4d-1$ arithmetic operations to compute each output:

- d exponentiations ω^{rk};
- d complex multiplications; and
- $d-1$ additions.

It follows that in total, we need to perform $3d-1$ operations for each of the N outputs $y(0),\dots,y(N-1)$, so each step of the main loop requires $O(dN)$ operations. Finally, in the notation of Algorithm 11.2.4, there are n steps in the main loop, so the total time required for the FFT is $O(dNn)$.

To get the exponential speedup we want out of the FFT, we need to set an upper bound D, independent of N, to the factors $d = \{H_i\} / \{H_{i-1}\}$ that occur at each step of the main loop. In other words, we need to restrict the N we use in the FFT to a particular class of positive integers, such as powers of 2 ($D=2$), integers of the form $2^a3^b5^c$ ($D=5$), and so on. If we make that assumption, then we can absorb the d in each step as a constant factor, and the time estimate for the FFT becomes $O(Nn)$.

Finally, since the size of the subgroup $|H_i|$ grows by a factor of at least 2 each time, starting with $|H_0| = 1$ and increasing to $|H_N| = N$, we see that $N \geq 2^n$. It follows that $n \leq \log_2(N)$, which means that our time estimate becomes $O(N\log_2 N)$.

In other words:

Theorem 11.3.6 (Time complexity of the FFT)**.** *Suppose we set a global upper bound D on the ratio $d = |H_i| / |H_{i-1}|$ in a subgroup chain ending in C_N, where D is independent of N. Then the FFT on $\mathbf{Z}/(N)$ requires no more than $O(N\log_2 N)$ arithmetic operations.* □

As promised, we have achieved an exponential speedup from $O(N^2)$.

Example 11.3.7. For one example of counting operations directly from the circuit diagram of an FFT, consider the circuit diagram for the FFT on $\mathbf{Z}/(6)$ based on the subgroup chain $C_1 \leq C_2 \leq C_6$, as shown in Figure 11.3.1. In step 1 of this FFT, each output is obtained by combining 2 inputs, or in other words, using an equation of the form

$$y(*) = x(*)\omega^* + x(*)\omega^*, \tag{11.3.10}$$

where the $*$ represents some unspecified element of $\mathbf{Z}/(6)$. Therefore, each output requires computing 2 exponentiations, 2 multiplications, and 1 addition, for a total of 12 exponentiations, 12 multiplications, and 6 additions. Similarly, in step 2, each output comes from combining 3 inputs in the form

$$y(*) = x(*)\omega^* + x(*)\omega^* + x(*)\omega^*, \tag{11.3.11}$$

so each output requires computing 3 exponentiations, 3 multiplications, and 2 additions, for a total of 18 exponentiations, 18 multiplications, and 12 additions.

Example 11.3.8. To give a larger example, consider the circuit diagram for the FFT on $\mathbf{Z}/(16)$ based on the subgroup chain $C_1 \leq C_2 \leq C_4 \leq C_8 \leq C_{16}$, as shown in Figure 11.3.4. We see that in every step, each output is obtained by combining 2 inputs in the form

$$y(*) = x(*)\omega^* + x(*)\omega^*, \tag{11.3.12}$$

so each output requires computing 2 exponentiations, 2 multiplications, and 1 addition, for a total of 32 exponentiations, 32 multiplications, and 16 additions in each of the 4 steps.

More generally, if we use the analogous subgroup chain for an arbitrary $N = 2^n$, by exactly the same reasoning, each of the $n = \log_2(N)$ steps requires $2N$ exponentiations, $2N$ multiplications, and N additions. We therefore have a total of $2N \log_2(N)$ exponentiations, $2N \log_2(N)$ multiplications, and $N \log_2(N)$ additions, as promised in Theorem 11.3.6.

Problems

11.3.1. For $N = 8$, consider the FFT based on the subgroup chain $C_1 \leq C_2 \leq C_4 \leq C_8$ (so $n = 3$).

(a) Draw the subgroup subdiagram for step 1 of this FFT. How many exponentiations ω^{rk} does step 1 require? How many complex multiplications? How many complex additions?

