Introduction to 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 there's the negatives. Which one don't you understand? After all, the mathematician G. H. Hardy 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.

What most concerned Hardy was that number theory not be used in warfare; he was a pacifist. Good for him, but if number theory is remote from *all* human activity, then why study it? We'll come back to that question later on, but ironically, we'll see that poor Hardy must be turning in his grave.

1 Divisibility

We'll be examining integer properties in these notes, so we'll adopt the convention that variables range over integers.

The true nature of number theory emerges from the first definition. We say that a ***divides*** b if there is an integer k such that $ak = b$. This is denoted $a \mid b$. For example:

$$7 \mid 63 \quad \text{because} \quad 7 \cdot 9 = 63$$

A consequence of this definition is that every number divides zero since $a \cdot 0 = 0$ for every integer a. If a divides b, then b is a ***multiple*** of a. For example, 63 is a multiple of 7.

This 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. 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. Interestingly, computer scientists have found ways to turn these difficulties to their advantage. Every time you buy a book from Amazon, check your grades on WebSIS, or use a PayPal account, you are relying on number theoretic algorithms.

DON'T PANIC— we're going to stick to some relatively benign parts of number theory. We won't put any of these super-hard unsolved problems on exams!

1.1 Facts About Divisibility

The lemma below states some basic facts about divisibility that are *not* difficult to prove:

Lemma 1.1. *The following statements about divisibility hold.*

1. *If $a \mid b$, then $a \mid bc$ for all c.*
2. *If $a \mid b$ and $b \mid c$, then $a \mid c$.*
3. *If $a \mid b$ and $a \mid c$, then $a \mid sb + tc$ for all s and t.*
4. *For all $c \neq 0$, $a \mid b$ if and only if $ca \mid cb$.*

Proof. We'll only prove part (2); the other proofs are similar.

Proof of (2): Since $a \mid b$, there exists an integer k_1 such that $ak_1 = b$. Since $b \mid c$, there exists an integer k_2 such that $bk_2 = c$. Substituting ak_1 for b in the second equation gives $ak_1k_2 = c$, which implies that $a \mid c$.

$\square$

A number $p > 1$ with no positive divisors other than 1 and itself is called a ***prime***. Every other number greater than 1 is called ***composite***. For example, 2, 3, 5, 7, 11, and 13 are all prime, but 4, 6, 8, and 9 are composite. The number 1 is considered neither prime nor composite. This is just a matter of definition, but reflects the fact that 1 does not behave like a prime in many contexts, such as the Fundamental Theorem of Arithmetic, which we'll come to shortly.

1.2 When Divisibility Goes Bad

As you learned in elementary school, if one number does *not* evenly divide another, then there is a "remainder" left over. More precisely, if you divide n by d, then you get a quotient q and a remainder r. This basic fact is the subject of a useful theorem:

Theorem 1.2 (Division Theorem). *Let n and d be integers such that $d > 0$. Then there exists a unique pair of integers q and r such that $n = qd + r$ and $0 \leq r < d$.*

As an example, suppose that $a = 10$ and $b = 2716$. Then the quotient is $q = 271$ and the remainder is $r = 6$, since $2716 = 271 \cdot 10 + 6$.

The remainder r in the Division Theorem is denoted n ***rem*** d. In other words, $n \text{ rem } d$ is the remainder when n is divided by d. For example, $32 \text{ rem } 5$ is the remainder when 32 is divided by 5, which is 2. Similarly, $-11 \text{ rem } 7 = 3$, since $-11 = (-2) \cdot 7 + 3$. There is a remainder operator built into many programming languages. For example, the expression "32 % 5" evaluates to 2 in Java, C, and C++. However, all these languages treat negative numbers strangely.

There are a couple of naming problems related to the Division Theorem. First, the theorem is often called the "Division Algorithm", even though it is not an algorithm in the modern sense. Second, *some* people use the notation "mod" (which is short for "modulo") instead of "rem". This is unfortunate, because "mod" has been used by mathematicians for centuries in a confusingly similar context, which we'll come to shortly. So we'll stick to rem here.

