8 Number Theory

Number theory is the study of the integers. *Why* anyone would want to study the integers is not immediately obvious. First of all, what’s to know? There’s 0, there’s 1, 2, 3, and so on, and, oh yeah, -1, -2, ... Which one don’t you understand? Second, what practical value is there in it?

The mathematician G. H. Hardy delighted at its impracticality~~;~~ he wrote:

> [Number theorists] may be justified in rejoicing that there is one science, at any rate, and that their own, whose very remoteness from ordinary human activities should keep it gentle and clean.

Hardy was ~~specially~~ concerned that number theory not be used in warfare; he was a pacifist. You may applaud his sentiments, but he got it wrong: number theory underlies modern cryptography, which is what makes secure online communication possible. Secure communication is of course crucial in war ~~— which may leave~~ poor Hardy spinning in his grave. It’s also central to online commerce. Every time you buy a book from Amazon, use a certificate to access a web page, or use a PayPal account, you are relying on number theoretic algorithms.

Number theory also provides an excellent environment for us to practice and apply the proof techniques that we developed in previous chapters. We’ll work out properties of greatest common divisors (gcd’s) and use them to prove that integers factor uniquely into primes. Then we’ll introduce modular arithmetic and work out enough of its properties to explain the RSA public key crypto-system.

Since we’ll be focusing on properties of the integers, we’ll adopt the default convention in this chapter that **variables range over the set, $\mathbb{Z}$, of integers**.

8.1 Divisibility

The nature of number theory emerges as soon as we consider the *divides* relation.

Definition 8.1.1. *a divides b* (notation $a \mid b$) iff there is an integer k such that

$$ak = b.$$

The divides relation comes up so frequently that multiple synonyms for it are used all the time. The following phrases all say the same thing:

- $a \mid b$,
- a divides b,
- a is a *divisor* of b,
- a is a *factor* of b,
- b is *divisible* by a,
- b is a *multiple* of a.

Some immediate consequences of Definition 8.1.1 are that for all n

$$n \mid 0, \qquad n \mid n, \text{ and } \qquad \pm 1 \mid n.$$

Also,

$$0 \mid n \text{ ~~IMPLIES~~ } n = 0.$$

Dividing seems simple enough, but let’s play with this definition. The Pythagoreans, an ancient sect of mathematical mystics, said that a number is *perfect* if it equals the sum of its positive integral divisors, excluding itself. For example, $6 = 1 + 2 + 3$ and $28 = 1 + 2 + 4 + 7 + 14$ are perfect numbers. On the other hand, 10 is not perfect because $1 + 2 + 5 = 8$, and 12 is not perfect because $1 + 2 + 3 + 4 + 6 = 16$. Euclid characterized all the *even* perfect numbers around 300 BC (see Problem 8.3). But is there an *odd* perfect number? More than two thousand years later, we still don’t know! All numbers up to about 10^{300} have been ruled out, but no one has proved that there isn’t an odd perfect number waiting just over the horizon.

So a half-page into number theory, we’ve strayed past the outer limits of human knowledge. This is pretty typical; number theory is full of questions that are easy to pose, but incredibly difficult to answer. We’ll mention a few more such questions in later sections.[1]

8.1.1 Facts about Divisibility

The following lemma collects some basic facts about divisibility.

Lemma 8.1.2.

1. *If $a \mid b$ and $b \mid c$, then $a \mid c$.*

[1] *Don’t Panic* —we’re going to stick to some relatively benign parts of number theory. These super-hard unsolved problems rarely get put on problem sets.

2. *If $a \mid b$ and $a \mid c$, then $a \mid sb + tc$ for all s and t.*

3. *For all $c \neq 0$, $a \mid b$ if and only if $ca \mid cb$.*

Proof. These facts all follow directly from Definition 8.1.1. To illustrate this, we'll prove just part 2:

Given that $a \mid b$, there is some $k_1 \in \mathbb{Z}$ such that $ak_1 = b$. Likewise, $ak_2 = c$, so

$$sb + tc = s(k_1a) + t(k_2a) = (sk_1 + tk_2)a.$$

Therefore $sb + tc = k_3a$ where $k_3 ::= (sk_1 + tk_2)$, which means that

$$a \mid sb + tc.$$

■

A number of the form $sb + tc$ is called an *integer linear combination* of b and c, or, since in this chapter we're only talking about integers, just a *linear combination*. So Lemma 8.1.2.2 can be rephrased as

If a divides b and c, then a divides every linear combination of b and c.

We'll be making good use of linear combinations, so let's get the general definition on record:

Definition 8.1.3. An integer n is a *linear combination* of numbers $b_0, \ldots, b_k$ iff

$$n = s_0b_0 + s_1b_1 + \cdots + s_kb_k$$

for some integers $s_0, \ldots, s_k$.

8.1.2 When Divisibility Goes Bad

As you learned in elementary school, if one number does *not* evenly divide another, you get a "quotient" and a "remainder" left over. More precisely:

Theorem 8.1.4. *[Division Theorem][2] Let n and d be integers such that $d > 0$. Then there exists a unique pair of integers q and r, such that*

$$n = q \cdot d + r \text{ AND } 0 \leq r < d. \tag{8.1}$$

[2]This theorem is often called the "Division Algorithm," but we prefer to call it a theorem since it does not actually describe a division procedure for computing the quotient and remainder.

The number q is called the *quotient* and the number r is called the *remainder* of n divided by d. We use the notation $\text{qcnt}(n, d)$ for the quotient and $\text{rem}(n, d)$ for the remainder. For example, $\text{qcnt}(2716, 10) = 271$ and $\text{rem}(2716, 10) = 6$, since $2716 = 271 \cdot 10 + 6$. Similarly, $\text{rem}(-11, 7) = 3$, since $-11 = (-2) \cdot 7 + 3$.

There is a remainder operator built into many programming languages. For example, "32 % 5" will be familiar as remainder notation to programmers in Java, C, and C++; it evaluates to $\text{rem}(32, 5) = 2$ in all three languages. On the other hand, these languages treat remainders involving negative numbers idiosyncratically, so if you program in one those languages, remember to stick to the definition according to the Division Theorem 8.1.4.

The remainder on division by n is a number in the (integer) *interval* from 0 to $n-1$. Such intervals come up so often that it is useful to have a simple notation for them.

$$\begin{aligned}
(k, n) &::= \{i \mid k < i < n\}, \\
(k, n] &::= (k, n) \cup \{n\}, \\
[k, n) &::= \{k\} \cup (k, n), \\
[k, n] &::= \{k\} \cup (k, n) \cup \{n\} = \{i \mid k \le i \le n\}.
\end{aligned}$$

8.1.3 Die Hard

Die Hard 3 is just a B-grade action movie, but we think it has an inner message: everyone should learn at least a little number theory. In Section 5.4.4, we formalized a state machine for the Die Hard jug-filling problem using 3 and 5 gallon jugs, and also with 3 and 9 gallon jugs, and came to different conclusions about bomb explosions. What's going on in general? For example, how about getting 4 gallons from 12- and 18-gallon jugs, getting 32 gallons with 899- and 1147-gallon jugs, or getting 3 gallons into a jug using just 21- and 26-gallon jugs?

It would be nice if we could solve all these silly water jug questions at once. This is where number theory comes in handy.

A Water Jug Invariant

Suppose that we have water jugs with capacities a and b with $b \ge a$. Let's carry out some sample operations of the state machine and see what happens, assuming the b-jug is big enough:

$$\begin{aligned}
(0,0) &\to (a,0) && \text{fill first jug}\\
&\to (0,a) && \text{pour first into second}\\
&\to (a,a) && \text{fill first jug}\\
&\to (2a-b,b) && \text{pour first into second (assuming } 2a \ge b)\\
&\to (2a-b,0) && \text{empty second jug}\\
&\to (0,2a-b) && \text{pour first into second}\\
&\to (a,2a-b) && \text{fill first}\\
&\to (3a-2b,b) && \text{pour first into second (assuming } 3a \ge 2b)
\end{aligned}$$

What leaps out is that at every step, the amount of water in each jug is a linear combination of a and b. This is easy to prove by induction on the number of transitions:

Lemma 8.1.5 (Water Jugs)**.** *In the* Die Hard *state machine of Section 5.4.4 with jugs of sizes a and b, the amount of water in each jug is always a linear combination of a and b.*

Proof. The induction hypothesis, $P(n)$, is the proposition that after n transitions, the amount of water in each jug is a linear combination of a and b.

Base case ($n = 0$): $P(0)$ is true, because both jugs are initially empty, and $0 \cdot a + 0 \cdot b = 0$.

Inductive step: Suppose the machine is in state (x, y) after n steps, that is, the little jug contains x gallons and the big one contains y gallons. There are two cases:

- If we fill a jug from the fountain or empty a jug into the fountain, then that jug is empty or full. The amount in the other jug remains a linear combination of a and b. So $P(n + 1)$ holds.
- Otherwise, we pour water from one jug to another until one is empty or the other is full. By our assumption, the amount x and y in each jug is a linear combination of a and b before we begin pouring. After pouring, one jug is either empty (contains 0 gallons) or full (contains a or b gallons). Thus, the other jug contains either $x + y$ gallons, $x + y - a$, or $x + y - b$ gallons, all of which are linear combinations of a and b since x and y are. So $P(n + 1)$ holds in this case as well.

Since $P(n + 1)$ holds in any case, this proves the inductive step, completing the proof by induction. ■

So we have established that the jug problem has a preserved invariant, namely, the amount of water in every jug is a linear combination of the capacities of the jugs. Lemma 8.1.5 has an important corollary:

Corollary. *Getting 4 gallons from 12- and 18-gallon jugs, and likewise getting 32 gallons from 899- and 1147-gallon jugs,*

Bruce dies!

Proof. By the Water Jugs Lemma 8.1.5, with 12- and 18-gallon jugs, the amount in any jug is a linear combination of 12 and 18. This is always a multiple of 6 by Lemma 8.1.2.2, so Bruce can't get 4 gallons. Likewise, the amount in any jug using 899- and 1147-gallon jugs is a multiple of 31, so he can't get 32 either. ■

But the Water Jugs Lemma doesn't tell the complete story. For example, it leaves open the question of getting 3 gallons into a jug using just 21- and 26-gallon jugs: the only positive factor of both 21 and 26 is 1, and of course 1 divides 3, so the Lemma neither rules out nor confirms the possibility of getting 3 gallons.

A bigger issue is that we've just managed to recast a pretty understandable question about water jugs into a technical question about linear combinations. This might not seem like a lot of progress. Fortunately, linear combinations are closely related to something more familiar, ~~namely~~ greatest common divisors, and these will help us solve the general water jug problem.

8.2 The Greatest Common Divisor

A *common divisor* of a and b is a number that divides them both. The *greatest common divisor* of a and b is written $\gcd(a, b)$. For example, $\gcd(18, 24) = 6$.

As long as a and b are not both 0, they will have a gcd. The gcd turns out to be a very valuable piece of information about the relationship between a and b and for reasoning about integers in general. We'll be making lots of use of gcd's in what follows.

Some immediate consequences of the definition of gcd are that for $n > 0$,

$$\gcd(n, n) = n, \qquad \gcd(n, 1) = 1, \text{and} \qquad \gcd(n, 0) = n,$$

where the last equality follows from the fact that everything is a divisor of 0.

8.2.1 Euclid's Algorithm

The first thing to figure out is how to find gcd's. A good way called *Euclid's Algorithm* has been known for several thousand years. It is based on the following elementary observation.

Lemma 8.2.1. *For $b \neq 0$,*

$$\gcd(a, b) = \gcd(b, \text{rem}(a, b)).$$

Proof. By the Division Theorem 8.1.4,

$$a = qb + r \tag{8.2}$$

where $r = \text{rem}(a, b)$. So a is a linear combination of b and r, which implies that any divisor of b and r is a divisor of a by Lemma 8.1.2.2. Likewise, r is a linear combination, $a - qb$, of a and b, so any divisor of a and b is a divisor of r. This means that a and b have the same common divisors as b and r, and so they have the same *greatest* common divisor. ■

Lemma 8.2.1 is useful for quickly computing the greatest common divisor of two numbers. For example, we could compute the greatest common divisor of 1147 and 899 by repeatedly applying it:

$$\begin{aligned}
\gcd(1147, 899) &= \gcd\left(899, \underbrace{\text{rem}(1147, 899)}_{=248}\right) \\
&= \gcd(248, \text{rem}(899, 248) = 155) \\
&= \gcd(155, \text{rem}(248, 155) = 93) \\
&= \gcd(93, \text{rem}(155, 93) = 62) \\
&= \gcd(62, \text{rem}(93, 62) = 31) \\
&= \gcd(31, \text{rem}(62, 31) = 0) \\
&= 31
\end{aligned}$$

This calculation that $\gcd(1147, 899) = 31$ was how we figured out that with water jugs of sizes 1147 and 899, Bruce dies trying to get 32 gallons.

On the other hand, applying Euclid's algorithm to 26 and 21 gives

$$\gcd(26, 21) = \gcd(21, 5) = \gcd(5, 1) = 1,$$

so we can't use the reasoning above to rule out Bruce getting 3 gallons into the big jug. As a matter of fact, because the gcd here is 1, Bruce *will* be able to get any number of gallons into the big jug up to its capacity. To explain this, we will need a little more number theory.

Euclid's Algorithm as a State Machine

~~By the way,~~ Euclid's algorithm can easily be formalized as a state machine. The set of states is $\mathbb{N}^2$ and there is one transition rule:

$$(x, y) \longrightarrow (y, \text{rem}(x, y)), \tag{8.3}$$

for $y > 0$. By Lemma 8.2.1, the gcd stays the same from one state to the next. That means the predicate

$$\gcd(x, y) = \gcd(a, b)$$

is a preserved invariant on the states (x, y). This preserved invariant is, of course, true in the start state (a, b). So by the Invariant Principle, if y ever becomes 0, the invariant will be true and so

$$x = \gcd(x, 0) = \gcd(a, b).$$

Namely, the value of x will be the desired gcd.

What's more, x, and therefore also y, gets to be 0 pretty fast. To see why, note that starting from (x, y), two transitions leads to a state whose the first coordinate is $\text{rem}(x, y)$, which is at most half the size of x.[3] Since x starts off equal to a and gets halved or smaller every two steps, it will reach its minimum value —which is $\gcd(a, b)$ —after at most $2 \log a$ transitions. After that, the algorithm takes at most one more transition to terminate. In other words, Euclid's algorithm terminates after at most $1 + 2 \log a$ transitions.[4]

8.2.2 The Pulverizer

We will get a lot of mileage out of the following key fact:

Theorem 8.2.2. *The greatest common divisor of a and b is a linear combination of a and b. That is,*

$$\gcd(a, b) = sa + tb,$$

for some integers s and t.

We already know from Lemma 8.1.2.2 that every linear combination of a and b is divisible by any common factor of a and b, so it is certainly divisible by the greatest

[3] In other words,

$$\text{rem}(x, y) \le x/2 \qquad \text{for } 0 < y \le x. \tag{8.4}$$

This is immediate if $y \le x/2$, since the remainder of x divided by y is less than y by definition. On the other hand, if $y > x/2$, then $\text{rem}(x, y) = x - y < x/2$.

[4] A tighter analysis shows that at most $\log_\varphi(a)$ transitions are possible where φ is the *golden ratio* $(1 + \sqrt{5})/2$, see Problem 8.10.

of these common divisors. Since any constant multiple of a linear combination is also a linear combination, Theorem 8.2.2 implies that any multiple of the gcd is a linear combination, giving:

Corollary 8.2.3. *An integer is a linear combination of a and b iff it is a multiple of* $\gcd(a, b)$.