(b) Same, but for step 2.

(c) Same, but for step 3.

11.3.2. For $N = 12$, consider the FFT based on the subgroup chain $C_1 \leq C_3 \leq C_6 \leq C_{12}$ (so $n = 3$).

(a) Draw the subgroup subdiagram for step 1 of this FFT. How many exponentiations ω^{rk} does step 1 require? How many complex multiplications? How many complex additions?

(b) Same, but for step 2.

(c) Same, but for step 3.

11.3.3. For $N = 27$, consider the FFT based on the subgroup chain $C_1 \leq C_3 \leq C_9 \leq C_{27}$ (so $n = 3$).

(a) Draw the subgroup subdiagram for step 1 of this FFT. How many exponentiations ω^{rk} does step 1 require? How many complex multiplications? How many complex additions?

(b) Same, but for step 2.

(c) Same, but for step 3.

11.3.4. For $N = 30$, consider the FFT based on the subgroup chain $C_1 \leq C_3 \leq C_6 \leq C_{30}$ (so $n = 3$).

(a) Draw the subgroup subdiagram for step 1 of this FFT. How many exponentiations ω^{rk} does step 1 require? How many complex multiplications? How many complex additions?

(b) Same, but for step 2.

(c) Same, but for step 3.

11.3.5. For $N = 30$, consider the FFT based on the subgroup chain $C_1 \leq C_2 \leq C_{10} \leq C_{30}$ (so $n = 3$).

(a) Draw the subgroup subdiagram for step 1 of this FFT. How many exponentiations ω^{rk} does step 1 require? How many complex multiplications? How many complex additions?

(b) Same, but for step 2.

(c) Same, but for step 3.

11.3.6. For $N = 30$, consider the FFT based on the subgroup chain $C_1 \leq C_5 \leq C_{15} \leq C_{30}$ (so $n = 3$).

(a) Draw the subgroup subdiagram for step 1 of this FFT. How many exponentiations ω^{rk} does step 1 require? How many complex multiplications? How many complex additions?

(b) Same, but for step 2.

(c) Same, but for step 3.

11.4 A "proof of concept" FFT multiplication algorithm

-
- What can you do faster with the FFT?

crude convolution-based multiplication for polynomials

11.5 The Schönhage-Strassen multiplication algorithm

-
- What can you do faster with the FFT?

Schönhage and Strassen.

11.6 The DFT in the language of group theory

To understand why the FFT works, we'll first need to translate our basic definitions about signals, the DFT, and the FFT from writing signals in terms of $\mathbf{Z}/(N) = \{0, \dots, N-1\}$ to writing signals in terms of $G = \langle \omega \rangle = \left\{1, \omega, \dots, \omega^{N-1}\right\}$.

Definition 11.6.1. Fix $N \in \mathbf{N}$, and let $\omega = \omega_N = e^{2\pi i/N}$ and $G = \langle \omega \rangle = \left\{1, \omega, \dots, \omega^{N-1}\right\}$. We define a *signal on* G to be a function $F : G \to \mathbf{C}$, or in other words, a complex-valued function with domain G. Note that any signal $f : \mathbf{Z}/(N) \to \mathbf{C}$ can be rewritten equivalently as a signal $F(g)$ on G by the formula

$$F(\omega^n) = f(n). \tag{11.6.1}$$

(Remember that because the order of ω is N, by Theorem 10.2.2, the expression ω^n in (11.6.1) is unambiguous even though n is only defined mod N.)

One of the first things revealed by changing from signals on $\mathbf{Z}/(N)$ to signals on $G = \langle \omega \rangle$ is that the DFT of a signal on G is best considered not as a signal on G but as a signal on $\mathbf{Z}/(N)$.

