Applied and industrial algebra

Tim Hsu, San José State University

February 12, 2019

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

(In case you were wondering, cryptography also falls under the heading of algebra you can use to make money in the real world, but that topic is well-covered in many other books, and it seems like it's about time to give some other material its moment in the sun.)

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.

outline of rest of book

One feature of this book that may strike more experienced readers as unusual is that actual rigorous definitions of objects often appear long after the book has been using those object. For example, the idea of ring first appears in Section 2.1, but the definition of ring is delayed until Chapter 4.

there are 108 problems.

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 | 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 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.

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 and closure 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.

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 $x + y \in S$. □

As you'll see, the above structure is worth learning, as it comes up regularly in this book.

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:*

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

In other words, the rationals $\mathbf{Q}$ 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* $\mathbf{R}$ 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* $\mathbf{C}$, 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 8.

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

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 context. 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!

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. However, we're still warming up here, so instead, let's start with what looks 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 is 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 4.1.1.

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 $|-10000| > |3|$. If we use the term "greater" in this sense of greater absolute value, then (for example) $\gcd(6, 10)$ could be either $+2$ or -2. This $\pm$ ambiguity turns out to be useful later, so we hereby set the convention that we only need to compute $\gcd(a, b)$ up to associates (Definition 2.1.6), i.e., up to $\pm$.

Remark 2.2.6. I know, I promised that this book would focus on things you can use to make money, and now so much 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 an answer of -2 instead of 2.

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 K–12 knowledge of divisors (see Problem 2.2.2). 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. The goal of this problem is to refine Naive Algorithm 2.2.9 so that we can be sure it needs fewer steps than indicated by the analysis given above.

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

(b) Explain how to modify Naive Algorithm 2.2.9 so that if a, b, and n are positive integers, with $a, b \leq n$, then the number of divisions required to compute $\gcd(a, b)$ is no more than $C\sqrt{n}$, where C is some constant. (Suggestion: Find an upper bound for the number of divisors of an integer of size at most n.)

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

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

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; if you're familiar with proof by induction, you may 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 $q = \left\lfloor \frac{a}{d} \right\rfloor$. 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. Suppose a and d are positive integers, and suppose also that q_i and r_i $(i = 1, 2)$ are nonnegative integers such that
$$a = q_1 d + r_1, \qquad\qquad a = q_2 d + r_2, \tag{2.3.10}$$
with $0 \leq r_1, r_2 < d$. Prove that $r_1 = r_2$ and $q_1 = q_2$. (Suggestion: How big can $|r_2 - r_1|$ be?)

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

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

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 money-making thing, 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,

$$r_{i-2} = q_i r_{i-1} + r_i \qquad \text{with } 0 \leq |r_i| \leq \frac{|r_{i-1}|}{2}. \tag{2.4.5}$$

   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

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.)

Example 2.5.2. We use Euclidean Rewriting to find an integer solution to $105x + 39y = \gcd(105, 39)$. First off, the Euclidean Algorithm gives:

$$\begin{aligned} 105 &= 2(39) + 27 \\ 39 &= 1(27) + 12 \\ 27 &= 2(12) + 3 \\ 12 &= 4(3), \end{aligned} \tag{2.5.4}$$

so $\gcd(39, 27) = 3$. Keeping in mind that $a = 105$ and $b = 39$, and rewriting starting from the first equation, we see that

$$\begin{aligned} 27 &= 105 - 2(39) = a - 2b, \\ 12 &= 39 - 1(27) = b - 1(a - 2b) = 3b - a, \\ 3 &= 27 - 2(12) = (a - 2b) - 2(3b - a) = 3a - 8b. \end{aligned} \tag{2.5.5}$$

So our final answer is $x = 3$, $y = -8$; or in other words, $3(105) - 8(39) = 3$. (Check this!)

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

Corollary 2.5.3. *Let a and b be nonzero integers. For $c \in \mathbf{Z}$, the equation*

$$ax + by = \gcd(a, b) \tag{2.5.6}$$

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 x and y, 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 $a(xq) + b(yq) = c$. □

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 relatively simple "no-brainer" 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 the corresponding no-brainer method. We can sometimes measure this kind of speed-up by doing computer experiments, but it 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 $\lim_{n\to\infty}$ 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 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?)