We'll prove Theorem 8.2.2 directly by explaining how to find s and t. This job is tackled by a mathematical tool that dates back to sixth-century India, where it was called *kuttak*, which means "The Pulverizer." Today, the Pulverizer is more commonly known as "the extended Euclidean GCD algorithm," because it is so close to Euclid's Algorithm.

For example, following Euclid's Algorithm, we can compute the GCD of 259 and 70 as follows:

$$\begin{aligned}
\gcd(259, 70) &= \gcd(70, 49) && \text{since rem}\,(259,\ 70) = 49\\
&= \gcd(49, 21) && \text{since rem}\,(70,\ 49) = 21\\
&= \gcd(21, 7) && \text{since rem}\,(49,\ 21) = 7\\
&= \gcd(7, 0) && \text{since rem}\,(21,\ 7) = 0\\
&= 7.
\end{aligned}$$

The Pulverizer goes through the same steps, but requires some extra bookkeeping along the way: as we compute $\gcd(a, b)$, we keep track of how to write each of the remainders (49, 21, and 7, in the example) as a linear combination of a and b. This is worthwhile, because our objective is to write the last nonzero remainder, which is the GCD, as such a linear combination. For our example, here is this extra bookkeeping:

x	y	$(\text{rem}\,(x,\ y))$	$=$	$x - q \cdot y$
259	70	49	$=$	$259 - 3 \cdot 70$
70	49	21	$=$	$70 - 1 \cdot 49$
			$=$	$70 - 1 \cdot (259 - 3 \cdot 70)$
			$=$	$-1 \cdot 259 + 4 \cdot 70$
49	21	7	$=$	$49 - 2 \cdot 21$
			$=$	$(259 - 3 \cdot 70) - 2 \cdot (-1 \cdot 259 + 4 \cdot 70)$
			$=$	$\boxed{3 \cdot 259 - 11 \cdot 70}$
21	7	0		

We began by initializing two variables, $x = a$ and $y = b$. In the first two columns above, we carried out Euclid's algorithm. At each step, we computed $\text{rem}\,(x,\ y)$ which equals $x - \text{qcnt}(x, y) \cdot y$. Then, in this linear combination of x and y, we replaced x and y by equivalent linear combinations of a and b, which we already had computed. After simplifying, we were left with a linear combination of a and b equal to $\text{rem}(x, y)$, as desired. The final solution is boxed.

This should make it pretty clear how and why the Pulverizer works. If you have doubts, it may help to work through Problem 8.9, where the Pulverizer is formalized as a state machine and then verified using an invariant that is an extension of the one used for Euclid's algorithm.

Since the Pulverizer requires only a little more computation than Euclid's algorithm, you can "pulverize" very large numbers very quickly by using this algorithm. As we will soon see, its speed makes the Pulverizer a very useful tool in the field of cryptography.

Now we can restate the Water Jugs Lemma 8.1.5 in terms of the greatest common divisor:

Corollary 8.2.4. *Suppose that we have water jugs with capacities a and b. Then the amount of water in each jug is always a multiple of* $\gcd(a, b)$.

For example, there is no way to form 4 gallons using 3- and 6-gallon jugs, because 4 is not a multiple of $\gcd(3, 6) = 3$.

8.2.3 One Solution for All Water Jug Problems

Corollary 8.2.3 says that 3 can be written as a linear combination of 21 and 26, since 3 is a multiple of $\gcd(21, 26) = 1$. So the Pulverizer will give us integers s and t such that

$$3 = s \cdot 21 + t \cdot 26 \tag{8.5}$$

Now the coefficient s could be either positive or negative. However, we can readily transform this linear combination into an equivalent linear combination

$$3 = s' \cdot 21 + t' \cdot 26 \tag{8.6}$$

where the coefficient s' is positive. The trick is to notice that if in equation (8.5) we increase s by 26 and decrease t by 21, then the value of the expression $s \cdot 21 + t \cdot 26$ is unchanged overall. Thus, by repeatedly increasing the value of s (by 26 at a time) and decreasing the value of t (by 21 at a time), we get a linear combination $s' \cdot 21 + t' \cdot 26 = 3$ where the coefficient s' is positive. (Of course t' must then be negative; otherwise, this expression would be much greater than 3.)

Now we can form 3 gallons using jugs with capacities 21 and 26: We simply repeat the following steps s' times:

1. Fill the 21-gallon jug.

2. Pour all the water in the 21-gallon jug into the 26-gallon jug. If at any time the 26-gallon jug becomes full, empty it out, and continue pouring the 21-gallon jug into the 26-gallon jug.

At the end of this process, we must have emptied the 26-gallon jug exactly $-t'$ times. Here's why: we've taken $s' \cdot 21$ gallons of water from the fountain, and we've poured out some multiple of 26 gallons. If we emptied fewer than $-t'$ times, then by (8.6), the big jug would be left with at least $3 + 26$ gallons, which is more than it can hold; if we emptied it more times, the big jug would be left containing at most $3 - 26$ gallons, which is nonsense. But once we have emptied the 26-gallon jug exactly $-t'$ times, equation (8.6) implies that there are exactly 3 gallons left.

Remarkably, we don't even need to know the coefficients s' and t' in order to use this strategy! Instead of repeating the outer loop s' times, we could just repeat *until we obtain 3 gallons*, since that must happen eventually. Of course, we have to keep track of the amounts in the two jugs so we know when we're done. Here's the solution using this approach starting with empty jugs, that is, at $(0, 0)$:

$$
\begin{array}{ccccccccc}
\xrightarrow{\text{fill 21}} & (21,0) & \xrightarrow{\text{pour 21 into 26}} & & & & & & (0,21) \\
\xrightarrow{\text{fill 21}} & (21,21) & \xrightarrow{\text{pour 21 to 26}} & (16,26) & \xrightarrow{\text{empty 26}} & (16,0) & \xrightarrow{\text{pour 21 to 26}} & & (0,16) \\
\xrightarrow{\text{fill 21}} & (21,16) & \xrightarrow{\text{pour 21 to 26}} & (11,26) & \xrightarrow{\text{empty 26}} & (11,0) & \xrightarrow{\text{pour 21 to 26}} & & (0,11) \\
\xrightarrow{\text{fill 21}} & (21,11) & \xrightarrow{\text{pour 21 to 26}} & (6,26) & \xrightarrow{\text{empty 26}} & (6,0) & \xrightarrow{\text{pour 21 to 26}} & & (0,6) \\
\xrightarrow{\text{fill 21}} & (21,6) & \xrightarrow{\text{pour 21 to 26}} & (1,26) & \xrightarrow{\text{empty 26}} & (1,0) & \xrightarrow{\text{pour 21 to 26}} & & (0,1) \\
\xrightarrow{\text{fill 21}} & (21,1) & \xrightarrow{\text{pour 21 to 26}} & & & & & & (0,22) \\
\xrightarrow{\text{fill 21}} & (21,22) & \xrightarrow{\text{pour 21 to 26}} & (17,26) & \xrightarrow{\text{empty 26}} & (17,0) & \xrightarrow{\text{pour 21 to 26}} & & (0,17) \\
\xrightarrow{\text{fill 21}} & (21,17) & \xrightarrow{\text{pour 21 to 26}} & (12,26) & \xrightarrow{\text{empty 26}} & (12,0) & \xrightarrow{\text{pour 21 to 26}} & & (0,12) \\
\xrightarrow{\text{fill 21}} & (21,12) & \xrightarrow{\text{pour 21 to 26}} & (7,26) & \xrightarrow{\text{empty 26}} & (7,0) & \xrightarrow{\text{pour 21 to 26}} & & (0,7) \\
\xrightarrow{\text{fill 21}} & (21,7) & \xrightarrow{\text{pour 21 to 26}} & (2,26) & \xrightarrow{\text{empty 26}} & (2,0) & \xrightarrow{\text{pour 21 to 26}} & & (0,2) \\
\xrightarrow{\text{fill 21}} & (21,2) & \xrightarrow{\text{pour 21 to 26}} & & & & & & (0,23) \\
\xrightarrow{\text{fill 21}} & (21,23) & \xrightarrow{\text{pour 21 to 26}} & (18,26) & \xrightarrow{\text{empty 26}} & (18,0) & \xrightarrow{\text{pour 21 to 26}} & & (0,18) \\
\xrightarrow{\text{fill 21}} & (21,18) & \xrightarrow{\text{pour 21 to 26}} & (13,26) & \xrightarrow{\text{empty 26}} & (13,0) & \xrightarrow{\text{pour 21 to 26}} & & (0,13) \\
\xrightarrow{\text{fill 21}} & (21,13) & \xrightarrow{\text{pour 21 to 26}} & (8,26) & \xrightarrow{\text{empty 26}} & (8,0) & \xrightarrow{\text{pour 21 to 26}} & & (0,8) \\
\xrightarrow{\text{fill 21}} & (21,8) & \xrightarrow{\text{pour 21 to 26}} & (3,26) & \xrightarrow{\text{empty 26}} & (3,0) & \xrightarrow{\text{pour 21 to 26}} & & (0,3)
\end{array}
$$

The same approach works regardless of the jug capacities and even regardless of the amount we're trying to produce! Simply repeat these two steps until the desired amount of water is obtained:

1. Fill the smaller jug.
2. Pour all the water in the smaller jug into the larger jug. If at any time the larger jug becomes full, empty it out, and continue pouring the smaller jug into the larger jug.

By the same reasoning as before, this method eventually generates every multiple —up to the size of the larger jug —of the greatest common divisor of the jug capacities, namely, all the quantities we can possibly produce. No ingenuity is needed at all!

So now we have the complete water jug story:

Theorem 8.2.5. *Suppose that we have water jugs with capacities a and b. For any $c \in [0, a]$, it is possible to get c gallons in the size a jug iff c is a multiple of* $\gcd(a, b)$.

8.3 Prime Mysteries

Some of the greatest mysteries and insights in number theory concern properties of prime numbers:

Definition 8.3.1. A *prime* is a number greater than 1 that is divisible only by itself and 1. A number other than 0, 1, and -1 that is not a prime is called *composite*.[5]

Here are three famous mysteries:

Twin Prime Conjecture There are infinitely many primes p such that $p + 2$ is also a prime.

In 1966 Chen showed that there are infinitely many primes p such that $p + 2$ is the product of at most two primes. So the conjecture is known to be *almost* true!

Conjectured Inefficiency of Factoring Given the product of two large primes $n = pq$, there is no efficient procedure to recover the primes p and q. That is, no *polynomial time* procedure (see Section 3.5) guaranteed to find p and q in a number of steps bounded by a polynomial $\log n$, which is the number of bits in the binary representation of n.

[5]So 0, 1, and -1 are the only integers that are neither prime nor composite.

The best algorithm known is the "number field sieve," which runs in time proportional to:

$$e^{1.9(\ln n)^{1/3}(\ln\ln n)^{2/3}}.$$

This number grows more rapidly than any polynomial in $\log n$ and is infeasible when n has 300 digits or more.

Efficient factoring is a mystery of particular importance in computer science, as we'll explain later in this chapter.

Goldbach Conjecture We've already mentioned Goldbach's Conjecture 1.1.8 several times: every even integer greater than two is equal to the sum of two primes. For example, $4 = 2 + 2$, $6 = 3 + 3$, $8 = 3 + 5$, etc.

In 1939 Schnirelman proved that every even number can be written as the sum of not more than 300,000 primes, which was a start. Today, we know that every even number is the sum of at most 6 primes.

Primes show up erratically in the sequence of integers. In fact, their distribution seems almost random:

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, \ldots.$$

One of the great insights about primes is that their density among the integers has a precise limit. Namely, let $\pi(n)$ denote the number of primes up to n:

Definition 8.3.2.

$$\pi(n) ::= |\{p \in [2, n] \mid p \text{ is prime}\}|.$$

For example, $\pi(1) = 0$, $\pi(2) = 1$, and $\pi(10) = 4$ because 2, 3, 5, and 7 are the primes less than or equal to 10. Step by step, π grows erratically according to the erratic spacing between successive primes, but its overall growth rate is known to smooth out to be the same as the growth of the function $n/\ln n$:

Theorem 8.3.3 (*Prime Number Theorem*)**.**

$$\lim_{n\to\infty} \frac{\pi(n)}{n/\ln n} = 1.$$

Thus, primes gradually taper off. As a rule of thumb, about 1 integer out of every $\ln n$ in the vicinity of n is a prime.

The Prime Number Theorem was conjectured by Legendre in 1798 and proved a century later by de la Vallee Poussin and Hadamard in 1896. However, after his death, a notebook of Gauss was found to contain the same conjecture, which he apparently made in 1791 at age 15. (You ~~sort of~~ have to feel sorry for all the otherwise "great" mathematicians who had the misfortune of being contemporaries of Gauss.)

A proof of the Prime Number Theorem is beyond ~~our scope~~, but there is a manageable proof (see Problem 8.14) of a related result that is sufficient for our applications:

Theorem 8.3.4 (Chebyshev's Theorem on Prime Density)**.** *For $n > 1$,*

$$\pi(n) > \frac{n}{3 \ln n}.$$

A Prime for Google

In late 2004 a billboard appeared in various locations around the country:

$$\left\{ \begin{array}{c} \text{first 10-digit prime found} \\ \text{in consecutive digits of } e \end{array} \right\} \textbf{. com}$$

Substituting the correct number for the expression in curly-braces produced the URL for a Google employment page. The idea was that Google was interested in hiring the sort of people that could and would solve such a problem.

How hard is this problem? Would you have to look through thousands or millions or billions of digits of e to find a 10-digit prime? The rule of thumb derived from the Prime Number Theorem says that among 10-digit numbers, about 1 in

$$\ln 10^{10} \approx 23$$

is prime. This suggests that the problem isn't really so hard! Sure enough, the first 10-digit prime in consecutive digits of e appears quite early:

$$\begin{aligned} e = &2.718281828459045235360287471352662497757247093699959574966 \\ &967627724076630353547594571382178525166427\mathbf{7427466391}9320030 \\ &5992181741359662904357290033429526059563073813232862794340\ldots \end{aligned}$$

8.4 The Fundamental Theorem of Arithmetic

There is an important fact about primes that you probably already know: every positive integer number has a *unique* prime factorization. So every positive integer can be built up from primes in *exactly one way*. These quirky prime numbers are the building blocks for the integers.

Since the value of a product of numbers is the same if the numbers appear in a different order, there usually isn't a unique way to express a number as a product of primes. For example, there are three ways to write 12 as a product of primes:

$$12 = 2 \cdot 2 \cdot 3 = 2 \cdot 3 \cdot 2 = 3 \cdot 2 \cdot 2.$$

What's unique about the prime factorization of 12 is that any product of primes equal to 12 will have exactly one 3 and two 2's. This means that if we *sort* the primes by size, then the product really will be unique.

Let's state this more carefully. A sequence of numbers is *weakly decreasing* when each number in the sequence is at least as big as the numbers after it. Note that a sequence of just one~~ number as well as a sequence of no numbers —the empty sequence — is~~ weakly decreasing by this definition.

Theorem 8.4.1. *[Fundamental Theorem of Arithmetic] Every positive integer is a product of a* unique *weakly decreasing sequence of primes.*

For example, 75237393 is the product of the weakly decreasing sequence of primes

$$23, 17, 17, 11, 7, 7, 7, 3,$$

and no other weakly decreasing sequence of primes will give 75237393.[6]

Notice that the theorem would be false if 1 were considered a prime; for example, 15 could be written as $5 \cdot 3$, or $5 \cdot 3 \cdot 1$, or $5 \cdot 3 \cdot 1 \cdot 1$, ...