Famous Problems in Number Theory

Fermat's Last Theorem Do there exist positive integers x, y, and z such that

$$x^n + y^n = z^n$$

for some integer $n > 2$? In a book he was reading around 1630, Fermat claimed to have a proof, but not enough space in the margin to write it down. Wiles finally gave a proof of the theorem in 1994, after seven years of working in secrecy and isolation in his attic. His proof did not fit in any margin.

Goldbach Conjecture Is every even integer greater than or equal to 4 the sum of two primes? For example, $4 = 2 + 2$, $6 = 3 + 3$, $8 = 3 + 5$, etc. The conjecture holds for all numbers up to 10^{16}. 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.

Twin Prime Conjecture Are there 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!

Primality Testing Is there an efficient way to determine whether n is prime? An amazingly simple, yet efficient method was finally discovered in 2002 by Agrawal, Kayal, and Saxena. Their paper began with a quote from Gauss emphasizing the importance and antiquity of the problem even in his time— two centuries ago.

Factoring Given the product of two large primes $n = pq$, is there an efficient way to recover the primes p and q? The best known algorithm is the "number field sieve", which runs in time proportional to:

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

This is infeasible when n has a couple hundred digits or more.

We're not going to prove the Division Theorem, but there is an important feature that you should notice. The theorem asserts that the quotient q and remainder r *exist* and also that these values are *unique*. Thus, the Division Theorem is one example of an "existence and uniqueness" theorem; there are many others. Not surprisingly, the proof of such a theorem always has two parts:

- A proof that something exists, such as the quotient q and remainder r.
- A proof that nothing else fits the bill; that is, there is no other quotient q' and remainder r'.

2 Die Hard

Simon: On the fountain, there should be 2 jugs, do you see them? A 5-gallon and a 3-gallon. Fill one of the jugs with exactly 4 gallons of water and place it on the scale and the timer will stop. You must be precise; one ounce more or less will result in detonation. If you're still alive in 5 minutes, we'll speak.

Bruce: Wait, wait a second. I don't get it. Do you get it?

Samuel: No.

Bruce: Get the jugs. Obviously, we can't fill the 3-gallon jug with 4 gallons of water.

Samuel: Obviously.

Bruce: All right. I know, here we go. We fill the 3-gallon jug exactly to the top, right?

Samuel: Uh-huh.

Bruce: Okay, now we pour this 3 gallons into the 5-gallon jug, giving us exactly 3 gallons in the 5-gallon jug, right?

Samuel: Right, then what?

Bruce: All right. We take the 3-gallon jug and fill it a third of the way...

Samuel: No! He said, "Be precise." Exactly 4 gallons.

Bruce: Sh - -. Every cop within 50 miles is running his a - - off and I'm out here playing kids games in the park.

Samuel: Hey, you want to focus on the problem at hand?

This is from the movie *Die Hard 3: With a Vengeance*. Samuel L. Jackson and Bruce Willis have to disarm a bomb planted by the diabolical Simon Gruber. Fortunately, they find a solution in the nick of time. (No doubt reading the script helped.) On the surface, *Die Hard 3* is just a B-grade action movie; however, it seems that the inner message of the film is that everyone should learn at least a little number theory.

Unfortunately, Hollywood never lets go of a gimmick. They're planning to keep the *Die Hard* series going with:

Die Hard 4: Die Hardest Bruce goes on vacation and— shockingly— happens into a terrorist plot. To save the day, he must make 3 gallons using 21 and 26 gallon jugs.

Die Hard 5: Die of Old Age Bruce must save his assisted living facility from a criminal mastermind by forming 2 gallons with 899 and 1147 gallon jugs.

Die Hard 6: Die Once and For All Bruce has to make 4 gallons using 3 and 6-gallon jugs.

It would be nice if we could solve all these silly water jug questions at once. In particular, how can one form g gallons using jugs with capacities a and b?

That's where number theory comes in handy.

2.1 Finding an Invariant Property

Suppose that we have water jugs with capacities a and b. Let's carry out a few arbitrary operations and see what happens. The state of the system at each step is described below with a pair of numbers (x, y), where x is the amount of water in the jug with capacity a and y is the amount in the jug with capacity b.