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

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

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,

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

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.*

[‡]Again, really, analysis.

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!

We close out this chapter by finally getting around to explaining how you can use the Euclidean Algorithm to make money.

Real-life Application 2.6.9. 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!

Unsolved Problem 2.6.10. What other numerical problems, besides factoring, can you reduce to the Euclidean Algorithm? See (list of references) [].

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. (Take the operation of adding two single-digit numbers as your fundamental unit of time.)

(b) Same, but for multiplying a and b. (Take the operation of either adding or multiplying two single-digit numbers as your unit of time.)

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 amnount 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 $n \geq 1$, $r_n \leq \dfrac{r_{n-2}}{2}$. (Suggestion: Consider the cases $r_{n-1} \geq \dfrac{r_{n-2}}{2}$ and $r_{n-1} < \dfrac{r_{n-2}}{2}$.)

(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.

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 the side of Mr. Seeger 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 n

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 n)*.

As we will see in Chapter 4, to define a ring, we need to define an underlying set and define some notion of addition and multiplication on that set. We start with a preliminary definition.

Definition 3.1.1. Let n be a positive integer. For any integer k, k *reduced (mod n)* is the remainder you get when you divide k by n. In other words, if, for some $q, r \in \mathbf{Z}$,

$$k = qn + r \qquad \text{with } 0 \leq r < n, \tag{3.1.1}$$

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

Note that the uniqueness part of the Division Theorem 2.3.1 is what makes Definition 3.1.1 unambigous.

Definition 3.1.2. Let n be a positive integer. We define the ring $\mathbf{Z}/(n)$, or the *integers (mod n)*, as follows.

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

Definition 3.1.2 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 $\mathbf{Z}/(n)$, so we introduce the following idea.

Definition 3.1.3. Let n be a positive integer. To say that two integers a and b are *congruent (mod n)*, or equivalently, that a is equal to b (mod n), means that $b - a$ is divisible by n. Put another way, a is equal to b (mod n) exactly when

$$a = qn + b \tag{3.1.2}$$

for some integer q.

So, for example, the reduction of k (mod n) (Definition 3.1.1) is congruent to k (mod n). More generally, we have the following *Congruent Substitution Principle*.

> **Congruent Substitution Principle:** If we are working in the ring $\mathbf{Z}/(n)$, we can always replace any integer a with any integer congruent to a (mod n).

Even more generally, we have that:

The $n = 0$ Principle: Arithmetic in $\mathbf{Z}/(n)$ is like regular arithmetic, except that we declare that $n = 0$, 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 $n = 0$ Principle).

Example 3.1.4. 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 n is odd, and we're working on something multiplicative, it can be helpful to list the elemnts of $\mathbf{Z}/(n)$ using negatives. For example, the elements of $\mathbf{Z}/(11)$ can be written as:

$$\begin{aligned}\{0,1,2,3,4,5,6,7,8,9,10\} &= \{0,1,2,3,4,5,-5,-4,-3,-2,-1\} \\ &= \{-5,-4,-3,-2,-1,0,1,2,3,4,5\}\,.\end{aligned} \tag{3.1.3}$$

Example 3.1.5. In $\mathbf{Z}/(13)$, $2(7) = 1$, so we can think of 7 as being equal to $\dfrac{1}{2}$ 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:

$$\frac{1}{8} = \left(\frac{1}{2}\right)^3 = 7^3 = 343 = 5, \tag{3.1.4}$$

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

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

Ask Yourself 3.1.6. On the one hand, will the Congruence Substitution Principle, or more generally, the $n = 0$ Principle, always work consistently in $\mathbf{Z}/(n)$, 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 3.1.2, 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 3.1.6 is yes. However, we'll need many more layers of abstract nonsense to prove that fact efficiently, so we'll wait to do that until Chapter 6. Until then, several problems below give you a chance to play around in $\mathbf{Z}/(p)$ (p a prime) with the following interesting idea.

Definition 3.1.7. 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.)

Problems

3.1.1. Let n be a positive integer, and let k be an arbitrary integer. Prove that k is congruent mod n 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 $\mathbf{Z}/(n)$.