There is a certain wonder in unique factorization, especially in view of the prime number mysteries we've already mentioned. It's a mistake to take it for granted, even if you've known it since you were in a crib. In fact, unique factorization actually fails for many integer-like sets of numbers, ~~for example,~~ the complex numbers of the form $n + m\sqrt{-5}$ for $m, n \in \mathbb{Z}$ (see Problem 8.16).

The Fundamental Theorem is also called the *Unique Factorization Theorem*, which is a more descriptive, less pretentious, name —but~~ hey,~~ we really want to get your attention to the importance and non-obviousness of unique factorization.

[6]The "product" of just one number is defined to be that number, and the product of no numbers is by convention defined to be 1. So each prime, p, is uniquely the product of the primes in the length-one sequence consisting solely of p, and 1, which remember is not a prime, is even so uniquely the product of the empty sequence.

8.4.1 Proving Unique Factorization

The Fundamental Theorem is not hard to prove, but we'll need a couple of preliminary facts.

Lemma 8.4.2. *If p is a prime and $p \mid ab$, then $p \mid a$ or $p \mid b$.*

~~Now~~ Lemma 8.4.2 follows immediately from Unique Factorization: the primes in the product ab are exactly the primes from a and from b. But proving the lemma this way would be cheating: we're going to need this lemma to prove Unique Factorization, ~~and~~ it would be circular to assume it. Instead, we'll use the properties of gcd's and linear combinations to give an easy, noncircular way to prove Lemma 8.4.2.

Proof. One case is if $\gcd(a, p) = p$. Then the claim holds, because a is a multiple of p.

Otherwise, $\gcd(a, p) \neq p$. In this case $\gcd(a, p)$ must be 1, since 1 and p are the only positive divisors of p. Now $\gcd(a, p)$ is a linear combination of a and p, so we have $1 = sa + tp$ for some s, t. Then $b = s(ab) + (tb)p$, that is, b is a linear combination of ab and p. Since p divides both ab and p, it also divides their linear combination b. ■

A routine induction argument extends this statement to:

Lemma 8.4.3. *Let p be a prime. If $p \mid a_1 a_2 \cdots a_n$, then p divides some a_i.*

Now we're ready to prove the Fundamental Theorem of Arithmetic.

Proof. Theorem 2.3.1 showed, using the Well Ordering Principle, that every positive integer can be expressed as a product of primes. So we just have to prove this expression is unique. We will use Well Ordering to prove this too.

The proof is by contradiction: assume, contrary to the claim, that there exist positive integers that can be written as products of primes in more than one way. By the Well Ordering Principle, there is a smallest integer with this property. Call this integer n, and let

$$\begin{aligned} n &= p_1 \cdot p_2 \cdots p_j, \\ &= q_1 \cdot q_2 \cdots q_k, \end{aligned}$$

where both products are in weakly decreasing order and $p_1 \le q_1$.

If $q_1 = p_1$, then n/q_1 would also be the product of different weakly decreasing sequences of primes, namely,

$$\begin{gathered} p_2 \cdots p_j, \\ q_2 \cdots q_k. \end{gathered}$$

Figure 8.1 Alan Turing

Since $n/q_1 < n$, this can't be true, so we conclude that $p_1 < q_1$.

Since the p_i's are weakly decreasing, all the p_i's are less than q_1. But

$$q_1 \mid n = p_1 \cdot p_2 \cdots p_j,$$

so Lemma 8.4.3 implies that q_1 divides one of the p_i's, which contradicts the fact that q_1 is bigger than all them. ■

8.5 Alan Turing

The man pictured in Figure 8.1 is Alan Turing, the most important figure in the history of computer science. For decades, his fascinating life story was shrouded by government secrecy, societal taboo, and even his own deceptions.

At age 24, Turing wrote a paper entitled *On Computable Numbers, with an Application to the Entscheidungsproblem*. The crux of the paper was an elegant way to model a computer in mathematical terms. This was a breakthrough, because it allowed the tools of mathematics to be brought to bear on questions of computation. For example, with his model in hand, Turing immediately proved that there exist problems that no computer can solve —no matter how ingenious the programmer. Turing's paper is all the more remarkable because he wrote it in 1936, a full decade before any electronic computer actually existed.

The word "Entscheidungsproblem" in the title refers to one of the 28 mathematical problems posed by David Hilbert in 1900 as challenges to mathematicians of the 20th century. Turing knocked that one off in the same paper. And perhaps you've heard of the "Church-Turing thesis"? Same paper. So Turing was obviously a brilliant guy who generated lots of amazing ideas. But this lecture is about one of Turing's less-amazing ideas. It involved codes. It involved number theory. And it was sort of stupid.

Let's look back to the fall of 1937. Nazi Germany was rearming under Adolf Hitler, world-shattering war looked imminent, and —like us —Alan Turing was pondering the usefulness of number theory. He foresaw that preserving military secrets would be vital in the coming conflict and proposed a way *to encrypt communications using number theory*. This is an idea that has ricocheted up to our own time. Today, number theory is the basis for numerous public-key cryptosystems, digital signature schemes, cryptographic hash functions, and electronic payment systems. Furthermore, military funding agencies are among the biggest investors in cryptographic research. Sorry Hardy!

Soon after devising his code, Turing disappeared from public view, and half a century would pass before the world learned the full story of where he'd gone and what he did there. We'll come back to Turing's life in a little while; for now, let's investigate the code Turing left behind. The details are uncertain, since he never formally published the idea, so we'll consider a couple of possibilities.

8.5.1 Turing's Code (Version 1.0)

The first challenge is to translate a text message into an integer so we can perform mathematical operations on it. This step is not intended to make a message harder to read, so the details are not too important. Here is one approach: replace each letter of the message with two digits ($A = 01$, $B = 02$, $C = 03$, etc.) and string all the digits together to form one huge number. For example, the message "victory" could be translated this way:

	v	i	c	t	o	r	y
$\rightarrow$	22	09	03	20	15	18	25

Turing's code requires the message to be a prime number, so we may need to pad the result with some more digits to make a prime. The Prime Number Theorem indicates that padding with relatively few digits will work. In this case, appending the digits 13 gives the number 2209032015182513, which is prime.

Here is how the encryption process works. In the description below, m is the unencoded message (which we want to keep secret), m^* is the encrypted message (which the Nazis may intercept), and k is the key.

Beforehand The sender and receiver agree on a *secret key*, which is a large prime k.

Encryption The sender encrypts the message m by computing:

$$m^* = m \cdot k$$

Decryption The receiver decrypts m^* by computing:

$$\frac{m^*}{k} = m.$$

For example, suppose that the secret key is the prime number $k = 22801763489$ and the message m is "victory." Then the encrypted message is:

$$\begin{aligned} m^* &= m \cdot k \\ &= 2209032015182513 \cdot 22801763489 \\ &= 50369825549820718594667857 \end{aligned}$$

There are a couple of basic questions to ask about Turing's code.

1. How can the sender and receiver ensure that m and k are prime numbers, as required?

   The general problem of determining whether a large number is prime or composite has been studied for centuries, and tests for primes that worked well in practice were known even in Turing's time. In the past few decades, fast, guaranteed primality tests have been found as described in the text box below.

2. Is Turing's code secure?

   The Nazis see only the encrypted message $m^* = m \cdot k$, so recovering the original message m requires factoring m^*. Despite immense efforts, no really efficient factoring algorithm has ever been found. It appears to be a fundamentally difficult problem. So, although a breakthrough someday can't be ruled out, the conjecture that there is no efficient way to factor is widely accepted. In effect, Turing's code puts to practical use his discovery that there are limits to the power of computation. Thus, provided m and k are sufficiently large, the Nazis seem to be out of luck!

This all sounds promising, but there is a major flaw in Turing's code.

Primality Testing

It's easy to see that an integer n is prime iff it is not divisible by any number from 2 to $\lfloor \sqrt{n} \rfloor$ (see Problem 1.7). Of course this naive way to test if n is prime takes more than $\sqrt{n}$ steps, which is exponential in the *size* of n measured by the number of digits in the decimal or binary representation of n. Through the early 1970's, no prime testing procedure was known that would never blow up like this.

In 1974, Volker Strassen invented a simple, fast *probabilistic* primality test. Strassens's test gives the right answer when applied to any prime number, but has some probability of giving a wrong answer on a nonprime number. However, the probability of a wrong answer on any given number is so tiny that relying on the answer is the best bet you'll ever make.

Still, the theoretical possibility of a wrong answer was intellectually bothersome —even if the probability of being wrong was a lot less than the probability of an undetectable computer hardware error leading to a wrong answer. Finally in 2002, in an amazing, breakthrough paper beginning with a quote from Gauss emphasizing the importance and antiquity of primality testing, Manindra Agrawal, Neeraj Kayal, and Nitin Saxena presented a thirteen line description of a polynomial time primality test.

In particular, the Agrawal *et al.* test is guaranteed to give the correct answer about primality of any number n in about $(\log n)^{12}$ steps, that is, a number of steps bounded by a twelfth degree polynomial in the length (in bits) of the input, n. This definitively places primality testing way below the problems of exponential difficulty.

Unfortunately, a running time that grows like a 12th degree polynomial is much too slow for practical purposes, and probabilistic primality tests remain the method used in practice today. It's reasonable to expect that improved nonprobabilistic tests will be discovered, but matching the speed of the known probabilistic tests remains a daunting challenge.

8.5.2 Breaking Turing’s Code (Version 1.0)

Let’s consider what happens when the sender transmits a *second* message using Turing’s code and the same key. This gives the Nazis two encrypted messages to look at:

$$m_1^* = m_1 \cdot k \qquad \text{and} \qquad m_2^* = m_2 \cdot k$$

The greatest common divisor of the two encrypted messages, m_1^* and m_2^*, is the secret key k. And, as we’ve seen, the GCD of two numbers can be computed very efficiently. So after the second message is sent, the Nazis can recover the secret key and read *every* message!

A mathematician as brilliant as Turing is not likely to have overlooked such a glaring problem, and we can guess that he had a slightly different system in mind, one based on *modular* arithmetic.

8.6 Modular Arithmetic

On the first page of his masterpiece on number theory, *Disquisitiones Arithmeticae*, Gauss introduced the notion of “congruence.” Now, Gauss is another guy who managed to cough up a half-decent idea every now and then, so let’s take a look at this one. Gauss said that *a is* congruent *to b modulo n* iff $n \mid (a - b)$. This is written

$$a \equiv b \pmod{n}.$$

For example:

$$29 \equiv 15 \pmod{7} \quad \text{because } 7 \mid (29 - 15).$$

It’s not useful to allow a modulus $n \le 1$, and so we will assume from now on that moduli are greater than 1.

There is a close connection between congruences and remainders:

Lemma 8.6.1 (Remainder)**.**

$$a \equiv b \pmod{n} \quad \textit{iff} \quad \text{rem}(a, n) = \text{rem}(b, n).$$

Proof. By the Division Theorem 8.1.4, there exist unique pairs of integers q_1, r_1 and q_2, r_2 such that:

$$a = q_1 n + r_1$$
$$b = q_2 n + r_2,$$

where $r_1, r_2 \in [0, n)$. Subtracting the second equation from the first gives:

$$a - b = (q_1 - q_2)n + (r_1 - r_2),$$

where $r_1 - r_2$ is in the interval $(-n, n)$. Now $a \equiv b \pmod{n}$ if and only if n divides the left side of this equation. This is true if and only if n divides the right side, which holds if and only if $r_1 - r_2$ is a multiple of n. Given the bounds on $r_1 - r_2$, this happens precisely when $r_1 = r_2$, that is, when $\text{rem}(a, n) = \text{rem}(b, n)$. ■

So we can also see that

$$29 \equiv 15 \pmod{7} \quad \text{because } \text{rem}(29, 7) = 1 = \text{rem}(15, 7).$$

Notice that even though "(mod 7)" appears on the end, the $\equiv$ symbol isn't any more strongly associated with the 15 than with the 29. It would ~~really~~, be clearer to write $29 \equiv_{\text{mod } 7} 15$, for example, but the notation with the modulus at the end is firmly entrenched, and ~~we'll just live with it~~.

The Remainder Lemma 8.6.1 explains why the congruence relation has properties like an equality relation. In particular, the following properties[7] follow immediately:

Lemma 8.6.2.

$$\begin{array}{rrll} & a \equiv a & \pmod{n} & \text{(reflexivity)} \\ a \equiv b \quad \textit{iff} & b \equiv a & \pmod{n} & \text{(symmetry)} \\ (a \equiv b \text{ and } b \equiv c) \quad \textit{implies} & a \equiv c & \pmod{n} & \text{(transitivity)} \end{array}$$

We'll make frequent use of another immediate corollary of the Remainder Lemma 8.6.1:

Corollary 8.6.3.

$$a \equiv \text{rem}(a, n) \pmod{n}$$

Still another way to think about congruence modulo n is that it *defines a partition of the integers into n sets so that congruent numbers are all in the same set*. For example, suppose that we're working modulo 3. Then we can partition the integers into 3 sets as follows:

$$\begin{array}{ccccccccc} \{ & \ldots, & -6, & -3, & 0, & 3, & 6, & 9, & \ldots \} \\ \{ & \ldots, & -5, & -2, & 1, & 4, & 7, & 10, & \ldots \} \\ \{ & \ldots, & -4, & -1, & 2, & 5, & 8, & 11, & \ldots \} \end{array}$$

[7] Binary relations with these properties are called *equivalence relations*, see Section 9.10.

according to whether their remainders on division by 3 are 0, 1, or 2. The upshot is that when arithmetic is done modulo n there are really only n different kinds of numbers to worry about, because there are only n possible remainders. In this sense, modular arithmetic is a simplification of ordinary arithmetic.

The next most useful fact about congruences is that they are *preserved* by addition and multiplication:

Lemma 8.6.4 (Congruence)**.** *If* $a \equiv b \pmod{n}$ *and* $c \equiv d \pmod{n}$*, then*

1. $a + c \equiv b + d \pmod{n}$,
2. $ac \equiv bd \pmod{n}$.

Proof. We have that n divides $(b - a)$ which is equal to $(b + c) - (a + c)$, so

$$a + c \equiv b + c \pmod{n}.$$

Also, n divides $(d - c)$, so by the same reasoning

$$b + c \equiv b + d \pmod{n}.$$

Combining these according to Lemma 8.6.2, we get

$$a + c \equiv b + d \pmod{n}.$$

The proof for multiplication is virtually identical, using the fact that if n divides $(b - a)$, then it obviously divides $(bc - ac)$ as well. ■

8.7 Remainder Arithmetic

The Congruence Lemma 8.6.1 says that two numbers are congruent iff their remainders are equal, so we can understand congruences by working out arithmetic with remainders. And if all we want is the remainder modulo n of a series of additions, multiplications, subtractions applied to some numbers, we can take remainders at every step so that the entire computation only involves number in the range $[0, n)$.

General Principle of Remainder Arithmetic

To find the remainder modulo n of the result of a series of additions and multiplications, applied to some integers

- replace each integer by its remainder modulo n,
- keep each result of an addition or multiplication in the range $[0, n)$ by immediately replacing any result outside that range by its remainder on division by n.