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

Of course, we're making some assumptions about the relative capacities of the two jugs here. But another point leaps out: at every step, the amount of water in each jug is of the form

$$s \cdot a + t \cdot b \tag{1}$$

for some integers s and t. This sounds like an assertion that we might be able to prove by induction!

An expression of the form (1) is called an ***integer linear combination*** of a and b, but in these notes we'll just call it a ***linear combination***, since we're only talking integers.

In class, we are going to prove the following lemma:

Lemma 2.1. *Suppose that we have water jugs with capacities a and b. Then the amount of water in each jug is always a linear combination of a and b.*

This theorem has an important corollary, which we will also prove in class.

Corollary 2.2. *Bruce dies.*

Lemma 2.1 isn't very satisfying. We've just managed to recast a pretty understandable question about water jugs into a complicated question about linear combinations. This might not seem like progress. Fortunately, linear combinations are closely related to something more familiar and that will help us solve the water jug problem.

3 The Greatest Common Divisor

The ***greatest common divisor*** of a and b is exactly what you'd guess: the largest number that is a divisor of both a and b. It is denoted $\gcd(a, b)$. For example, $\gcd(18, 24) = 6$.

Probably some junior high math teacher made you compute greatest common divisors for no apparent reason until you were blue in the face. But, amazingly, the greatest common divisor actually turns out to be quite useful for reasoning about the integers. Specifically, the quantity $\gcd(a, b)$ is a valuable piece of information about the relationship between the numbers a and b. So we'll make arguments about greatest common divisors all the time.

3.1 Linear Combinations and the GCD

The theorem below relates the greatest common divisor to linear combinations. This theorem is *very* useful; take the time to understand it and then remember it!

Theorem 3.1. *The greatest common divisor of a and b is equal to the smallest positive linear combination of a and b.*

For example, the greatest common divisor of 52 and 44 is 4. And, sure enough, 4 is a linear combination of 52 and 44:

$$6 \cdot 52 + (-7) \cdot 44 = 4$$

Furthermore, no linear combination of 52 and 44 is equal to a smaller positive integer.

Proof. Let m be the smallest positive linear combination of a and b. We'll prove that $m = \gcd(a, b)$ by showing both $\gcd(a, b) \leq m$ and $m \leq \gcd(a, b)$.

First, we show that $\gcd(a, b) \leq m$. By the definition of common divisor, $\gcd(a, b) \mid a$ and $\gcd(a, b) \mid b$. Therefore, for every pair of integers s and t:

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

Thus, in particular, $\gcd(a, b)$ divides m, and so $\gcd(a, b) \leq m$.

Now, we show that $m \leq \gcd(a, b)$. We do this by showing that $m \mid a$. A symmetric argument shows that $m \mid b$, which means that m is a common divisor of a and b. Thus, m must be less than or equal to the *greatest* common divisor of a and b.

All that remains is to show that $m \mid a$. By the Division Algorithm, there exists a quotient q and remainder r such that:

$$a = q \cdot m + r \qquad \text{(where } 0 \leq r < m\text{)}$$

Recall that $m = sa + tb$ for some integers s and t. Subtituting in for m and rearranging terms gives:

$$\begin{aligned} a &= q \cdot (sa + tb) + r \\ r &= (1 - qs)a + (-qt)b \end{aligned}$$

We've just expressed r as a linear combination of a and b. However, m is the *smallest* positive linear combination and $0 \leq r < m$. The only possibility is that the remainder r is not positive; that is, $r = 0$. This implies $m \mid a$. □

The proof notes that every linear combination of a and b is a multiple of $\gcd(a, b)$. Conversely, since $\gcd(a, b)$ is a linear combination of a and b, every multiple of $\gcd(a, b)$ is as well. This establishes a corollary:

Corollary 3.2. *Every linear combination of a and b is a multiple of $\gcd(a, b)$ and vice versa.*

Now we can restate the water jugs lemma in terms of the greatest common divisor:

Corollary 3.3. *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 $\gcd(3, 6) = 3$.