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

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

3.1.3. This problem relies on Definition 3.1.7.

(a) Experiment and look for a pattern: How many residues are there mod 7? Mod 11? Mod 13, 17, 19, 23, 29, 31, 37? (Suggestion: For p an odd prime, try squaring all elements of $\mathbf{Z}/(p)$, and remember that residues are by definition nonzero.)

(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 data from the previous part.)

3.1.4. For which odd primes p is -1 a quadratic residue (Definition 3.1.7)? Try the odd primes up to 32 and see if you can find a pattern in which p work and which don't. (Suggestion: Try to solve $x^2 = -1$ in $\mathbf{Z}/(p)$ by listing all squares in $\mathbf{Z}/(p)$, as in Problem 3.1.3.)

3.1.5. For which odd primes p is 2 a quadratic residue (Definition 3.1.7)? Try the odd primes up to 64 and see if you can find a pattern in which p work and which don't. (Suggestion: Try to solve $x^2 = 2$ in $\mathbf{Z}/(p)$ by listing all squares in $\mathbf{Z}/(p)$, as in Problem 3.1.3.)

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

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

$$\begin{aligned} & ax = b \text{ in } \mathbf{Z}/(n) \\ \Leftrightarrow & ax = qn + b \text{ in } \mathbf{Z}, \text{ for some } n \in \mathbf{Z} \\ \Leftrightarrow & ax + ny = b \text{ in } \mathbf{Z}, \text{ for some } y \in \mathbf{Z}, \end{aligned} \tag{3.2.1}$$

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

Therefore, in $\mathbf{Z}/(n)$, Bezout's Identity 2.5.1 and Corollary 2.5.3 become:

Corollary 3.2.2. *For $a, b \in \mathbf{Z}/(n)$, $ax = b$ has a solution $x \in \mathbf{Z}/(n)$ exactly when $\gcd(a, n)$ 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 $26x = 6$ in $\mathbf{Z}/(34)$. The point of (3.2.1) is that solving $26x = 6$ in $\mathbf{Z}/(34)$ is equivalent to solving

$$26x + 34y = 6 \tag{3.2.2}$$

in $\mathbf{Z}$, and then ignoring y. To solve (3.2.2), we use the Euclidean Algorithm to find $\gcd(34, 26)$:

$$\begin{aligned} 34 &= 1(26) + 8 \\ 26 &= 3(8) + 2 \\ 8 &= 4(2). \end{aligned} \tag{3.2.3}$$

Applying Euclidean Reduction with $a = 34$, $b = 26$, we get

$$\begin{aligned} 8 &= 34 - 1(26) = a - b \\ 2 &= 26 - 3(8) = b - 3(8) = b - 3(a - b) = 4b - 3a. \end{aligned} \tag{3.2.4}$$

Therefore, $4(26) - 3(34) = 2$, which mean that $26(4) = 2$ in $\mathbf{Z}/(34)$. (Remember, in $\mathbf{Z}/(34)$, $34 = 0$!). Multiplying both sides by 3, we get that $26(12) = 6$ in $\mathbf{Z}/(34)$, i.e., $x = 12$ is one solution to $26x = 6$ in $\mathbf{Z}/(34)$. (If you're curious, the reason we keep saying "one solution" is that there's another solution, namely, $x = 29 = -5$, 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 $x \in \mathbf{Z}/(p)$.*

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 $\mathbf{Z}/(n)$ is for n not prime. This calls, of course, for some definitions.

Definition 3.2.5. 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 $\mathbf{Z}/(n)$.) 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 in the integers because $\frac{1}{2}$ is not an integer.

Definition 3.2.6. 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 $\mathbf{C}$. The significance of Corollary 3.2.4 is that the ring $\mathbf{Z}/(p)$ is also a field. We indicate the importance of the fact that $\mathbf{Z}/(p)$ is a field by calling it by the following alternate name.

Definition 3.2.7. 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.) Alas, as we shall see, the term *order* is overused in algebra, but here it refers to the fact that there are p elements in $\mathbf{Z}/(p)$.