For example, suppose we want to find

$$\text{rem}\left((44427^{3456789} + 15555858^{5555})403^{6666666},\ 36\right). \tag{8.7}$$

This looks really daunting if you think about computing these large powers and then taking remainders. For example, the decimal representation of $44427^{3456789}$ has about 20 million digits, so we certainly don't want to go that route. But remembering that integer exponents specify a series of multiplications, we follow the General Principle and replace the numbers being multiplied by their remainders. Since $\text{rem}(44427,\ 36) = 3$, $\text{rem}(15555858,\ 36) = 6$, and $\text{rem}(403,\ 36) = 7$, we find that (8.7) equals the remainder on division by 36 of

$$(3^{3456789} + 6^{5555})7^{6666666}. \tag{8.8}$$

That's a little better, but $3^{3456789}$ has about a million digits in its decimal representation, so we still don't want to compute that. But let's look at the remainders of the first few powers of 3:

$$\begin{aligned}
\text{rem}(3,\ 36) &= 3\\
\text{rem}\left(3^2,\ 36\right) &= 9\\
\text{rem}\left(3^3,\ 36\right) &= 27\\
\text{rem}\left(3^4,\ 36\right) &= 9.
\end{aligned}$$

We got a repeat of the second step, $\text{rem}\left(3^2,\ 36\right)$ after just two more steps. This means means that starting at 3^2, the sequence of remainders of successive powers of 3 will keep repeating every 2 steps. So a product of an odd number of three of more 3's will have the same remainder modulo 36 as a product of just three 3's. Therefore,

$$\text{rem}\left(3^{3456789},\ 36\right) = \text{rem}\left(3^3,\ 36\right) = 27.$$

What a win!

Powers of 6 are even easier because $\text{rem}\left(6^2, 36\right) = 0$, so 0's keep repeating after the second step. Powers of 7 repeat after six steps, but on the fifth step you get a 1, so (8.8) successively simplifies to be the remainders of the following terms:

$$\begin{aligned}
&(3^{3456789} + 6^{5555})7^{6666666}\\
&(3^3 + 6^2 \cdot 6^{5553})(7^6)^{1111111}\\
&(3^3 + 0 \cdot 6^{5553})1^{1111111}\\
&= 27.
\end{aligned}$$

Notice that **it would be a disastrous blunder to replace an exponent by its remainder**. The General Principle applies to numbers that are *operands* of plus and times, whereas the exponent is a number that controls how many multiplications to perform. Watch out for this blunder.

8.7.1 The ring $\mathbb{Z}_n$

It's time to be more precise about the General Principle and why it works. To begin, let's introduce the notation $+_n$ for doing an addition and then immediately taking a remainder on division by n, as specified by the General Principle; likewise for multiplying:

$$\begin{aligned}
i +_n j &::= \text{rem}\,(i + j,\ n)\,,\\
i \cdot_n j &::= \text{rem}\,(ij,\ n)\,.
\end{aligned}$$

The General Principle is simply the repeated application of the following lemma which provides the formal justification for remainder arithmetic:

Lemma 8.7.1.

$$\text{rem}\,(i + j,\ n) = \text{rem}\,(i,\ n) +_n \text{rem}\,(j,\ n)\,, \tag{8.9}$$

$$\text{rem}\,(ij,\ n) = \text{rem}\,(i,\ n) \cdot_n \text{rem}\,(j,\ n)\,. \tag{8.10}$$

Proof. By Corollary 8.6.3, $i \equiv \text{rem}\,(i,\ n)$ and $j \equiv \text{rem}\,(j,\ n)$, so by the Congruence Lemma 8.6.4

$$i + j \equiv \text{rem}\,(i,\ n) + \text{rem}\,(j,\ n) \pmod{n}.$$

By Corollary 8.6.3 again, the remainders on each side of this congruence are equal, which immediately gives (8.9). An identical proof applies to (8.10). ■

The set of integers in the range $[0, n)$ together with the operations $+_n$ and $\cdot_n$ is referred to as $\mathbb{Z}_n$, the *ring of integers modulo* n. As a consequence of Lemma 8.7.1, the familiar rules of arithmetic hold in $\mathbb{Z}_n$, for example:[8]

$$\begin{aligned}
(i \cdot_n j) \cdot_n k &= i \cdot_n (j \cdot_n k) && \text{(associativity of } \cdot_n\text{)},\\
(i +_n j) +_n k &= i +_n (j +_n k) && \text{(associativity of } +_n\text{)},\\
1 \cdot_n k &= k && \text{(identity for } \cdot_n\text{)},\\
0 +_n k &= k && \text{(identity for } +_n\text{)},\\
k +_n (-k) &= 0 && \text{(inverse for } +_n\text{)},\\
i +_n j &= j +_n i && \text{(commutativity of } +_n\text{)}\\
i \cdot_n (j +_n k) &= (i \cdot_n j) +_n (i \cdot_n k) && \text{(distributivity)},\\
i \cdot_n j &= j \cdot_n i && \text{(commutativity of } \cdot_n\text{)}
\end{aligned}$$

Associativity implies the familiar fact that it's safe to omit the parentheses in products:

$$k_1 \cdot_n k_2 \cdot_n \cdots \cdot_n k_m$$

comes out the same no matter how it is parenthesized.

The overall theme is that remainder arithmetic is a lot like ordinary arithmetic. But there are a couple of exceptions we're about to examine.

8.8 Turing's Code (Version 2.0)

In 1940, France had fallen before Hitler's army, and Britain stood alone against the Nazis in western Europe. British resistance depended on a steady flow of supplies brought across the north Atlantic from the United States by convoys of ships. These convoys were engaged in a cat-and-mouse game with German "U-boats"—submarines—which prowled the Atlantic, trying to sink supply ships and starve Britain into submission. The outcome of this struggle pivoted on a balance of information: could the Germans locate convoys better than the Allies could locate U-boats or vice versa?

Germany lost.

[8] A set with addition and multiplication operations that satisy these equalities is known as a *commutative ring*. In addition to $\mathbb{Z}_n$, the integers, reals, and polynomials with integer coefficients, are all examples of commutative rings. On the other hand, the set $\{\mathbf{T}, \mathbf{F}\}$ of truth values with OR for addition and AND for multiplication is *not* a ring; it satisfies most, but not all, of these equalities.

But a critical reason behind Germany's loss was made public only in 1974: Germany's naval code, *Enigma*, had been broken by the Polish Cipher Bureau[9] and the secret had been turned over to the British a few weeks before the Nazi invasion of Poland in 1939. Throughout much of the war, the Allies were able to route convoys around German submarines by listening in to German communications. The British government didn't explain *how* Enigma was broken until 1996. When ~~it~~ was finally released (by the US), ~~the story~~ revealed that Alan Turing had joined the secret British codebreaking effort at Bletchley Park in 1939, where he became the lead developer of methods for rapid, bulk decryption of German Enigma messages. Turing's Enigma deciphering was an invaluable contribution to the Allied victory over Hitler.

Governments are always tight-lipped about cryptography, but the half-century of official silence about Turing's role in breaking Enigma and saving Britain may be related to some disturbing events after the war~~.~~ More on that later. Let's get back to number theory and consider an alternative interpretation of Turing's code. Perhaps we had the basic idea right (multiply the message by the key), but erred in using *conventional* arithmetic instead of *modular* arithmetic. Maybe this is what Turing meant:

Beforehand The sender and receiver agree on a large number n, which may be made public. (This will be the modulus for all our arithmetic.) As in Version 1.0, they also agree that some prime number $k < n$ will be the secret key.

Encryption As in Version 1.0, the message m should be another prime in $[0, n)$. The sender encrypts the message m to produce m^* by computing mk, but this time in $\mathbb{Z}_n$:

$$m^* ::= m \cdot_n k \tag{8.11}$$

Decryption (Uh-oh.)

The decryption step is a problem. We might hope to decrypt in the same way as before by dividing the encrypted message m^* by the key k. The difficulty is that m^* is the *remainder* when mk is divided by n. So dividing m^* by k might not even give us an integer!

This decoding difficulty can be overcome with a better understanding of when it is ok to divide by k in modular arithmetic.

[9]See http://en.wikipedia.org/wiki/Polish_Cipher_Bureau.

8.9 Multiplicative Inverses and Cancelling

The *multiplicative inverse* of a number x is another number x^{-1} such that

$$x^{-1} \cdot x = 1.$$

From now on, when we say "inverse," we mean multiplicative inverse.

For example, over the rational numbers, $1/3$ is, of course, an inverse of 3, since,

$$\frac{1}{3} \cdot 3 = 1.$$

In fact, with the sole exception of 0, every rational number n/m has an inverse, namely, m/n. On the other hand, over the integers, only 1 and -1 have inverses. Over the ring $\mathbb{Z}_n$, things get a little more complicated. For example, in $\mathbb{Z}_{15}$, 2 is a multiplicative inverse of 8, since

$$2 \cdot_{15} 8 = 1.$$

On the other hand, 3 does not have a multiplicative inverse in $\mathbb{Z}_{15}$. We can prove this by contradiction: suppose there was an inverse j for 3, that is

$$1 = 3 \cdot_{15} j$$

Then multiplying both sides of this equality by 5 —in the ring $\mathbb{Z}_{15}$ —leads directly to the contradiction $5 = 0$:

$$\begin{aligned} 5 &= 5 \cdot_{15} (3 \cdot_{15} j) \\ &= (5 \cdot_{15} 3) \cdot_{15} j \\ &= 0 \cdot_{15} j = 0, \end{aligned}$$

So there can't be any such inverse j.

So some numbers have inverses modulo 15 and others don't. This may seem a little unsettling at first, but there's a simple explanation of what's going on.

8.9.1 Relative Primality

Integers that have no prime factor in common are called *relatively prime*.[10] This is the same as having no common divisor (prime or not) greater than 1. It is also equivalent to saying $\gcd(a, b) = 1$.

For example, 8 and 15 are relatively prime, since $\gcd(8, 15) = 1$. On the other hand, 3 and 15 are not relatively prime, since $\gcd(3, 15) = 3 \neq 1$. This turns out to explain why 8 has an inverse over $\mathbb{Z}_{15}$ and 3 does not.

[10]Other texts call them *coprime*.

Lemma 8.9.1. *If k is relatively prime to n, then k has an inverse in $\mathbb{Z}_n$.*

Proof. If k is relatively prime to n, then $\gcd(n, k) = 1$ by definition of gcd. So we can use the Pulverizer from section 8.2.2 to find a linear combination of n and k equal to 1:

$$sn + tk = 1.$$

So taking remainders of division by n of both sides of this equality, and then applying Lemma 8.7.1, we get

$$(\text{rem}(s, n) \cdot_n \text{rem}(n, n)) +_n (\text{rem}(t, n) \cdot_n k) = 1.$$

But $\text{rem}(n, n) = 0$, so

$$\text{rem}(t, n) \cdot_n k = 1$$

Thus, $\text{rem}(t, n)$ is a multiplicative inverse of k. ■

By the way, it's nice to know that when they exist, inverses are unique. That is,

Lemma 8.9.2. *If i and j are both inverses of k in $\mathbb{Z}_n$, then $i = j$.*

Proof.

$$i = i \cdot_n 1 = i \cdot_n (k \cdot_n j) = (i \cdot_n k) \cdot_n j = 1 \cdot_n j = j.$$

■

So the proof of Lemma 8.9.1 shows that the unique inverse in $\mathbb{Z}_n$ for any k relatively prime to n can be found simply by taking the remainder of the coefficient of k in a linear combination of k and n that equals 1.

Notice that working with a prime modulus, p, is attractive because, like the rational and real numbers, in $\mathbb{Z}_p$ every nonzero number has an inverse. But arithmetic modulo a composite is really only a little more painful than working modulo a prime —though you may think this is like the doctor saying, “This is only going to hurt a little,” before he jams a big needle in your arm.

8.9.2 Cancellation

Another sense in which real numbers are nice is that it's ok to cancel common factors. In other words, if we know that $rt = st$ for real numbers r, s, t, then as long as $t \neq 0$, we can cancel the t's and conclude that $r = s$. In general, cancellation is *not* valid in $\mathbb{Z}_n$. For example,

$$4 \cdot_{15} 10 = 1 \cdot_{15} 10 \pmod{15},$$

but cancelling the 10's leads to the absurd conclusion that 4 equals 1.

The fact that multiplicative terms cannot be canceled is the most significant way in which congruences differ from ordinary integer equations.

Definition 8.9.3. A number k is *cancellable* modulo in $\mathbb{Z}_n$ iff

$$a \cdot_n k = b \cdot_n k \quad \text{implies} \quad a = b$$

for all $a, b \in [0, n)$.

If a number is relatively prime to 15, it can be cancelled by multiplying by its inverse. So cancelling obviously works for numbers that have inverses:

Lemma 8.9.4. *If k has an inverse modulo n, then k is cancellable modulo n.*

But 10 is not relatively prime to 15, and that's why it is not cancellable. More generally, if k is not relatively prime to n, then it's easy to see that it isn't cancellable in $\mathbb{Z}_n$. Namely, suppose $\gcd(k, n) = m > 1$. So k/m and n/m are positive integers, and we have

$$\begin{aligned} (n/m) \cdot k &= n \cdot (k/m), \\ \text{rem}\left((n/m) \cdot k,\ n\right) &= \text{rem}\left(n \cdot (k/m),\ n\right), \\ (n/m) \cdot_n k &= 0 = 0 \cdot_n k. \end{aligned}$$

Now k can't be cancelled or we would reach the false conclusion that $n/m = 0$.

To summarize, we have

Theorem 8.9.5. *The following are equivalent for $k \in [0, n)$:*

$$\begin{aligned} &\gcd(k, n) = 1, \\ &k \textit{ has an inverse in } \mathbb{Z}_n, \\ &k \textit{ is cancellable in } \mathbb{Z}_n. \end{aligned}$$

8.9.3 Decrypting (Version 2.0)

Multiplicative inverses are the key to decryption in Turing's code. Specifically, we can recover the original message by multiplying the encoded message by the $\mathbb{Z}_n$-inverse, j, of the key:

$$m^* \cdot_n j = (m \cdot_n k) \cdot_n j = m \cdot_n (k \cdot_n j) = m \cdot_n 1 = m.$$

So all we need to decrypt the message is to find an inverse of the secret key k, which will be easy using the Pulverizer —providing k has an inverse. But k is positive and less than the modulus n, so one simple way to ensure that k is relatively prime to the modulus is to have n be a prime number.

8.9.4 Breaking Turing’s Code (Version 2.0)

The Germans didn’t bother to encrypt their weather reports with the highly-secure Enigma system. After all, so what if the Allies learned that there was rain off the south coast of Iceland? But~~,~~ amazingly, this practice provided the British with a critical edge in the Atlantic naval battle during 1941.

The problem was that some of those weather reports had originally been transmitted using Enigma from U-boats out in the Atlantic. Thus, the British obtained both unencrypted reports and the same reports encrypted with Enigma. By comparing the two, the British were able to determine which key the Germans were using that day and could read all other Enigma-encoded traffic. Today, this would be called a *known-plaintext attack*.

Let’s see how a known-plaintext attack would work against Turing’s code. Suppose that the Nazis know both the plain text, m, and its encrypted form, m^*. Now in Version 2.0,

$$m^* = m \cdot_n k$$

and since m is positive and less than the prime n, the Nazis can use the Pulverizer to find the $\mathbb{Z}_n$-inverse, j, of m. Now

$$j \cdot_n m^* = j \cdot_n (m \cdot_n k) = (j \cdot_n m) \cdot_n k = 1 \cdot_n k = k.$$

So by computing $j \cdot_n m^* = k$, the Nazis get the secret key and can then decrypt any message!

This is a huge vulnerability, so Turing’s hypothetical Version 2.0 code has no practical value. Fortunately, Turing got better at cryptography after devising this code; his subsequent deciphering of Enigma messages surely saved thousands of lives, if not the whole of Britain.

8.9.5 Turing Postscript

A few years after the war, Turing’s home was robbed. Detectives soon determined that a former homosexual lover of Turing’s had conspired in the robbery. So they arrested him —that is, they arrested Alan Turing —because homosexuality was a British crime punishable by up to two years in prison at that time. Turing was sentenced to a hormonal “treatment” for his homosexuality: he was given estrogen injections. He began to develop breasts.