Definition 11.6.2. Let N, ω, and G be as in Definition 11.6.1, and let F be a signal on G. We define the *Discrete Fourier Transform*, or *DFT*, of F to be the function $\hat{F} : \mathbf{Z}/(N) \to \mathbf{C}$ given by

$$\hat{F}(k) = \sum_{g \in G} F(g) g^{-k}. \tag{11.6.2}$$

Ask Yourself 11.6.3. Why is the DFT described in Definition 11.6.2 consistent with the DFT described in Definition 9.4.1? In other words, if F is the signal on G corresponding to the signal f on $\mathbf{Z}/(N)$, why is $\hat{F}(k) = \hat{f}(k)$?

Move up to right after FFT; maybe combine with next section?
Point of

$$y(jk) = \sum_{r=0}^{d-1} x(jm + rk) \omega^{-rkj} \tag{11.6.3}$$

is: Let $b = \omega^{rk}$ run over all elements of a transversal. Set the jth element of the subgroup part of the vector to the linear combination of the jth elements of the old vector (mod N/m) translated by the coset rep b with a coefficient of b^{-j}.

11.7 Proof of the FFT

The FFT is such a remarkable idea that even if you're somehow still skeptical about the point of abstraction, you must have wondered: Does the FFT really always work? And if so, why? In other words, as we asked earlier:

Motivating Problem 11.7.1. How can we prove that the FFT *always* produces the right answer?

As we started to explain in Section 11.3, the answer to Motivating Problem 11.7.1 lies in the theory we have developed about subgroup chains, coset representatives, and transversals. We start with the following observation.

Theorem 11.7.2. *Let $K \leq H \leq G$ be a subgroup chain. Suppose $\{a_j \mid 1 \leq i \leq m\}$ is a transversal for H in G and $\{b_k \mid 1 \leq i \leq r\}$ is a transversal for K in H. Then*

$$\{a_j b_k \mid 1 \leq j \leq m,\ 1 \leq k \leq r\} \tag{11.7.1}$$

is a transversal for K in G.

Proof. If G is the disjoint union

$$G = a_1H \cup a_2H \cup \cdots \cup a_mH \tag{11.7.2}$$

and H is the disjoint union

$$H = b_1K \cup b_2K \cup \cdots \cup b_rK \tag{11.7.3}$$

then subtituting (11.7.3) into (11.7.2), we get the disjoint union

$$\begin{aligned} G = {} & (a_1b_1K \cup a_1b_2K \cup \cdots \cup a_1b_rK) \\ & \cup (a_2b_1K \cup a_2b_2K \cup \cdots \cup a_2b_rK) \\ & \vdots \\ & \cup (a_mb_1K \cup a_mb_2K \cup \cdots \cup a_mb_rK). \end{aligned} \tag{11.7.4}$$

The theorem follows. □

When we combine Theorem 11.7.2 with a subgroup chain, we get the following method for expressing the elements of a group as a product of transversals.

Definition 11.7.3. As usual, fix $N \in \mathbf{N}$ and $\omega = e^{2\pi i/N}$, and let

$$C_1 = H_0 \leq H_1 \leq \cdots \leq H_{n-1} \leq H_n = C_N \tag{11.7.5}$$

be a chain of subgroups of C_N that starts at the trivial subgroup $C_1 = \{1\}$ and ends at the group C_N itself. A *factorization* of the subgroup chain (11.7.5) is a choice of a transversal T_i for H_{i-1} in H_i for each $1 \leq i \leq n$.

Note that by Theorem 11.7.2, if $\{a_j\}$ is a transversal for H_{i+1} in H_{i+2} and $\{b_k\}$ is a transversal for H_i in H_{i+1}, then $\{a_jb_k\}$ is a transversal for H_i in H_{i+2}; similarly, a factorization gives a transversal of the form $\{a_jb_kc_\ell\}$ for each H_i in H_{i+3}, and so on. We therefore define the *product transversal* for H_i in H_{i+t} associated to a given factorization to be the transversal obtained as the product of t transversals of that FTS in this manner. Note that by definition, the key property of product transversals is that if $\{a_j\}$ is the product transveral for H_i in H_m and $\{b_k\}$ is the product transversal for H_m in H_r, then $\{a_jb_k\}$ is the product transversal for H_i in H_r.