Problems

3.2.1. Use the Euclidean Algorithm either to find one solution for the following linear equations in $\mathbf{Z}/(n)$, for the indicated n, 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 n is a fixed positive integer, a and b are nonzero elements of $\mathbf{Z}/(n)$, and x is a solution to $ax = b$ in $\mathbf{Z}/(n)$. Let $d = \dfrac{n}{\gcd(a, n)}$. Prove that for any integer k, $x + kd$ is also a solution to $ax = b$ in $\mathbf{Z}/(n)$. (Conversely, one can prove that this is the complete list of solutions to $ax = b$ in $\mathbf{Z}/(n)$.)

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 $b = c$. (Suggestion: Consider cab.)

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 $\mathbf{Z}/(n)$:

- Make a list of the set $U(n)$ of all units in $\mathbf{Z}/(n)$. (Suggestion: According to Corollary 3.2.2, when does the equation $ax = 1$ have a solution in $\mathbf{Z}/(n)$?)
- Make a "multiplication table" for $U(n)$. That is, make a table whose rows and columns are labelled with the elements of $U(n)$, and in the entry corresponding to the row labelled a and the column labelled b, write in the element ab (reduced mod n).

(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 $R = \mathbf{Z}$, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{C}$, $\mathbf{Z}/(n)$) 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 $\mathbf{Z}$, $\mathbf{Q}$, $\mathbf{R}$, $\mathbf{C}$, $\mathbf{Z}/(n)$ 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

$$\sum_{i=1}^{n} a_i x^i = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0, \tag{3.3.1}$$

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:

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

where all additions use the addition operation from R.

Similarly, we define the product of two polynomials by:

$$\begin{array}{rrrrrrrrrrrrrrr} & & & & & & & & a_n x^n & + & \cdots & + & a_2 x^2 & + & a_1 x + a_0 \\ & & & & & & & & b_k x^k & + & \cdots & + & b_2 x^2 & + & b_1 x + b_0 \\ \hline & & & & & & & & a_n b_0 x^n & + & \cdots & + & a_2 b_0 x^2 & + & a_1 b_0 x + a_0 b_0 \\ & & & & & & a_n b_1 x^{n+1} & + & \cdots & + & a_2 b_1 x^3 & + & a_1 b_1 x^2 & + & a_0 b_1 x \\ & & & a_n b_2 x^{n+2} & + & \cdots & & + & a_2 b_2 x^4 & + & a_1 b_2 x^3 & + & a_0 b_2 x^2 & & \\ & \iddots & & & \iddots & & & \iddots & & & \iddots & & \iddots & & \\ \hline a_n b_k x^{n+k} & + & & & & \cdots & & & & & & + & c_2 x^2 & + & c_1 x + c_0 \end{array}$$

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

$$\begin{aligned} c_0 &= a_0 b_0 \\ c_1 &= a_1 b_0 + a_0 b_1 \\ c_2 &= a_2 b_0 + a_1 b_1 + a_2 b_0 \\ &\vdots \\ c_m &= \sum_{i+j=m} a_i b_j = a_m b_0 + \cdots + a_0 b_m \\ &\vdots \\ c_{n+k} &= a_n b_k \end{aligned} \tag{3.3.3}$$

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, all of the above nonsense 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.

Example 3.3.2. So you can get a feel for multiplication of polynomials with coefficients in a ring other than ordinary real numbers or complex numbers, consider $p(x) = 5x^3-3x^2+2x+7$ and $q(x) = 4x^2 + 3x - 2$ 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.

$$\begin{array}{r} 5x^3 - 3x^2 + 2x + 7 \\ 4x^2 - 3x - 2 \\ \hline 3x^3 + 6x^2 - 4x - 1 \\ -2x^4 - 4x^3 - 6x^2 + 5x \\ 7x^5 + x^4 + 8x^3 + 2x^2 \\ \hline 7x^5 - x^4 + 7x^3 + 2x^2 + x^2 - 1 \end{array} \tag{3.3.4}$$

In this calculation, we use the Congruence Substitution Principle freely (and a little bit at whim) to make coefficients smaller. For example, we get the line $3x^3 + 6x^2 - 4x - 1$ by using the fact that $-14 = -1$ and $-10 = 3$ in $\mathbf{F}_{13}$. Mini-exercise: Check for yourself that the rest of the calculation is done correctly, as long as $13 = 0$.