Three years later, Alan Turing, the founder of computer science, was dead. His mother explained what happened in a biography of her own son. Despite her repeated warnings, Turing carried out chemistry experiments in his own home. Apparently, her worst fear was realized: by working with potassium cyanide while eating an apple, he poisoned himself.

However, Turing remained a puzzle to the very end. His mother was a devout woman who considered suicide a sin. And, other biographers have pointed out, Turing had previously discussed committing suicide by eating a poisoned apple. Evidently, Alan Turing, who founded computer science and saved his country, took his own life in the end, and in just such a way that his mother could believe it was an accident.

Turing's last project before he disappeared from public view in 1939 involved the construction of an elaborate mechanical device to test a mathematical conjecture called the Riemann Hypothesis. This conjecture first appeared in a sketchy paper by Bernhard Riemann in 1859 and is now one of the most famous unsolved problems in mathematics.

8.10 Euler's Theorem

The RSA cryptosystem examined in the next section, and other current schemes for encoding secret messages, involve computing remainders of numbers raised to large powers. A basic fact about remainders of powers follows from a theorem due to Euler about congruences.

Definition 8.10.1. For $n > 0$, define[11]

$$\phi(n) ::= \text{the number of integers in } [0, n), \text{ that are relatively prime to } n.$$

This function ϕ is known as Euler's ϕ function.[12]

For example, $\phi(7) = 6$ because all 6 positive numbers in $[0, 7)$ are relatively prime to the prime number 7. Only 0 is not relatively prime to 7. Also, $\phi(12) = 4$ since 1, 5, 7, and 11 are the only numbers in $[0, 12)$ that are relatively prime to 12.

More generally, if p is prime, then $\phi(p) = p - 1$ since every positive number in $[0, p)$ is relatively prime to p. When n is composite, however, the ϕ function gets a little complicated. We'll get back to it in the next section.

Theorem 8.10.2 (Euler's Theorem)**.** *If n and k are relatively prime, then*

$$k^{\phi(n)} \equiv 1 \pmod{n}. \tag{8.12}$$

[11] Since 0 is not relatively prime to anything, $\phi(n)$ could equivalently be defined using the interval $[1, n)$ instead of $[0, n)$.

[12] Some texts call it Euler's *totient function*.

The Riemann Hypothesis

The formula for the sum of ~~an~~ infinite geometric series says:

$$1 + x + x^2 + x^3 + \cdots = \frac{1}{1-x}$$

Substituting $x = \frac{1}{2^s}$, $x = \frac{1}{3^s}$, $x = \frac{1}{5^s}$, and so on for each prime number gives a sequence of equations:

$$\begin{aligned}
1 + \frac{1}{2^s} + \frac{1}{2^{2s}} + \frac{1}{2^{3s}} + \cdots &= \frac{1}{1-1/2^s}\\
1 + \frac{1}{3^s} + \frac{1}{3^{2s}} + \frac{1}{3^{3s}} + \cdots &= \frac{1}{1-1/3^s}\\
1 + \frac{1}{5^s} + \frac{1}{5^{2s}} + \frac{1}{5^{3s}} + \cdots &= \frac{1}{1-1/5^s}\\
&\text{etc.}
\end{aligned}$$

Multiplying together all the left sides and all the right sides gives:

$$\sum_{n=1}^{\infty} \frac{1}{n^s} = \prod_{p \in \text{primes}} \left(\frac{1}{1-1/p^s} \right)$$

The sum on the left is obtained by multiplying out all the infinite series and applying the Fundamental Theorem of Arithmetic. For example, the term $1/300^s$ in the sum is obtained by multiplying $1/2^{2s}$ from the first equation by $1/3^s$ in the second and $1/5^{2s}$ in the third. Riemann noted that every prime appears in the expression on the right. So he proposed to learn about the primes by studying the equivalent, but simpler expression on the left. In particular, he regarded s as a complex number and the left side as a function, $\zeta(s)$. Riemann found that the distribution of primes is related to values of s for which $\zeta(s) = 0$, which led to his famous conjecture:

Definition 8.9.6. The *Riemann Hypothesis*: Every nontrivial zero of the zeta function $\zeta(s)$ lies on the line $s = 1/2 + ci$ in the complex plane.

A proof would immediately imply, among other things, a strong form of the Prime Number Theorem.
Researchers continue to work intensely to settle this conjecture, as they have for over a century. It is another of the Millennium Problems whose solver will earn \$1,000,000 from the Clay Institute.

Rephrased in terms of the ring $\mathbb{Z}_n$, (8.12) is equivalent to

$$k^{\phi(n)} = 1 \qquad (\mathbb{Z}_n) \tag{8.13}$$

Here, and in the rest of this section, the arithmetic is done in $\mathbb{Z}_n$. In particular, $k^{\phi(n)}$ is the $\cdot_n$-product of k with itself $\phi(n)$ times.

Equation (8.13) will follow from a series of easy lemmas.

Definition 8.10.3. Let $\text{gcd1}\{n\}$ be the integers in $[0, n)$, that are relatively prime to n:[13]

$$\text{gcd1}\{n\} ::= \{k \in [0, n) \mid \gcd(k, n) = 1\}. \tag{8.14}$$

Consequently,

$$\phi(n) = |\,\text{gcd1}\{n\}|.$$

We know every element in $\text{gcd1}\{n\}$ has a $\mathbb{Z}_n$-inverse (Theorem 8.9.5) and therefore is cancellable. Also $\text{gcd1}\{n\}$ is closed under multiplication in $\mathbb{Z}_n$:

Lemma 8.10.4. *If $j, k \in \text{gcd1}\{n\}$, then $j \cdot k \in \text{gcd1}\{n\}$.*

There are lots of easy ways to prove this (see Problem 8.45).

Definition 8.10.5. Define the *order* of $k \in [0, n)$ over $\mathbb{Z}_n$ to be

$$\text{ord}(k, n) ::= \min\{m \geq 0 \mid k^m = 1\}.$$

If no power of k equals 1 in $\mathbb{Z}_n$, then $\text{ord}(k, n) ::= \infty$.

Lemma 8.10.6. *Every element of $\text{gcd1}\{n\}$ has finite order.*

Proof. Suppose $k \in \text{gcd1}\{n\}$. We need to show is that some power of k over $\mathbb{Z}_n$ equals 1.

But since $\text{gcd1}\{n\}$ has fewer than n elements, some number must occur twice in the list

$$k^1,\ k^2,\ \ldots,\ k^n.$$

That is,

$$k^{i+m} = k^i \tag{8.15}$$

for some $m > 0$ and $i \in [0, n)$. But k is cancellable over $\mathbb{Z}_n$, so we can cancel the first i of the k's on both sides of (8.15) to get

$$k^m = 1.$$

■

[13] Other texts use the notation n^* for $\text{gcd1}\{n\}$.

Now let's work out an example that illustrates the remaining ideas needed to prove Euler's Theorem. Suppose $n = 28$, so

$$\begin{aligned} \text{gcd1}\{28\} &= \{1, 3, 5, 9, 11, 13, 15, 17, 19, 23, 25, 27\}, \text{ and} \\ \phi(28) &= |\,\text{gcd1}\{28\}| = 12. \end{aligned} \tag{8.16}$$

We pick any element of gcd1{28}, for example, 9. Let P_9 be all the positive powers of 9 in $\mathbb{Z}_{28}$, so

$$P_9 ::= \{9, 9^2, \dots, 9^k, \dots\}.$$

The order of 9 in $\mathbb{Z}_{28}$ turns out to be 3, since $9^2 = 25$ and $9^3 = 1$. So P_9 really has just these 3 elements:

$$P_9 = \{9, 25, 1\}.$$

Definition 8.10.7. For any $m \in [0, n)$ and subset $P \subseteq [0, n)$, define

$$mP ::= \{m \cdot p \mid p \in P\}.$$

Let's look at $3P_9$. Multiplying each of the elements in P_9 by 3 gives

$$3P_9 = \{27, 19, 3\}.$$

The first thing to notice is that $3P_9$ also has 3 elements. We could have predicted this: different elements of P_9 must map to different elements of $3P_9$ since $3 \in \text{gcd1}\{28\}$ is cancellable.

Lemma 8.10.8. *For any set, $P \subseteq [0, n)$, if $k \in \text{gcd1}\{n\}$, then*

$$|P| = |kP|. \tag{8.17}$$

Proof. Define a function $f_k : P \to kP$ by the rule

$$f_k(p) ::= k \cdot p.$$

The function f_k is total and surjective by definition. It is also an injection because

$$f_k(p_1) = f_k(p_2)$$

means

$$k \cdot p_1 = k \cdot p_2,$$

which implies that $p_1 = p_2$ since $k \in \text{gcd1}\{n\}$ is cancellable. This shows that f_k is a bijection, and (8.17) follows by the Mapping Rule 4.5. ■

Continuing with the example, the next number in the list (8.16) of elements of gcd1{28} is 5, so let's look at

$$5P_9 = \{17, 13, 5\}.$$

Again $5P_9$ has 3 elements since $5 \in \text{gcd1}\{28\}$, but now notice something else: $5P_9$ has no elements in common with $3P_9$, and neither $3P_9$ nor $5P_9$ have any elements in common with P_9. The following lemma explains this.

Lemma 8.10.9. *Let $P_k ::= \{k, k^2, \dots, k^i, \dots\}$ be the set of powers of some element $k \in \text{gcd1}\{n\}$, and suppose $a, b \in [0, n)$. If the sets aP_k and bP_k have an element in common, then $aP_k = bP_k$.*

Proof. So suppose aP_k and bP_k have an element in common. That is,

$$ak^i = bk^j$$

for some $i, j \geq 0$. Then multiplying both sides of this equality by an arbitrary power of k, we conclude that a times any large power of k equals b times another large power of k, and conversely, b times any large power of k equals a times a large power of k. But since $k \in \text{gcd1}\{n\}$ has finite order, every element in P_k can be expressed as a large power of k, and we conclude that $aP_k = bP_k$. ■

Notice that since $P_9 = 1P_9$, Lemma 8.10.9 explains not only why $3P_9$ and $5P_9$ don't overlap, but also why neither of them overlaps with P_9.

The next number in the list of elements of gcd1{28} is 9, which brings us to

$$9P_9 = \{25, 1, 9\} = P.$$

Of course we could have predicted that $9P = P$ without actually multiplying each element of P_9 by 9; since $1 \in P_9$, we know that $9 = 9 \cdot 1 \in 9P_9$, so $9P_9$ and $1P_9$ have the element 9 in common, and therefore must be equal according to Lemma 8.10.9.

Next, we come to

$$11P_9 = \{15, 23, 11\}.$$

Now we're done, because we have 4 different size 3 subsets of gcd1{28}, and since gcd1{28} has 12 elements, we must have them all. That is,

$$\text{gcd1}\{28\} = 1P_9 \cup 3P_9 \cup 5P_9 \cup 11P_9.$$

This means there's no need to examine mP_9 for any of the remaining numbers $m \in \text{gcd1}\{28\}$ since they are bound to overlap with, and therefore be equal to, one of the four sets $1P_9, 3P_9, 5P_9$, and $11P_9$, that we already have. For example, we could conclude without further calculation that the next set, $13P_9$, must be the same as $5P_9$, since both include the number 13.

We can also see why the size of P_9 had to divide $\phi(28)$ —because $\text{gcd1}\{28\}$ is a union of non-overlapping sets of the same size as P_9.

Lemma 8.10.10. *If* $k \in \text{gcd1}\{n\}$*, then*

$$\text{ord}(k,n) \mid \phi(n).$$

Proof. Let P_k be the powers of k, so P_k has $\text{ord}(k,n)$ elements, namely,

$$P_k = \{k, k^2, \dots, k^{\text{ord}(k,n)}\}$$

By Lemma 8.10.4, both P_k and mP_k are subsets of $\text{gcd1}\{n\}$ for $m \in \text{gcd1}\{n\}$. Since $1 \in P_k$, we have $m \in mP_k$ for all $m \in [0,n)$. Therefore,

$$\text{gcd1}\{n\} = \bigcup_{m \in \text{gcd1}\{n\}} mP_k.$$

By Lemma 8.10.8, $|mP_k| = \text{ord}(k,n)$, and by Lemma 8.10.9, distinct mP_k's don't overlap, it follows that

$$|\text{gcd1}\{n\}| = \text{ord}(k,n) \cdot |\{mP_k \mid m \in \text{gcd1}\{n\}\}|.$$

So $\text{ord}(k,n)$ divides $|\text{gcd1}\{n\}| = \phi(n)$. ■

In particular, Lemma 8.10.10 implies that $\phi(n) = \text{ord}(k,n) \cdot c$ for some number c, and so

$$k^{\phi(n)} = k^{\text{ord}(k,n) \cdot c} = \left(k^{\text{ord}(k,n)}\right)^c = 1^c = 1. \tag{8.18}$$

Euler's theorem now follows immediately, since it is simply the restatement of the $\mathbb{Z}_n$ equation (8.18) in terms of congruence mod n.

Euler's theorem offers another way to find inverses modulo n: if k is relatively prime to n, then $k^{\phi(n)-1}$ is a $\mathbb{Z}_n$-inverse of k, and we can compute this power of k efficiently using fast exponentiation. However, this approach requires computing $\phi(n)$. In the next section, we'll show that computing $\phi(n)$ is easy *if* we know the prime factorization of n. But we know that finding the factors of n is generally hard to do when n is large, and so the Pulverizer remains the best approach to computing inverses modulo n.

Fermat's Little Theorem

For the record, we mention a famous special case of Euler's Theorem that was known to Fermat a century earlier.

Corollary 8.10.11 (Fermat's Little Theorem)**.** *Suppose p is a prime and k is not a multiple of p. Then:*

$$k^{p-1} \equiv 1 \pmod{p}$$

8.10.1 Computing Euler's ϕ Function

RSA works using arithmetic modulo the product of two large primes, so we begin with an elementary explanation of how to compute $\phi(pq)$ for primes p and q:

Lemma 8.10.12.

$$\phi(pq) = (p-1)(q-1)$$

for primes $p \neq q$.

Proof. Since p and q are prime, any number that is not relatively prime to pq must be a multiple of p or a multiple of q. Among the pq numbers in $[0, pq)$, there are precisely q multiples of p and p multiples of q. Since p and q are relatively prime, the only number in $[0, pq)$ that is a multiple of both p and q is 0. Hence, there are $p + q - 1$ numbers in $[0, pq)$ that are *not* relatively prime to n. This means that

$$\begin{aligned}\phi(pq) &= pq - (p + q - 1)\\ &= (p-1)(q-1),\end{aligned}$$

as claimed.[14] ■

The following theorem provides a way to calculate $\phi(n)$ for arbitrary n.

Theorem 8.10.13.

(a) *If p is a prime, then $\phi(p^k) = p^k - p^{k-1}$ for $k \geq 1$.*

(b) *If a and b are relatively prime, then $\phi(ab) = \phi(a)\phi(b)$.*

Here's an example of using Theorem 8.10.13 to compute $\phi(300)$:

$$\begin{aligned}\phi(300) &= \phi(2^2 \cdot 3 \cdot 5^2)\\ &= \phi(2^2) \cdot \phi(3) \cdot \phi(5^2) && \text{(by Theorem 8.10.13.(b))}\\ &= (2^2 - 2^1)(3^1 - 3^0)(5^2 - 5^1) && \text{(by Theorem 8.10.13.(a))}\\ &= 80.\end{aligned}$$

[14] This proof previews a kind of counting argument that we will explore more fully in Part III.