Example 11.7.4. Factorization used in standard FFT. (Exercise)

In brief, the exponential speedup in the FFT can be explained by the fact that a factorization for the subgroup chain (11.7.5) involves a set S of $O(n) = O(\log N)$ elements such that every element of C_N can be expressed uniquely as a product of a suitably chosen subset of S. Moreover, we can use subgroup chain factorizations to give a concise (if not necessarily easy to understand) proof that the FFT works.

First, we rewrite and generalize the FFT using multiplicative notation and factorized transversal systems as follows.

Definition 11.7.5. Fix $N \in \mathbf{N}$ and $\omega = e^{2\pi i/N}$, let

$$C_1 = H_0 \leq H_1 \leq \cdots \leq H_{n-1} \leq H_n = C_N, \tag{11.7.6}$$

and let $\{T_i \mid 1 \leq i \leq n\}$ be a factorization of the subgroup chain (11.7.6), where T_i is a transversal for H_{i-1} in H_i. Furthermore, for $0 \leq m \leq r$, let S_i be the associated product transversal for H_i in $H_n = G$, where $S_0 = H_n = G$ and $S_n = \{1\}$. Let $m_i = |H_i|$ and $s_i = |S_i|$, and note that since s_i is the number of cosets of H_i in $H_n = C_N$, $m_i s_i = N$.

The *generalized FFT given by the factorization* $\{T_i\}$ is defined by defining the functions $\varphi_i : S_i \times (\mathbf{Z}/(m_i)) \to \mathbf{C}$ $(0 \leq i \leq n)$ recursively as follows. For $i = 0$, let $\varphi_0(g, 0) = F(g)$ for all $g \in C_N$ (as $m_0 = 1$). Then, given φ_{i-1}, for $a \in S_i$ and $k \in \mathbf{Z}/(m_i)$, we define

$$\varphi_i(a, k) = \sum_{b \in T_i} \varphi_{i-1}(ab, k) b^{-k}. \tag{11.7.7}$$

The output of the generalized FFT is then the function $\varphi_n(1, k)$ $(k \in C_N)$.

Example 11.7.6. restricts to previous FFT

key point is that $\varphi_i(1, j)$ is stored in jth element of H_i, i.e., matrix extry $j \cdot$(index of H_i). More generally $\varphi_i(a, j)$ is stored in translate of that by coset rep a.

In general, we see that the generalized FFT takes n steps to compute, each of which involves $O(dN)$ arithmetic operations, for a total complexity of $O(dNn) = O(N \log N)$, assuming d is constant or at least uniformly bounded above (exercise). It therefore remains to show that the output $\varphi_n(1, k)$ is equal to the DFT $\hat{f}(k)$. To prove this fact, we introduce the following interpolation between the initial signal $F(\omega^n)$ and the DFT $\hat{f}(k)$.

Definition 11.7.7. Fix notation as in Definition 11.6.1, and let

$$C_1 = H_0 \leq H_1 \leq \cdots \leq H_{n-1} \leq H_n = G. \tag{11.7.8}$$

Choose a factorization for (11.7.8), let S_i be the associated product transversal for H_i in C_N, and let $m_i = |H_i|$. For $0 \leq i \leq n$, we define the *partial DFT* $\hat{F}_i : S_i \times \mathbf{Z}/(m_i) \to \mathbf{C}$ by the formula

$$\hat{F}_i(a, k) = \sum_{h \in H_i} F(ah) h^{-k} \tag{11.7.9}$$

for each $a \in S_i$ and $k \in \mathbf{Z}/(m_i)$.