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 3.3.3. Let $f(x) = a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$, 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_kx^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 3.3.6). This statement is not hard to prove, except for the slight hitch that it isn't true, as the following example shows.

Example 3.3.4. 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 3.3.6 actually work, we need to make an additional assumption about the coefficient ring R.

Definition 3.3.5. 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 4.2.1). As for our other favorite example $\mathbf{Z}/(n)$, if n is not prime (i.e., $n = ab$ for integers $a, b > 1$), then $\mathbf{Z}/(n)$ 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 3.3.6. *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 3.3.6 has a number of useful consequences, such as the following corollaries, whose proofs are left to you.

Corollary 3.3.7. *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 3.3.8. *Let R be a ring with the zero factor property, and suppose $u(x)$ is a unit (Definition 3.2.5) in $R[x]$. Then $u(x)$ is actually 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).

(a) $(9x^3 + 10x^2 + 3x - 7)(2x^2 - 5x + 3)$ in $\mathbf{F}_{19}[x]$.
(b) $(9x^3 + 10x^2 + 3x - 7)(2x^2 - 5x + 3)$ in $\mathbf{F}_{17}[x]$.
(c) $(x^7 + x^5 + x^4 + x^3 + x + 1)(x^4 + x^3 + x^2 + 1)$ in $\mathbf{F}_2[x]$.
(d) $(x^9 + x^6 + x^5 + x^3 + x^2 + x + 1)(x^5 + x^4 + 1)$ in $\mathbf{F}_2[x]$.

3.3.2. This problem considers the zero factor property (Definition 3.3.5).

(a) Prove (by giving an example) that $\mathbf{Z}/(6)$ does not have the zero factor property.
(b) Prove that if n is *composite* (i.e., $n = ab$ for some integers $a, b > 1$), then $\mathbf{Z}/(n)$ does not have the zero factor property.

3.3.3. 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 $\deg(g(x)) \leq \dfrac{\deg(f(x))}{2}$.

3.3.4. 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 are there in $\mathbf{F}_p[x]$? Describe how you would list all of them.
(c) How many monic polynomials of degree ≤ 6 *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. The details are important enough that we'll express them as a theorem.

Theorem 3.4.1 (The Division Theorem)**.** *Let a and d be positive integers. There exist unique nonnegative integers q and r such that*

$$a = dq + r, \qquad \text{with } 0 \leq r < d. \tag{3.4.1}$$

long division
remainder theorem, factor theorem
degree n has at most n roots

Problems

3.4.1. polynomial long division

3.4.2. degree 3 factoring

3.4.3. small irreducibles

3.5 The Euclidean algorithm for polynomials

Definition 3.5.1. GCD (not obviously unique!!)

mo prob: right way?
what made it work before?
euc algorithm
proof
cor: divides
cor: GCD unique up to associates

Real-life Application 3.5.2. factoring
fact: derivative
So provides first step in factoring

Problems

3.5.1. compute GCD

3.5.2. Prove euc algorithm works

3.5.3. How long does it take? long div, then GCD

3.6 Bezout's identity for polynomials

Third verse, same as the first: Repeat Section 2.5.

Theorem 3.6.1 (Bezout's Identity for polynomials)**.** *Let $a(x)$ and $b(x)$. The equation*

$$ax + by = \gcd(a, b) \tag{3.6.1}$$

has a solution

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

$$r_1 = r_{-1} - q_1 r_0. \tag{3.6.2}$$

Substituting $r_{-1} = a$ and $r_0 = b$, and combining coefficients on the right-hand side of (3.6.2), 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

$$r_2 = r_0 - q_2 r_1, \tag{3.6.3}$$

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.)

Example 3.6.2. We use Euclidean Rewriting to find an integer solution to

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

Corollary 3.6.3. *Let a and b be nonzero integers. For $c \in \mathbf{Z}$, the equation*

$$ax + by = \gcd(a, b) \tag{3.6.4}$$

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

Problems

3.6.1. compute Bezout

Index