To prove Theorem 8.10.13.(a), notice that every pth number among the p^k numbers in $[0, p^k)$ is divisible by p, and only these are divisible by p. So $1/p$ of these numbers are divisible by p and the remaining ones are not. That is,

$$\phi(p^k) = p^k - (1/p)p^k = p^k - p^{k-1}.$$

We'll leave a proof of Theorem 8.10.13.(b) to Problem 8.43.

As a consequence of Theorem 8.10.13, we have

Corollary 8.10.14. *For any number n, if $p_1, p_2, \ldots, p_j$ are the (distinct) prime factors of n, then*

$$\phi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\cdots\left(1 - \frac{1}{p_j}\right).$$

We'll give another proof of Corollary 8.10.14 in a few weeks based on rules for counting.

8.11 RSA Public Key Encryption

Turing's code did not work as he hoped. However, his essential idea —using number theory as the basis for cryptography —succeeded spectacularly in the decades after his death.

In 1977, Ronald Rivest, Adi Shamir, and Leonard Adleman at MIT proposed a highly secure cryptosystem (called **RSA**) based on number theory. The purpose of the RSA scheme is to transmit secret messages over public communication channels. As with Turing's codes, the messages transmitted will actually be nonnegative integers of some fixed size.

Moreover, RSA has a major advantage over traditional codes: the sender and receiver of an encrypted message need not meet beforehand to agree on a secret key. Rather, the receiver has both a *private key*, which they guard closely, and a *public key*, which they distribute as widely as possible. A sender wishing to transmit a secret message to the receiver encrypts their message using the receiver's widely-distributed public key. The receiver can then decrypt the received message using their closely-held private key. The use of such a *public key cryptography* system allows you and Amazon, for example, to engage in a secure transaction without meeting up beforehand in a dark alley to exchange a ~~key~~.

Interestingly, RSA does not operate modulo a prime, as Turing's hypothetical Version 2.0 may have, but rather modulo the product of *two* large primes — typically primes that are hundreds of digits long. Also, instead of encrypting by multiplication with a secret key, RSA exponentiates to a secret power —which is why Euler's Theorem is central to understanding RSA.

The scheme for RSA public key encryption appears in the box.

If the message m is relatively prime to n, Euler's Theorem immediately implies that this way of decoding the encrypted message indeed reproduces the original unencrypted message. In fact, the decoding always works —even in (the highly unlikely) case that m is not relatively prime to n. The details are worked out in Problem 8.57.

Why is RSA thought to be secure? It would be easy to figure out the private key d if you knew p and q —you could do it the same way the **Receiver** does using the Pulverizer. But assuming the conjecture that it is hopelessly hard to factor a number that is the product of two primes with hundreds of digits, an effort to factor n is not going to break RSA.

Could there be another approach to reverse engineer the private key d from the public key that did not involve factoring n? Not really. It turns out that given just the private and the public keys, it is easy to factor n (a proof of this is sketched in Problem 8.59). So if we are confident that factoring is hopelessly hard, then we can be equally confident that finding the private key just from the public key will be hopeless.

But even if we are confident that an RSA private key won't be found, this doesn't rule out the possibility of decoding RSA messages in a way that sidesteps the private key. It is an important unproven conjecture in cryptography that *any* way of cracking RSA —not just by finding the secret key —would imply the ability to factor. This would be a much stronger theoretical assurance of RSA security than is presently known.

But the real reason for confidence is that RSA has withstood all attacks by the world's most sophisticated cryptographers for over 30 years. Despite decades of these attacks, no significant weakness has been found. That's why the mathematical, financial, and intelligence communities are betting the family jewels on the security of RSA encryption.

You can hope that with more studying of number theory, you will be the first to figure out how to do factoring quickly and, among other things, break RSA. But be further warned that even Gauss worked on factoring for years without a lot to show for his efforts —and if you do figure it out, you might wind up meeting some humorless fellows working for a Federal agency. . . .

The RSA Cryptosystem

A **Receiver** who wants to be able to receive secret numerical messages creates a *private key*, which they keep secret, and a *public key* which they make publicly available. Anyone with the public key can then be a **Sender** who can publicly send secret messages to the **Receiver** —even if they have never communicated or shared any information besides the public key.
Here is how they do it:

Beforehand The **Receiver** creates a public key and a private key as follows.

1. Generate two distinct primes, p and q. These are used to generate the private key, and they must be kept hidden. (In current practice, p and q are chosen to be hundreds of digits long.)
2. Let $n ::= pq$.
3. Select an integer $e \in [0, n)$ such that $\gcd(e, (p-1)(q-1)) = 1$.
   The *public key* is the pair (e, n). This should be distributed widely.
4. Let the *private key* $d \in [0, n)$ be the inverse of e in the ring $\mathbb{Z}_{(p-1)(q-1)}$. This private key can be found using the Pulverizer. The private key d should be kept hidden!

Encoding To transmit a message $m \in [0, n)$ to **Receiver**, a **Sender** uses the public key to encrypt m into a numerical message

$$m^* ::= m^e \ (\mathbb{Z}_n).$$

The **Sender** can then publicly transmit m^* to the **Receiver**.

Decoding The **Receiver** decrypts message m^* back to message m using the private key:

$$m = (m^*)^d \ (\mathbb{Z}_n).$$

8.12 What has SAT got to do with it?

So why does the world, or at least the world's secret codes, fall apart if there is an efficient test for satisfiability (SAT) as we claimed in Section 3.5? To explain this, remember that RSA can be managed computationally because multiplication of two primes is fast, but factoring a product of two primes seems to be overwhelmingly demanding.

Now, designing digital multiplication circuits is completely routine. This means we can easily build a digital circuit out of AND, OR, and NOT gates that can take two input strings u, v of length n, and a third input string, z, of length $2n$, and "checks" if the numbers represented by u and v are both greater than 1 and that z represents their product. The circuit gives output 1 if z represents such a product and gives output 0 otherwise.

Now here's how to factor any number with a length $2n$ representation using a SAT solver. Fix the z input to be the representation of the number to be factored. Set the first digit of the u input to 1, and do a SAT test to see if there is a satisfying assignment of values for the remaining bits of u and v. That is, see if the remaining bits of u and v can be filled in to cause the circuit to give output 1. If there is such an assignment, fix the first bit of u to 1, otherwise fix the first bit of u to be 0. Now do the same thing to fix the second bit of u and then third, proceeding in this way through all the bits of u and then of v. The result is that after $2n$ SAT tests, we have found an assignment of values for u and v that makes the circuit give output 1. So u and v represent factors of the number represented by z. This means that if SAT could be done in time bounded by a degree d polynomial in n, then $2n$ digit numbers can be factored in time bounded by a polynomial in n of degree $d + 1$. In sum, if SAT was easy, then so is factoring, and so RSA would be easy to break.

Problems for Section 8.1

Practice Problems

Problem 8.1.
Prove that a linear combination of linear combinations of integers $a_0, \ldots, a_n$ is a linear combination of $a_0, \ldots, a_n$.

Problem 8.2. **(a)** Find integer coefficients, x, y, such that $25x+32y = \text{GCD}(25, 32)$.

(b) What is the inverse (mod 25) of 32?

Class Problems

Problem 8.3.
A number is *perfect* if it is equal to the sum of its positive divisors, other than itself. For example, 6 is perfect, because $6 = 1 + 2 + 3$. Similarly, 28 is perfect, because $28 = 1 + 2 + 4 + 7 + 14$. Explain why $2^{k-1}(2^k - 1)$ is perfect when $2^k - 1$ is prime.[15]

Problems for Section 8.2

Practice Problems

Problem 8.4.
Let

$$\begin{aligned} x &::= 21212121, \\ y &::= 12121212. \end{aligned}$$

Use the Euclidean algorithm to find the GCD of x and y. *Hint:* Looks scary, but it's not.

Problem 8.5.
Let

$$\begin{aligned} x &::= 17^{88} * 31^5 * 37^2 * 59^{1000} \\ y &::= 19^{(9^{22})} * 37^{12} * 53^{3678} * 59^{29}. \end{aligned}$$

(a) What is $\gcd(x, y)$?

(b) What is $\operatorname{lcm}(x, y)$?

(lcm is *least common multiple*.)

Class Problems

Problem 8.6.
Use the Euclidean Algorithm to prove that

$$\gcd(13a + 8b, 5a + 3b) = \gcd(a, b).$$

[15]Euclid proved this 2300 years ago. About 250 years ago, Euler proved the converse: *every* even perfect number is of this form (for a simple proof see `http://primes.utm.edu/notes/proofs/EvenPerfect.html`). As is typical in number theory, apparently simple results lie at the brink of the unknown. For example, it is not known if there are an infinite number of even perfect numbers or any odd perfect numbers at all.

Problem 8.7.

(a) Use the Pulverizer to find integers x, y such that

$$x30 + y22 = \gcd(30, 22).$$

(b) Now find integers x', y' with $0 \leq y' < 30$ such that

$$x'30 + y'22 = \gcd(30, 22)$$

Problem 8.8.
For nonzero integers, a, b, prove the following properties of divisibility and GCD'S. (You may use the fact that $\gcd(a, b)$ is an integer linear combination of a and b. You may *not* appeal to uniqueness of prime factorization because the properties below are needed to *prove* unique factorization.)

(a) Every common divisor of a and b divides $\gcd(a, b)$.

(b) If $a \mid bc$ and $\gcd(a, b) = 1$, then $a \mid c$.

(c) If $p \mid bc$ for some prime, p, then $p \mid b$ or $p \mid c$.

(d) Let m be the smallest integer linear combination of a and b that is positive. Show that $m = \gcd(a, b)$.

Homework Problems

Problem 8.9.
Define the Pulverizer State machine to have:

$$\begin{aligned}
\text{states} &::= \mathbb{N}^6 \\
\text{start state} &::= (a, b, 0, 1, 1, 0) \qquad \text{(where } a \geq b > 0) \\
\text{transitions} &::= (x, y, s, t, u, v) \longrightarrow \\
&\quad (y,\ \text{rem}\,(x,\ y)\,,\ u - sq,\ v - tq,\ s,\ t) \quad (\text{for } q = \text{qcnt}(x, y), y > 0).
\end{aligned}$$

(a) Show that the following properties are preserved invariants of the Pulverizer machine:

$$\gcd(x, y) = \gcd(a, b), \tag{8.19}$$
$$sa + tb = y, \text{ and} \tag{8.20}$$
$$ua + vb = x. \tag{8.21}$$

(b) Conclude that the Pulverizer machine is partially correct.

(c) Explain why the machine terminates after at most the same number of transitions as the Euclidean algorithm.

Problem 8.10.
Prove that the smallest positive integers $a \geq b$ for which, starting in state (a, b), the Euclidean state machine will make n transitions are $F(n+1)$ and $F(n)$, where $F(n)$ is the nth Fibonacci number.

Hint: Induction.

In a later chapter, we'll show that $F(n) \leq \varphi^n$ where φ is the golden ratio $(1+\sqrt{5})/2$. This implies that the Euclidean algorithm halts after at most $\log_{\varphi}(a)$ transitions. This is a somewhat smaller than the $2\log_2 a$ bound derived from equation (8.4).

Problem 8.11.
Let's extend the jug filling scenario of Section 8.1.3 to three jugs and a receptacle. Suppose the jugs can hold a, b, and c gallons of water, respectively.

The receptacle can be used to store an unlimited amount of water, but has no measurement markings. Excess water can be dumped into the drain. Among the possible moves are:

1. fill a bucket from the hose,
2. pour from the receptacle to a bucket until the bucket is full or the receptacle is empty, whichever happens first,
3. empty a bucket to the drain,
4. empty a bucket to the receptacle,
5. pour from one bucket to another until either the first is empty or the second is full,

(a) Model this scenario with a state machine. (What are the states? How does a state change in response to a move?)

(b) Prove that Bruce can get $k \in \mathbb{N}$ gallons of water into the receptacle using the above operations only if $\gcd(a, b, c) \mid k$.

(c) Prove conversely, that if $\gcd(a, b, c) \mid k$, then Bruce can get actually get k gallons of water into the receptacle.

Problem 8.12.
The binary-GCD state machine computes the GCD of a and b using only division by 2 and subtraction, which makes it run very efficiently on hardware that uses binary representation of numbers. In practice, it runs more quickly than the Euclidean algorithm state machine (8.3).

$$\begin{aligned}
\text{states} &::= \mathbb{N}^3 \\
\text{start state} &::= (a, b, 1) \qquad\qquad \text{(where } a > b > 0\text{)} \\
\text{transitions} &::= \text{ if } \min(x, y) > 0, \text{ then } (x, y, e) \longrightarrow \\
&\quad \text{the first possible state according to the rules:} \\
&\quad \begin{cases}
(1, 0, ex) & (\text{if } x = y) \\
(1, 0, e) & (\text{if } y = 1), \\
(x/2, y/2, 2e) & (\text{if } 2 \mid x \text{ and } 2 \mid y), \\
(y, x, e) & (\text{if } y > x) \\
(x, y/2, e) & (\text{if } 2 \mid y) \\
(x/2, y, e) & (\text{if } 2 \mid x) \\
(x - y, y, e) & (\text{otherwise}).
\end{cases}
\end{aligned}$$

(a) Prove that if this machine reaches a "final" state (x, y, e) in which no transition is possible, then $e = \gcd(a, b)$.

(b) Prove that the machine reaches a final state in at most $3 + 2 \log \max(a, b)$ transitions.

Hint: Strong induction on $\max(a, b)$.

Exam Problems

Problem 8.13.
Prove that $\gcd(mb + r, b) = \gcd(b, r)$ for all integers m, b, r.
Hint: We proved a similar result in class when r was a remainder in $[0, b)$.

Problems for Section 8.3

Homework Problems

Problem 8.14.

TBA - Chebyshvev lower bound in prime density, based on Shoup pp.75–76

Problems for Section 8.4

Class Problems

Problem 8.15. **(a)** Let $m = 2^9 5^{24} 11^7 17^{12}$ and $n = 2^3 7^{22} 11^{211} 13^1 17^9 19^2$. What is the $\gcd(m,n)$? What is the *least common multiple*, $\operatorname{lcm}(m,n)$, of m and n? Verify that

$$\gcd(m,n) \cdot \operatorname{lcm}(m,n) = mn. \tag{8.22}$$

(b) Describe in general how to find the $\gcd(m,n)$ and $\operatorname{lcm}(m,n)$ from the prime factorizations of m and n. Conclude that equation (8.22) holds for all positive integers m, n.

Homework Problems

Problem 8.16.

The set of complex numbers that are equal to $m + n\sqrt{-5}$ for some integers m, n is called $\mathbb{Z}[\sqrt{-5}]$. It will turn out that in $\mathbb{Z}[\sqrt{-5}]$, not all numbers have unique factorizations.

A sum or product of numbers in $\mathbb{Z}[\sqrt{-5}]$ is in $\mathbb{Z}[\sqrt{-5}]$, and since $\mathbb{Z}[\sqrt{-5}]$ is a subset of the complex numbers, all the usual rules for addition and multiplication are true for it. But some weird things do happen. For example, the prime 29 has factors:

(a) Find $x, y \in \mathbb{Z}[\sqrt{-5}]$ such that $xy = 29$ and $x \neq \pm 1 \neq y$.

On the other hand, the number 3 is still a "prime" even in $\mathbb{Z}[\sqrt{-5}]$. More precisely, a number $p \in \mathbb{Z}[\sqrt{-5}]$ is called *irreducible* over $\mathbb{Z}[\sqrt{-5}]$ iff when $xy = p$ for some $x, y \in \mathbb{Z}[\sqrt{-5}]$, either $x = \pm 1$ or $y = \pm 1$.

Claim. *The numbers* $3, 2 + \sqrt{-5}$*, and* $2 - \sqrt{-5}$ *are irreducible over* $\mathbb{Z}[\sqrt{-5}]$.