When we say that the partial DFT interpolates between F and $\hat{F}$, we mean the following. On the one hand, for $i = 0$, $S_0 = C_N$, $H_0 = \{1\}$, and $m_0 = 1$, we have

$$\hat{F}_0(g, 0) = F(g), \tag{11.7.10}$$

our original signal on G. Conversely, for $i = n$, $S_n = \{1\}$, $H_n = G$, and $m_n = N$, so for $k \in \mathbf{Z}/(N)$, we have

$$\hat{F}_n(1, k) = \sum_{h \in G} F(h) h^{-k} = \hat{f}(k), \tag{11.7.11}$$

the DFT of F. It therefore suffices to prove the following theorem.

Theorem 11.7.8. *Fix notation as in Definition 11.7.7. Then $\varphi_i = \hat{f}_i$ for $0 \leq i \leq n$.*

Proof. We proceed by induction. The base case $i = 0$ follows because $\varphi_0 = F = \hat{F}_0$, so assume by induction that $\varphi_{i-1} = \hat{F}_{i-1}$. As in Definition 11.7.5, for the given factorization $\{T_i\}$, let S_i be the associated product transversal for H_i in G. Then for $a \in S_i$ and $k \in \mathbf{Z}/(m_i)$, we have

$$\begin{aligned}
\varphi_i(a, k) &= \sum_{b \in T_i} \varphi_{i-1}(ab, k) b^{-k} \\
&= \sum_{b \in T_i} \hat{F}_{i-1}(ab, k) b^{-k} && \text{(by induction)} \\
&= \sum_{b \in T_i} \left(\sum_{h \in H_{i-1}} F((ab)h) h^{-k} \right) b^{-k} && (11.7.9) \\
&= \sum_{b \in T_i} \sum_{h \in H_{i-1}} F(a(bh))(bh)^{-k}.
\end{aligned} \tag{11.7.12}$$

One subtlety: For $a \in S_i$ and $b \in T_i$, by Theorem 11.7.2, each ab is an element of the associated product transversal S_{i-1} for H_{i-1} in G, and therefore, $\hat{F}_{i-1}(ab, k)$ is actually defined, making the use legitimate.

Now, since T_i is a transversal for H_{i-1} in H_i, by definition of transversal, if b runs over all elements of T_i and h runs over all elements of H_{i-1}, then $h' = bh$ runs over all elements of H_i. We therefore see from (11.7.12) that

$$\varphi_i(a, k) = \sum_{h' \in H_i} F(ah')(h')^{-k} = \hat{F}_i(a, k). \tag{11.7.13}$$

The theorem follows. □

I don't know about you, but I certainly find it remarkable how short the above proof is — in some ways, it looks like nothing happened! For this book, the proof of the FFT thereby represents the ultimate triumph of abstraction, in that we understood the FFT by

1. Using a different point of view (subgroups and cosets instead of, say, matrices and indices) to see the situation more clearly; and

2. Using the right language (transversals and factorizations) in which to express our proof.

Problems

11.7.1. Fix $N = 30$, let $\omega = e^{2\pi i/N}$, and consider the subgroup chain $H_0 \le H_1 \le H_2$, where $H_0 = \left\langle \omega^{15} \right\rangle$, $H_1 = \left\langle \omega^5 \right\rangle$, and $H_2 = \langle \omega \rangle = C_{30}$.

(a) Find a transversal $\{a_i\}$ for H_1 in H_2.

(b) Find a transversal $\{b_j\}$ for H_0 in H_1.

(c) Write out the cosets $\{a_i b_j H_0\}$ and verify that they partition H_2 (i.e., that H_2 is the disjoint union of those cosets).