3.2 Properties of the Greatest Common Divisor

We claimed that greatest common divisors are powerful tools for reasoning about the integers. So we'll often make use of some basic $\gcd$ facts:

Lemma 3.4. *The following statements about the greatest common divisor hold:*

1. *Every common divisor of a and b divides $\gcd(a, b)$.*
2. $\gcd(ka, kb) = k \cdot \gcd(a, b)$ *for all $k > 0$.*
3. *If $\gcd(a, b) = 1$ and $\gcd(a, c) = 1$, then $\gcd(a, bc) = 1$.*
4. *If $a \mid bc$ and $\gcd(a, b) = 1$, then $a \mid c$.*
5. $\gcd(a, b) = \gcd(b, a \text{ rem } b)$.

Here's the trick to proving these statements: translate the $\gcd$ world to the linear combination world using Theorem 3.1, argue about linear combinations, and then translate back using Theorem 3.1 again.

Proof. We prove only parts (3) and (4).

Proof of (3): The assumptions together with Theorem 3.1 imply that there exist integers s, t, u, and v such that:

$$sa + tb = 1$$
$$ua + vc = 1$$

Multiplying these two equations gives:

$$(sa + tb)(ua + vc) = 1$$

The left side can be rewritten as $a \cdot (asu + btu + csv) + b \cdot c(tv)$. This is a linear combination of a and bc that is equal to 1, so $\gcd(a, bc) = 1$ by Theorem 3.1.

Proof of (4): Theorem 3.1 says that $\gcd(ac, bc)$ is equal to a linear combination of ac and bc. Now $a \mid ac$ trivially and $a \mid bc$ by assumption. Therefore, a divides *every* linear combination of ac and bc. In particular, a divides $\gcd(ac, bc) = c \cdot \gcd(a, b) = c$. The first equality uses part (2) of this lemma, and the second uses the assumption that $\gcd(a, b) = 1$. □

Part (5) of the lemma 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 part(5):

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

This is called ***Euclid's algorithm***. The last equation might look wrong, but 31 is a divisor of both 31 and 0 since every integer divides 0.

This calculation, together with Corollary 3.3, implies that there is no way to measure out 2 gallons of water using jugs with capacities 1247 and 899; we can only obtain multiples of 31 gallons. This is good news– Bruce won't even survive Die Hard 5!

Let's see if Bruce can possibly make 3 gallons using 21 and 26-gallon jugs. First, we compute the greatest common divisor of 21 and 26 using Euclid's algorithm:

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

Now 3 is a multiple of 1, so we can't *rule out* the possibility that Bruce can form 3 gallons. On the other hand, we don't know he *can* do it either.

3.3 One Solution for All Water Jug Problems

Can Bruce form 3 gallons using 21 and 26-gallon jugs? This question is not so easy to answer without some number theory.

Corollary 3.2 says that 3 can be written as a linear combination of 21 and 26, since 3 is a multiple of $\gcd(21, 26) = 1$. In other words, there exist integers s and t such that:

$$3 = s \cdot 21 + t \cdot 26$$

We don't know what the coefficients s and t are, but we do know that they exist.

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

$$3 = s' \cdot 21 + t' \cdot 26$$

where the coefficient s' is positive. The trick is to notice that if we increase s by 26 in the original equation 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. Notice that t' must be negative; otherwise, this expression would be much greater than 3.

Now here's how to form 3 gallons using jugs with capacities 21 and 26:

- Repeat s' times:
  - Fill the 21-gallon jug.
  - Pour all the water in the 21-gallon jug into the 26-gallon jug. Whenever the 26-gallon jug becomes full, empty it out.