In particular, this Claim implies that the number 9 factors into irreducibles over $\mathbb{Z}[\sqrt{-5}]$ in two different ways:

$$3 \cdot 3 = 9 = (2 + \sqrt{-5})(2 - \sqrt{-5}). \tag{8.23}$$

So $\mathbb{Z}[\sqrt{-5}]$ is an example of what is called a *non-unique factorization* domain.

To verify the Claim, we'll appeal (without proof) to a familiar technical property of complex numbers given in the following Lemma.

Definition. For a complex number $c = r + si$ where $r, s \in \mathbb{R}$ and i is $\sqrt{-1}$, the *norm*, $|c|$, of c is $\sqrt{r^2 + s^2}$.

Lemma. *For $c, d \in \mathbb{C}$,*

$$|cd| = |c|\,|d|\,.$$

(b) Prove that $|x|^2 \neq 3$ for all $x \in \mathbb{Z}[\sqrt{-5}]$.

(c) Prove that if $x \in \mathbb{Z}[\sqrt{-5}]$ and $|x| = 1$, then $x = \pm 1$.

(d) Prove that if $|xy| = 3$ for some $x, y \in \mathbb{Z}[\sqrt{-5}]$, then $x = \pm 1$ or $y = \pm 1$.
Hint: $|z|^2 \in \mathbb{N}$ for $z \in \mathbb{Z}[\sqrt{-5}]$.

(e) Complete the proof of the Claim.

Problems for Section 8.6

Class Problems

Problem 8.17. (a) Prove if n is not divisible by 3, then $n^2 \equiv 1 \pmod{3}$.

(b) Show that if n is odd, then $n^2 \equiv 1 \pmod{8}$.

(c) Conclude that if p is a prime greater than 3, then $p^2 - 1$ is divisible by 24.

Problem 8.18.
The values of polynomial $p(n) ::= n^2 + n + 41$ are prime for all the integers from 0 to 39 (see Section 1.1). Well, p didn't work, but are there any other polynomials whose values are always prime? No way! In fact, we'll prove a much stronger claim.

Suppose q is a polynomial with integer coefficients whose domain is restricted to be the nonnegative integers. We'll say that q *produces multiples* if, for every nonzero value in the range of q, there are infinitely many multiples of that value also in the range.

For example, if q produces multiples and $q(4) = 7$, then there are infinitely many different multiples of 7 in the range of q, and of course, except for 7 itself, none of these multiples is prime.

Claim. *If q is not a constant function, then q produces multiples.*

(a) Prove that if $j \equiv k \pmod{n}$, then $q(j) \equiv q(k) \pmod{n}$.

Hint: The set, A, of polynomial functions with integer coefficients can be defined recursively:

- **Base cases**:
  - the identity function, $i(x) ::= x$ is in A.
  - for any integer, k, the constant function, $c(x) ::= k$ is in A.
- **Constructor cases**. If $r, s \in A$, then $r + s$ and $r \cdot s \in A$.

(b) Prove the Claim 8.18.

Claim 8.18 implies that if an integer polynomial is not constant then its range includes infinitely many nonprimes. This fact no longer holds true for multivariate polynomials. An amazing consequence of Matijesevich's solution to Hilbert's Tenth Problem, **TBA - reference**, is that multivariate polynomials can be understood as *general purpose* programs for generating sets of integers. If a set of nonnegative integers can be generated by *any* program, then it equals the set of nonnegative integers in the range of a multivariate integer polynomial! In particular, there is an integer polynomial $p(x_1, \ldots, x_7)$ whose nonnegative values as $x_1, \ldots, x_7$ range over $\mathbb{N}$ are precisely the set of all prime numbers!

Problems for Section 8.7

Practice Problems

Problem 8.19.
A majority of the following statements are equivalent to each other. List all statements in this majority. Assume that $n > 0$ and a and b are integers. Briefly explain your reasoning.

1. $a \equiv b \pmod{n}$
2. $a = b$
3. $\text{rem}(a, n) = \text{rem}(b, n)$
4. $n \mid (a - b)$
5. $\exists k \in \mathbb{Z}. a = b + nk$
6. $(a - b)$ is a multiple of n
7. $n \mid a$ OR $n \mid b$

Homework Problems

Problem 8.20.
Prove that congruence is preserved by arithmetic expressions. Namely, prove that

$$a \equiv b \pmod{n}, \tag{8.24}$$

then

$$\text{eval}(e, a) \equiv \text{eval}(e, b) \pmod{n}, \tag{8.25}$$

for all $e \in$ Aexp (see Section 6.4).

Problem 8.21.
The sum of the digits of the base 10 representation of an integer is congruent modulo 9 to that integer. For example

$$763 \equiv 7 + 6 + 3 \pmod{9}.$$

This is not always true for the hexadecimal (base 16) representation, however. For example,

$$(763)_{16} = 7 \cdot 16^2 + 6 \cdot 16 + 3 \equiv 1 \not\equiv 7 \equiv 7 + 6 + 3 \pmod{9}.$$

(a) For exactly what integers $k > 1$ is it true that the sum of the digits of the base 16 representation of an integer is congruent modulo k to that integer? Justify your answer.

(b) Give a rule that generalizes this sum-of-digits rule from base $b = 16$ to an arbitrary number base $b > 1$, and explain why your rule is correct.

Class Problems

Problem 8.22.
Find

$$\text{remainder}\left(9876^{3456789}\left(9^{99}\right)^{5555} - 6789^{3414259},\ 14\right). \tag{8.26}$$

Problem 8.23.
The following properties of equivalence mod n follow directly from its definition and simple properties of divisibility. See if you can prove them without looking up the proofs in the text.

(a) If $a \equiv b \pmod{n}$, then $ac \equiv bc \pmod{n}$.

(b) If $a \equiv b \pmod{n}$ and $b \equiv c \pmod{n}$, then $a \equiv c \pmod{n}$.

(c) If $a \equiv b \pmod{n}$ and $c \equiv d \pmod{n}$, then $ac \equiv bd \pmod{n}$.

(d) $\text{rem}(a, n) \equiv a \pmod{n}$.

Problem 8.24. **(a)** Why is a number written in decimal evenly divisible by 9 if and only if the sum of its digits is a multiple of 9? *Hint:* $10 \equiv 1 \pmod{9}$.

(b) Take a big number, such as 37273761261. Sum the digits, where every other one is negated:

$$3 + (-7) + 2 + (-7) + 3 + (-7) + 6 + (-1) + 2 + (-6) + 1 = -11$$

Explain why the original number is a multiple of 11 if and only if this sum is a multiple of 11.

Problem 8.25.
At one time, the Guinness Book of World Records reported that the "greatest human calculator" was a guy who could compute 13th roots of 100-digit numbers that were powers of 13. What a curious choice of tasks....

In this problem, we prove

$$n^{13} \equiv n \pmod{10} \tag{8.27}$$

for all n.

(a) Explain why (8.27) does not follow immediatetly from Euler's Theorem.

(b) Prove that

$$d^{13} \equiv d \pmod{10} \tag{8.28}$$

for $0 \le d < 10$.

(c) Now prove the congruence (8.27).

Problem 8.26. **(a)** Ten pirates find a chest filled with gold and silver coins. There are twice as many silver coins in the chest as there are gold. They divide the gold coins in such a way that the difference in the number of coins given to any two pirates is not divisible by 10. They will only take the silver coins if it is possible to divide them the same way. Is this possible, or will they have to leave the silver behind? Prove your answer.

(b) There are also 3 sacks in the chest, containing 5, 49, and 51 rubies respectively. The treasurer of the pirate ship is bored and decides to play a game with the following rules:

- He can merge any two piles together into one pile, and
- he can divide a pile with an even number of rubies into two piles of equal size.

He makes one move every day, and he will finish the game when he has divided the rubies into 105 piles of one. Is it possible for him to finish the game?

Exam Problems

Problem 8.27.
We define the sequence of numbers

$$a_n = \begin{cases} a_{n-1} + a_{n-2} + a_{n-3} + a_{n-4} & \text{if } n \geq 4, \\ 1 & \text{if } 0 \leq n \leq 3. \end{cases}$$

Prove that $a_n \equiv 1 \pmod 3$ for all $n \geq 0$.

Problems for Section 8.8

Exam Problems

Problem 8.28.
The set Aexp of Arithmetic Expressions in the variable x was defined recursively: expressions consisting solely of the variable x or an arabic numeral, k, were the base cases, and the contructors were forming the sum, $[\, e_1 + e_2 \,]$, product, $[\, e_1 * e_2 \,]$, or minus $-[\, e_1 \,]$ of Aexp’s e_1, e_2. Then the value $\text{eval}(e, n)$ of an Aexp e when the variable x is equal to the integer n has an immediate recursive definition based on the definition of Aexp’s.

Prove by structural induction that for all Aexp’s e,

$$\forall m, n, d \in \mathbb{Z}, d > 1.\, [m \equiv n \pmod d] \quad \text{IMPLIES} \quad [\text{eval}(e, m) \equiv \text{eval}(e, n) \pmod d]. \tag{8.29}$$

Hint: Be sure to consider **both base cases**. The proofs for the three constructors are very similar, so **just write out the case for the sum constructor**.

Problems for Section 8.9

Practice Problems

Problem 8.29.
What is the multiplicative inverse (mod 7) of 2? *Reminder*: by definition, your answer must be an integer between 0 and 6.

Problem 8.30. (a) Use the Pulverizer to find integers s, t such that

$$40s + 7t = \gcd(40, 7).$$

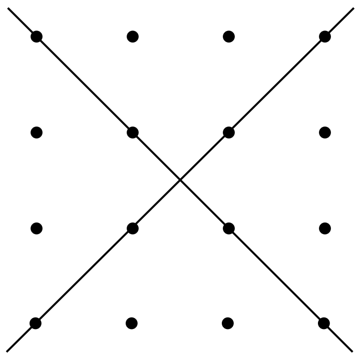
(b) Adjust your answer to part (a) to find an inverse modulo 40 of 7 in $[1, 40)$.

Class Problems

Problem 8.31.
Two nonparallel lines in the real plane intersect at a point. Algebraically, this means that the equations

$$\begin{aligned} y &= m_1 x + b_1 \\ y &= m_2 x + b_2 \end{aligned}$$

have a unique solution (x, y), provided $m_1 \neq m_2$. This statement would be false if we restricted x and y to the integers, since the two lines could cross at a noninteger point:



However, an analogous statement holds if we work over the integers *modulo a* *prime*, p. Find a solution to the congruences

$$y \equiv m_1 x + b_1 \pmod{p}$$
$$y \equiv m_2 x + b_2 \pmod{p}$$

when $m_1 \not\equiv m_2 \pmod{p}$. Express your solution in the form $x \equiv ? \pmod{p}$ and $y \equiv ? \pmod{p}$ where the ?'s denote expressions involving m_1, m_2, b_1, and b_2. You may find it helpful to solve the original equations over the reals first.

Problems for Section 8.10

Practice Problems

Problem 8.32.
Prove that $k \in [0, n)$ has an inverse modulo n iff it has an inverse in $\mathbb{Z}_n$.

Problem 8.33.
What is $\text{rem}\left(24^{78},\ 79\right)$? *Hint:* 79 is prime. You should not need to do any calculation!

Problem 8.34. **(a)** Prove that 22^{12001} has a multiplicative inverse modulo 175.

(b) What is the value of $\phi(175)$, where ϕ is Euler's function?

(c) What is the remainder of 22^{12001} divided by 175?

Problem 8.35.
How many numbers between 1 and 6042 (inclusive) are relatively prime to 3780? Hint: 53 is a factor.

Problem 8.36.
How many numbers between 1 and 3780 (inclusive) are relatively prime to 3780?

Problem 8.37.

(a) What is the probability that an integer from 1 to 360 selected with uniform probability is relatively prime to 360?

(b) What is the value of $\text{rem}\left(7^{98},\ 360\right)$?

Class Problems

Problem 8.38.
Find the last digit of $7^{7^{7^7}}$.

Problem 8.39.
Use Fermat’s theorem to find the inverse, i, of 13 modulo 23 with $1 \leq i < 23$.

Problem 8.40.
Let $S_k = 1^k + 2^k + \ldots + (p-1)^k$, where p is an odd prime and k is a positive multiple of $p-1$. Use Fermat’s theorem to prove that $S_k \equiv -1 \pmod{p}$.

Problem 8.41.
Let a and b be relatively prime positive integers.

(a) How many integers in the interval $[0, ab)$ are divisible by a?

(b) How many integers in the interval $[0, ab)$ are divisible by both a and b?

(c) How many integers in the interval $[0, ab)$ are divisible by either a or b?

(d) Now suppose $p \neq q$ are both primes. How many integers in the interval $[0, pq)$ are *not* relatively prime to pq? Observe that a different answer is required if p and q were merely relatively prime numbers a and b as in part (c).

(e) Conclude that

$$\phi(pq) = (p-1)(q-1).$$

Problem 8.42.
Suppose a, b are relatively prime and greater than 1. In this problem you will prove the *Chinese Remainder Theorem*, which says that for all m, n, there is an x such

that

$$x \equiv m \bmod a, \tag{8.30}$$
$$x \equiv n \bmod b. \tag{8.31}$$

Moreover, x is unique up to congruence modulo ab, namely, if x' also satisfies (8.30) and (8.31), then

$$x' \equiv x \bmod ab.$$

(a) Prove that for any m, n, there is some x satisfying (8.30) and (8.31).

Hint: Let b^{-1} be an inverse of b modulo a and define $e_a ::= b^{-1}b$. Define e_b similarly. Let $x = me_a + ne_b$.

(b) Prove that

$$[x \equiv 0 \bmod a \text{ AND } x \equiv 0 \bmod b] \quad \text{implies} \quad x \equiv 0 \bmod ab.$$

(c) Conclude that

$$\left[x \equiv x' \bmod a \text{ AND } x \equiv x' \bmod b\right] \quad \text{implies} \quad x \equiv x' \bmod ab.$$

(d) Conclude that the Chinese Remainder Theorem is true.

(e) What about the converse of the implication in part (c)?

Homework Problems

Problem 8.43.
Suppose a, b are relatively prime integers greater than 1. In this problem you will prove that Euler's function is *multiplicative*, namely, that

$$\phi(ab) = \phi(a)\phi(b).$$

The proof is an easy consequence of the Chinese Remainder Theorem (Problem 8.42).

(a) Conclude from the Chinese Remainder Theorem that the function $f : [0, ab) \to [0, a) \times [0, b)$ defined by

$$f(x) ::= (\text{rem}(x, a), \text{rem}(x, b))$$

is a bijection.

(b) For any positive integer, k, let gcd1$\{k\}$ be the integers in $[0, k)$ that are relatively prime to k. Prove that the function f from part (a) also defines a bijection from gcd1$\{ab\}$ to gcd1$\{a\} \times$ gcd1$\{b\}$.

(c) Conclude from the preceding parts of this problem that

$$\phi(ab) = \phi(a)\phi(b). \tag{8.32}$$

(d) Prove Corollary 8.10.14: for any number $n > 1$, if $p_1, p_2, \ldots, p_j$ are the (distinct) prime factors of n, then

$$\phi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\cdots\left(1 - \frac{1}{p_j}\right).$$

Problem 8.44.
The general version of the Chinese Remainder theorem (Problem 8.42) extends to more than two relatively prime moduli. Namely,

Theorem (General Chinese Remainder)**.** *Suppose $a_1, \ldots, a_k$ are integers greater than 1 and each is relatively prime to the others. Let $n ::= a_1 \cdot a_2 \cdots a_k$. Then for any integers $m_1, m_2, \ldots, m_k$, there is a unique $x \in [0, n)$ such that*

$$x \equiv m_i \pmod{a_i},$$

for $1 \le i \le k$.