11.7.2. Fix $N = 16$, let $\omega = e^{2\pi i/N}$, and consider the subgroup chain $H_0 \le H_1 \le H_2 \le H_3 \le H_4$, where $H_0 = \{1\}$, $H_1 = \left\langle \omega^8 \right\rangle$, $H_2 = \left\langle \omega^4 \right\rangle$, $H_3 = \left\langle \omega^2 \right\rangle$, and $H_4 = \langle \omega \rangle = C_{16}$. (write out product transversals)

11.7.3. Compare $N = 2^{6n}$, base 2, 4, 8.

11.7.4. Compare $N = 2^{6n}$, base 2, 4, 8.

11.8 The FFT on an arbitrary group

representations of a group

FFT on a group

Chapter 12

The big reveal

[SPOILER], I'm not a Republic serial villain. Do you seriously think I'd explain my masterstroke if there remained the slightest chance of you affecting its outcome? I did it thirty-five minutes ago.

— [SPOILER], *Watchmen*, Alan Moore

12.1 Our secret agenda

OK, it's not so secret an agenda, since we discussed it back in the introduction! But the fact is, while everything in this book is motivated by applications, we did actually touch on all of the fundamental objects of a "traditional", proof-centric course in abstract algebra. So, let's pull back the curtain now and look at what we've covered purely in terms of theory.

The fundamental objects of abstract algebra are groups (Definition 10.1.1), rings (Definition 4.2.2), and fields (Definition 4.2.10), with linear algebra (Chapter 5) sometimes appearing as a tool used in the theory of fields. One often-used approach to learning abstract algebra is to introduce those objects in logical order, in the sense of introducing objects with the fewest axioms first. In that order, we have:

- *Groups* have one operation and three axioms (associativity, identity, inverse). *Abelian* groups add the fourth axiom of commutativity of the operation.
- Taking a more careful approach to the definition of ring than we did in Chapter 4, a *ring* R has two operations, $+$ and $\cdot$, with $(R, +)$ being an abelian group (four axioms), $\cdot$ being associative (one axiom), and $+$ and $\cdot$ satisfying the distributive laws (two axioms). A *commutative* ring adds the axiom of commutativity of multiplication, and a ring with *identity* adds the axiom of a multiplicative identity 1 (as opposed to the additive identity 0).
- A *field* is a a commutative ring with identity that also has *inverses*.
- A *vector space* V over a field F is a set V with operations of vector addition $\mathbf{v} + \mathbf{w}$ and scalar multiplication $a\mathbf{v}$ ($a \in F$, $\mathbf{v}, \mathbf{w} \in V$) and a particular element $\mathbf{0}$, all

satisfying the properties listed in Problem 5.3.1. Note that in this approach, F^n and its subspaces are all examples of (abstract) vector spaces, and we can define span, linear independence, and basis just like we did in Chapter 5, except replacing subspaces of F^n with arbitrary vector spaces.

To go further with abstract algebra, even to be an informed consumer, almost certainly the next thing you should do is to focus on theorem-proving, both so you can learn the fundamental theory well and also so you can sharpen your proof skills enough to understand subsequent material. Good textbooks for that purpose include the following.

- Friendly books on the theory of groups, rings, and fields include Gallian [?] and Hungerford [?]. The former book is my favorite introduction to the theory of abstract algebra, and the latter book is particularly in tune with our approach here, in that it considers rings before groups, just like we did.
- Somewhat more adventurous but still well-written books on the theory of abstract algebra include Dummit and Foote [?] and Artin [?]. The latter book (another favorite of mine) is well-known both for having particularly advanced topics more commonly found in graduate algebra textbooks, and also for having *really* hard (but fun!) problems.
- For a friendly introduction to axiom-based linear algebra, try Messer [?], and for a more adventurous but still well-written textbook, see Axler [?].

Also, to take on more sophisticated uses of abstract algebra, you'll probably at least want to get acquainted with some other parts of math, such as:

- Probability and statistics;
- Graph theory; and
- Analysis (the theory of calculus).

Point being, many professional-grade applications of algebra also involve at least a bit of some other aspect(s) of mathematics (see the next section), so it helps to broaden your background. There are many great sources for the above topics, so I leave it as an exercise for the reader to find your favorites! [*]

12.2 What's next?

To wrap up this book, here are just a few topics in applied and industrial algebra that are currently (in 2024) at the forefront of research, organized by subfield of algebra, and with topics chosen to suit my personal tastes (sorry/not sorry).

[*]Though I can't resist one completely self-serving recommendation: [?], while designed to be a textbook used in a second course in analysis, actually works as an introduction to analysis for the ambitious or impatient, but nevertheless covers topics relevant to industrial applications like Fourier series, the Fourier transform, and Hilbert spaces.

- **Groups:** Probably the biggest piece of introductory abstract algebra missing from this book is a detailed study of nonabelian groups. It turns out that there are several areas of active research that generalize applications in this book from abelian groups to nonabelian groups, including:
  - Section 10.4 shows how the difficulty of the Discrete Logarithm Problem in certain finite abelian groups can be used to create secure public-key cryptography schemes. However, if we look at nonabelian groups, there are algorithmic problems that are even more difficult to solve; in fact, there are problems that can be proven to be algorithnmically *impossible* to solve! This idea has become the basis for a whole field of recent work on nonabelian groups in cryptography. For a survey of the field, see Kahrobaei, Flores, and Noce [**?**].
  - Similarly, generalizing the idea that the FFT is an application of the subgroup structure of cyclic groups (Chapter 11), it turns out that there is a particular combinatorial problem in group theory that, if solved for an appropriate family of groups, would provide an algorithm for matrix multiplication that reduces the time needed to multiply two matrices to (in some sense) roughly the same big-O amount of time required to read the relevant data! See the work of Cohn, Umans, and others [**?**, **?**, **?**]).
- **Rings and ideals:** *Algebraic geometry* is the study of solutions to polynomial equations, often (but not always) over a field. Two broad recent areas of applied research in algebraic geometry are:
  - *Algebraic statistics:* It turns out that many useful statistical models can be described as the solution set of a collection of polynomial identities and inequalities. We can therefore use the tools of algebraic geometry to improve statistical methods and analysis. See Améndola, Casanellas, and Garcia Puente [**?**] for an overview of the field.
  - *Elliptic curve cryptography:* See Section 10.4 and the references cited there for an overview.
- **Finite fields and linear algebra:** While we've spent a fair amount of time discussign error-correcting codes, we've really only barely scratched the surface of what's out there. There are a number of texts that serve as good general introductions to the whole field; see, for example, MacWilliams and Sloane [**?**].

  One example of a recent[†] innovation in coding theory is *low-density parity check codes*, which provide error-correction close to various theoretical limits. See MacKay [**?**] for an introduction.
- **All of the above:** The most algebraic area of current industrial research may well be *quantum computing*, which makes use of everything we've discussed and more. Some particularly mathematical highlights include:

[†]Sort of: LDPC codes were invented in 1963, but their usefulness was only recognized starting in the mid-1990s.

- The *quantum Fourier transform,* which, given a (still hypothetical) quantum computer, speeds up the FFT exponentially.[‡] The proverbial killer app for the QFT is *Shor's algorithm* and related algorithms, which can both quickly factor large numbers of the form pq (p, q prime) and also quickly solve the discrete log problem mod p (Section 10.4) — meaning that someone with an effective quantum computer can break most of the commonly used public-key cryptosystems used today.
- To create an effective quantum computer in practice, you have to be able to account for errors caused by "noise" coming from the environment in which your computer sits. That's where *quantum error-correcting codes*, the quantum analogue of the error-correcting codes we've studied here, come in.
- Of course, once your working quantum computer has broken cryptography as we know it, you'll need to look for some way to keep secrets that isn't susceptible to Shor's algorithm. That naturally leads to (what else) *quantum cryptography.*

Like I said, quantum computing really uses it all! For a thorough and very mathematical introduction and overview to the subject, see Nielsen and Chuang [**?**].