The proof is a routine induction on k using a fact that follows immediately from unique factorization: if a number is relatively prime to some other numbers, then it is relatively prime to their product.

Now suppose an n-bit number, N, was a product of relatively prime k-bit numbers, where n was big, but k was small enough to be handled by cheap and available arithmetic hardware units. Suppose a calculation requiring a large number of additions and multiplications modulo N had to be performed starting with some small set of n-bit numbers. For example, suppose we wanted to compute

$$\text{rem}\left((x-3)^{110033}((y+7)^{27123} - z^{4328})\ ,\ N\right)$$

which would require several dozen n-bit operations starting from the three numbers x, y, z.

Doing a multiplication or addition modulo N directly requires breaking up the n-bit numbers x, y, z and all the intermediate results of the mod N calculation into k-bit pieces, using the hardware to perform the additions and multiplications on the pieces, and then reassembling the k-bit results into an n-bit answer after each operation. Suppose N was a product of m relatively prime k-bit numbers.

Explain how the General Chinese Remainder Theorem offers a far more efficient approach to performing the required operations.

Exam Problems

Problem 8.45.
Prove that if k_1 and k_2 are relatively prime to n, then so is $k_1 \cdot_n k_2$,

(a) …using the fact that k is relatively prime to n iff k has an inverse modulo n

Hint: Recall that $k_1k_2 \equiv k_1 \cdot_n k_2 \pmod{n}$.

(b) …using the fact that k is relatively prime to n iff k is cancellable modulo n.

(c) …using the Unique Factorization Theorem and the basic GCD properties such as Lemma 8.2.1.

Problem 8.46.

Circle **true** or **false** for the statements below, and *provide counterexamples* for those that are **false**. Variables, a, b, c, m, n range over the integers and $m, n > 1$.

(a) $\gcd(1 + a, 1 + b) = 1 + \gcd(a, b)$. **true** **false**

(b) If $a \equiv b \pmod{n}$, then $p(a) \equiv p(b) \pmod{n}$
for any polynomial $p(x)$ with integer coefficients. **true** **false**

(c) If $a \mid bc$ and $\gcd(a, b) = 1$, then $a \mid c$. **true** **false**

(d) $\gcd(a^n, b^n) = (\gcd(a, b))^n$ **true** **false**

(e) If $\gcd(a, b) \neq 1$ and $\gcd(b, c) \neq 1$, then $\gcd(a, c) \neq 1$. **true** **false**

(f) If an integer linear combination of a and b equals 1,
then so does some integer linear combination of a^2 and b^2. **true** **false**

(g) If no integer linear combination of a and b equals 2,
then neither does any integer linear combination of a^2 and b^2. **true** **false**

(h) If $ac \equiv bc \pmod{n}$ and n does not divide c,
then $a \equiv b \pmod{n}$. **true** **false**

(i) Assuming a, b have inverses modulo n,
if $a^{-1} \equiv b^{-1} \pmod{n}$, then $a \equiv b \pmod{n}$. **true** **false**

(j) If $ac \equiv bc \pmod{n}$ and n does not divide c,
then $a \equiv b \pmod{n}$. **true** **false**

(k) If $a \equiv b \pmod{\phi(n)}$ for $a, b > 0$, then $c^a \equiv c^b \pmod{n}$. **true** **false**

(l) If $a \equiv b \pmod{nm}$, then $a \equiv b \pmod{n}$. **true** **false**

(m) If $\gcd(m, n) = 1$, then
$[a \equiv b \pmod{m}$ AND $a \equiv b \pmod{n}]$ iff $[a \equiv b \pmod{mn}]$ **true** **false**

(n) If $\gcd(a, n) = 1$, then $a^{n-1} \equiv 1 \pmod{n}$ **true** **false**

(o) If $a, b > 1$, then
[a has a inverse mod b iff b has an inverse mod a]. **true** **false**

Problem 8.47.
Find the remainder of $26^{1818181}$ divided by 297.
Hint: $1818181 = (180 \cdot 10101) + 1$; use Euler's theorem.

Problem 8.48.
Find an integer $k > 1$ such that n and n^k agree in their last three digits whenever n is divisible by neither 2 nor 5. *Hint:* Euler's theorem.

Problem 8.49.
What is the remainder of 63^{9601} divided by 220?

Problem 8.50.

(a) Explain why $(-12)^{482}$ has a multiplicative inverse modulo 175.

(b) What is the value of $\phi(175)$, where ϕ is Euler's function?

(c) Call a number from 0 to 174 *powerful* iff some positive power of the number is congruent to 1 modulo 175. What is the probability that a random number from 0 to 174 is powerful?

(d) What is the remainder of $(-12)^{482}$ divided by 175?

Problem 8.51. (a) Calculate the remainder of 35^{86} divided by 29.

(b) Part (a) implies that the remainder of 35^{86} divided by 29 is not equal to 1. So there there must be a mistake in the following proof, where all the congruences are taken with modulus 29:

$$\begin{aligned}
1 &\not\equiv 35^{86} && \text{(by part (a))} && (8.33)\\
&\equiv 6^{86} && \text{(since } 35 \equiv 6 \pmod{29}) && (8.34)\\
&\equiv 6^{28} && \text{(since } 86 \equiv 28 \pmod{29}) && (8.35)\\
&\equiv 1 && \text{(by Fermat's Little Theorem)} && (8.36)
\end{aligned}$$

Identify the exact line containing the mistake and explain the logical error.

Problem 8.52.
Give counterexamples for each of the statements below that are false.

(a) For integers a and b there are integers x and y such that: $ax + by = 1$

(b) $\gcd(mb + r, b) = \gcd(r, b)$ for all integers m, r and b.

(c) For every prime p and every integer k, $k^{p-1} \equiv 1 \pmod{p}$.

(d) For primes $p \neq q$, $\phi(pq) = (p-1)(q-1)$, where ϕ is Euler's totient fucntion.

(e) Suppose $a, b, c, d \in \mathbb{N}$ and a and b are relatively prime to d. Then

$$[ac \equiv bc \bmod d] \quad \text{IMPLIES} \quad [a \equiv b \bmod d].$$

Problems for Section 8.11

Practice Problems

Problem 8.53.
Suppose a cracker knew how to factor the RSA modulus n into the product of distinct primes p and q. Explain how the cracker could use the public key-pair (e, n) to find a private key-pair (d, n) that would allow him to read any message encrypted with the public key.

Problem 8.54.
Suppose the RSA modulus $n = pq$ is the product of distinct 200 digit primes p and q. A message $m \in [0, n)$ is called *dangerous* if $\gcd(m, n) = p$, because such an m can be used to factor n and so crack RSA. Circle the best estimate of the fraction of messages in $[0, n)$ that are dangerous.

$$\frac{1}{200} \qquad \frac{1}{400} \qquad \frac{1}{200^{10}} \qquad \frac{1}{10^{200}} \qquad \frac{1}{400^{10}} \qquad \frac{1}{10^{400}}$$

Class Problems

Problem 8.55.
Let's try out RSA!

(a) Go through the **beforehand** steps.

- Choose primes p and q to be relatively small, say in the range 10-40. In practice, p and q might contain hundreds of digits, but small numbers are easier to handle with pencil and paper.
- Try $e = 3, 5, 7, \ldots$ until you find something that works. Use Euclid's algorithm to compute the gcd.
- Find d (using the Pulverizer or Euler's Theorem).

When you're done, put your public key on the board. This lets another team send you a message.

(b) Now send an encrypted message to another team using their public key. Select your message m from the codebook below:

- 2 = Greetings and salutations!
- 3 = Yo, wassup?
- 4 = You guys are slow!
- 5 = All your base are belong to us.
- 6 = Someone on *our* team thinks someone on *your* team is kinda cute.
- 7 = You *are* the weakest link. Goodbye.

(c) Decrypt the message sent to you and verify that you received what the other team sent!

Problem 8.56. **(a)** Just as RSA would be trivial to crack knowing the factorization into two primes of n in the public key, explain why RSA would also be trivial to crack knowing $\phi(n)$.

(b) Show that if you knew n, $\phi(n)$, and that n was the product of two primes, then you could easily factor n.

Hint: Suppose $n = pq$, replace q by n/p in the expression for $\phi(n)$, and solve for p.

Problem 8.57.
A critical fact about RSA is, of course, that decrypting an encrypted message, m^*, always gives back the original message, m. Namely, if $n = pq$ where p and q are distinct primes, $m \in [0, pq)$, and

$$d \cdot e \equiv 1 \pmod{(p-1)(q-1)},$$

then

$$\text{rem}\left(\text{rem}\left(m^d, n\right)^e, n\right) = m. \tag{8.37}$$

We'll now prove this.

(a) Verify that if

$$(m^d)^e \equiv m \pmod{n}, \tag{8.38}$$

then (8.37) is true.

(b) Prove that if p is prime, then $m^a \equiv m \pmod{p}$ for all $a \in \mathbb{N}$ congruent to 1 mod $p - 1$.

(c) Prove that if $a \equiv b \pmod{p_i}$ for distinct primes $p_1, p_2, \dots, p_n$, then $a \equiv b \pmod{p_1 p_1 \cdots p_n}$.

(d) Prove
Lemma. *If n is a product of distinct primes and $a \in \mathbb{N}$ is $\equiv 1 \pmod{\phi(n)}$, then $m^a \equiv m \pmod{n}$.*

(e) Combine the previous parts to complete the proof of (8.37).

Homework Problems

Problem 8.58.
Although RSA has successfully withstood cryptographic attacks for a more than a

quarter century, it is not known that breaking RSA would imply that factoring is easy.

In this problem we will examine the *Rabin cryptosystem* that does have such a security certification. Namely, if someone has the ability to break the Rabin cryptosystem efficiently, then they also have the ability to factor numbers that are products of two primes.

Why should that convince us that it is hard to break the cryptosystem efficiently? Well, mathematicians have been trying to factor efficiently for centuries, and they still haven't figured out how to do it.

What is the Rabin cryptosystem? The public key will be a number N that is a product of two very large primes p, q such that $p \equiv q \equiv 3 \pmod{4}$. To send the message x, send $\text{rem}\left(x^2,\ N\right)$.[16]

The private key is the factorization of N, namely, the primes p, q. We need to show that if the person being sent the message knows p, q, then they can decode the message. On the other hand, if an eavesdropper who doesn't know p, q listens in, then we must show that they are very unlikely to figure out this message.

First some definitions. We know what it means for a number to be a square over the integers, that is s is a square if there is another integer x such that $s = x^2$. Over the numbers mod N, we say that s is a *square modulo* N if there is an x such that $s \equiv x^2 \pmod{N}$. If x is such that $0 \le x < N$ and $s \equiv x^2 \pmod{N}$, then x is the *square root of* s.

(a) What are the squares modulo 5? For each nonzero square in the interval $[0, 5)$, how many square roots does it have?

(b) For each integer in $[1, 15)$ that is relatively prime to 15, how many square roots (modulo 15) does it have? Note that all the square roots are *also* relatively prime to 15. We won't go through why this is so here, but keep in mind that this is a general phenomenon!

(c) Suppose that p is a prime such that $p \equiv 3 \pmod{4}$. It turns out that squares modulo p have exactly 2 square roots. First show that $(p + 1)/4$ is an integer. Next figure out the two square roots of 1 modulo p. Then show that you can find a "square root mod a prime p" of a number by raising the number to the $(p+1)/4$th power. That is, given s, to find x such that $s \equiv x^2 \pmod{p}$, you can compute $\text{rem}\left(s^{(p+1)/4},\ p\right)$.

(d) The Chinese Remainder Theorem (Problem 8.42) implies that if p, q are dis-

[16]We will see soon, that there are other numbers that would be encrypted by $\text{rem}\left(x^2,\ N\right)$, so we'll have to disallow those other numbers as possible messages in order to make it possible to decode this cryptosystem, but let's ignore that for now.

tinct primes, then s is a square modulo pq if and only if s is a square modulo p and s is a square modulo q. In particular, if $s \equiv x^2 \pmod{p} \equiv (x')^2 \pmod{p}$ and $s \equiv y^2 \pmod{p} \equiv (y')^2 \pmod{p}$ then s has exactly four square roots, namely,

$$s \equiv (xy)^2 \equiv (x'y)^2 \equiv (xy')^2 \equiv (x'y')^2 \pmod{pq}.$$

So, if you know p, q, then using the solution to part (c), you can efficiently find the square roots of s! Thus, given the private key, decoding is easy.

But what if you don't know p, q? Suppose $N ::= pq$, where p, q are two primes equivalent to $3 \pmod 4$. Let's assume that the evil message interceptor claims to have a program that can find all four square roots of any number modulo N. Show that he can actually use this program to efficiently find the factorization of N. Thus, unless this evil message interceptor is extremely smart and has figured out something that the rest of the scientific community has been working on for years, it is very unlikely that this efficient square root program exists!

Hint: Pick r arbitrarily from $[1, N)$. If $gcd(N, r) > 1$, then you are done (why?) so you can halt. Otherwise, use the program to find all four square roots of r, call them $r, -r, r', -r'$. Note that $r^2 \equiv r'^2 \pmod{N}$. How can you use these roots to factor N?

(e) If the evil message interceptor knows that the message is the encoding one of two possible candidate messages (that is, either “meet at dome at dusk” or “meet at dome at dawn”) and is just trying to figure out which of the two, then can he break this cryptosystem?

Problem 8.59.
You've seen how the RSA encryption scheme works, but why is it hard to break? In this problem, you will see that finding private keys is as hard as finding the prime factorizations of integers. Since there is a general consensus in the crypto community (enough to persuade many large financial institutions, for example) that factoring numbers with a few hundred digits requires astronomical computing resources, we can therefore be sure it will take the same kind of overwhelming effort to find RSA private keys of a few hundred digits. This means we can be confident the private RSA keys are not somehow revealed by the public keys [17]

For this problem, assume that $n = p \cdot q$ where p, q are both *odd* primes and that e is the public key and d the private key of the RSA protocol.. Let $x ::= e \cdot d - 1$.

[17] This is a very weak kind of “security” property, because it doesn't even rule out the possibility of deciphering RSA encoded messages by some method that did not require knowing the private key. Nevertheless, over twenty years experience supports the security of RSA in practice.

(a) Show that $\phi(n)$ divides x.

(b) Conclude that 4 divides x.

(c) Show that if $\gcd(r, n) = 1$, then $r^x \equiv 1 \pmod{n}$.

A *square root* of m modulo n is a nonnegative integer $s < n$ such that $s^2 \equiv m \pmod{n}$. Here is a nice fact to know: when n is a product of two odd primes, then every number m such that $\gcd(m, n) = 1$ has 4 square roots modulo n.

In particular, the number 1 has four square roots modulo n. The two trivial ones are 1 and $n - 1$ (which is $\equiv -1 \pmod{n}$). The other two are called the *nontrivial* square roots of 1.

(d) Since you know x, then for any integer, r, you can also compute the remainder, y, of $r^{x/2}$ divided by n. So $y^2 \equiv r^x \pmod{n}$. Now if r is relatively prime to n, then y will be a square root of 1 modulo n by part (c).

Show that if y turns out to be a *nontrivial* root of 1 modulo n, then you can factor n. *Hint:* From the fact that $y^2 - 1 = (y + 1)(y - 1)$, show that $y + 1$ must be divisible by exactly one of q and p.

(e) It turns out that at least half the positive integers $r < n$ that are relatively prime to n will yield y's in part (d) that are nontrivial roots of 1. Conclude that if, in addition to n and the public key, e, you also knew the private key d, then you can be sure of being able to factor n.

Index