Of course, like any proper post-credit scene, the above is really setting up a potential sequel that I may write someday. But for now, I'll just leave you with some wise words about mathematical discovery:

Oh, the movie never ends
It goes on and on and on and on

— Journey, "Don't Stop Believin'"

[‡]That's right, it takes less time to do a QFT than it does to read in the data! In fact, in some sense, the hard part of the algorithm is inputting the data and reading the result.

Appendix A

Unique factorization

A.1 The descending chain condition

blah blah ideal nonsense

A.2 Every PID is a UFD

definition of factoring terminating
 definition of UFD
 lemma: in a PID, irreducibles are prime
 thm: PID is UFD
 nonexample: not a PID
 non UFD (without proof)

Remark A.2.1. This is why there are so few Euclidean domains

Appendix B

Theory of finite fields

B.1 Kernels and the first isomorphism theorem

This chapter builds theory need to prove five facts. Since this isn't needed for the main exposition, mostly inquiry based.

definition of kernel of a ring homomorphism.

Kernel is an ideal.

1IT for rings

Problems

B.1.1.

B.2 The exponent of a finite abelian group

Definition of exponent.

Theorem B.2.1. *Let G be a finite abelian group of order n and exponent k. We have that:*

1. *k divides n; and*
2. *G has an element of order k.*

Remark B.2.2. Classification of finite abelian groups; statement as product of cyclic with each factor dividing the next. Classification implies Theorem B.2.1.

Problems

B.2.1.

B.3 Field extensions

definition of field extensions

field extension containing one root always exists

splitting field always exists.

Problems

B.3.1.

B.4 Proofs of facts about finite fields

definition primitive element

Cor: multiplicative group of a finite field is cyclic.

Proof: exponent

Cor: any finite field is $\mathbf{F}_p[x]/(m(x))$, $m(x)$ irreducible.

proof: substitution homomorphism, plug in primitive element, 1IT.

restate/quote magic polynomial corollary

Everything now stems from the magic polynomial!

lemma: generalize Frob automorphism.

Existence: splitting field of magic polynomial, subfield of $x^q = x$. Subfield (preserves addition) because Frob automorphism.

Uniqueness: If E also has order q, irreducible $m(x)$ for E divides magic polynomial, so $\mathbf{F}_q$ has some root of $m(x)$, generates same field by 1IT.

Problems

B.4.1. field of order 8 isom

B.4.2. finite fields of order 16 are isomorphic.

B.4.3.

B.5 Factoring the magic polynomial

Bonus: factorization of magic polynomial: each irreducible shows up exactly once.

Example: $x^{16} - x$ over $\mathbf{F}_2$.

Lem: Let $q = p^e$, $r = p^d$. Then d divides e if and only if $x^r - x$ divides $x^q - x$. Proof: Work mod $x^{p^d} - x$. If $e = nd$, then x^{p^e} is x raised to the p^d n times. Mod $x^{p^d} - x$, you get $x^{p^e} = x$. If $e = nd + r$, same idea gets a remainder $x^{p^r} - x$, no go.

every irreducible of degree d, where d divides e, divides the magic polynomial $x^q - x$, where $q = p^e$. Proof: $\mathbf{F}_p[x]/(f(x))$ generates field of order p^d, so $f(x)$ divides $x^{p^d} - x$.

Every irreducible factor of $x^q - x$, $q = p^e$ has degree d, where d divides e. Proof: generates subfield of order p^d.

Corollary: subfields are $\mathbf{F}_{p^d}$, where d divides e.

Irreducibles don't repeat: No repeats when you factor into linear factors over $\mathbf{F}_q$, so no repeats when you factor mod p. If there were a repeat mod p, there would be repeated roots over $\mathbf{F}_q$, and there wouldn't be enough factors of the magic polynomial.

Problems: degree 6.

Problems

B.5.1. (a) $x^4 - x$

(b) $x^8 - x$

(c) number of irreducibles of degree 6? $x^{64} - x$

B.5.2. ℓ a prime. How many irreducible polynomials in $\mathbf{F}_p[x]$ of degree ℓ?

Bibliography

Index