At the end of this process, there must be exactly 3 gallons in the 26-gallon jug! Here's why: we've taken $s' \cdot 21$ gallons of water from the fountain, we've poured out some multiple of 26 gallons, and in the end the 26-gallon jug holds somewhere between 0 and 26 gallons. Furthermore, we know:

$$s' \cdot 21 + t' \cdot 26 = 3$$

Thus, we must have emptied the 26-gallon jug exactly $-t'$ times; if we had emptied it fewer times, then there would be more than 26 gallons left. And we did not withdraw enough water from the fountain to empty the 26-gallon jug more than $-t'$ times. Thus, by the equation above, there must be 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 that approach gives:

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

The same approach works regardless of the jug capacities and even regardless the amount we're trying to produce! Simply proceed as follows:

- Repeat until the desired amount of water is obtained:
  - Fill the smaller jug.
  - Pour all the water in the smaller jug into the larger jug. Whenever the larger jug becomes full, empty it out.

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

3.4 The Pulverizer

We saw that no matter which pair of integers a and b we are given, there is always a pair of integer coefficients s and t such that

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

Furthermore, in the previous subsection, we gave a roundabout method of finding such s and t. However, that method is not very efficient. Here is a much better way:

This job is best tackled by a mathematical tool that dates 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", but that's lame. We're sticking with "Pulverizer".

Euclid's algorithm for finding the GCD of two numbers relies on repeated application of the equation:

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

For example, we can compute the GCD of 259 and 70 as follows:

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

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	$(x \text{ rem } y)$	$=$	$x - q \cdot y$
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 $x \text{ rem } y$, which can be written in the form $x - q \cdot y$. (Remember that the Division Algorithm says $x = q \cdot y + r$, where r is the remainder. We get $r = x - q \cdot y$ by rearranging terms.) Then we replaced x and y in this equation with 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 that was equal to the remainder as desired. The final solution is boxed.

4 The Fundamental Theorem of Arithmetic

We now have almost enough tools to prove something that you probably already know.

Theorem (Fundamental Theorem of Arithmetic). *Every positive integer n can be written in a unique way as a product of primes:*

$$n = p_1 \cdot p_2 \cdots p_j \qquad (p_1 \leq p_2 \leq \ldots \leq p_j)$$

Notice that the theorem would be false if 1 were considered a prime; for example, 15 could be written as $3 \cdot 5$ or $1 \cdot 3 \cdot 5$ or $1^2 \cdot 3 \cdot 5$. Also, we're relying on a standard convention: the product of an empty set of numbers is defined to be 1, much as the sum of an empty set of numbers is defined to be 0. Without this convention, the theorem would be false for $n = 1$.

There is a certain wonder in the Fundamental Theorem, even if you've known it since the crib. Primes show up erratically in the sequence of integers. In fact, their distribution seems almost random:

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

Basic questions about this sequence have stumped humanity for centuries. And yet we know that every natural number can be built up from primes in *exactly one way*. These quirky numbers are the building blocks for the integers. The Fundamental Theorem is not hard to prove, but we'll need a couple preliminary facts.

Lemma 4.1. *If p is a prime and $p \mid ab$, then $p \mid a$ or $p \mid b$.*

Proof. The greatest common divisor of a and p must be either 1 or p, since these are the only divisors of p. If $\gcd(a, p) = p$, then the claim holds, because a is a multiple of p. Otherwise, $\gcd(a, p) = 1$ and so $p \mid b$ by part (4) of Lemma 3.4. □

The Prime Number Theorem

Let $\pi(x)$ denote the number of primes less than or equal to x. For example, $\pi(10) = 4$ because 2, 3, 5, and 7 are the primes less than or equal to 10. Primes are very irregularly distributed, so the growth of π is similarly erratic. However, the Prime Number Theorem gives an approximate answer:

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

Thus, primes gradually taper off. As a rule of thumb, about 1 integer out of every $\ln x$ in the vicinity of x 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" mathematicans who had the misfortune of being contemporaries of Gauss.)

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:

$$\begin{aligned} e =& 2.71828182845904523536028747135266249775724709369995957496 6\\ & 96762772407663035354759457138217852516642\underline{7427466391}9320030 \\ & 599218174135966290435729003342952605956307381323286279434\ldots \end{aligned}$$

A routine induction argument extends this statement to the fact we assumed last time:

Lemma 4.2. *Let p be a prime. If $p \mid a_1 a_2 \ldots a_n$, then p divides some a_i.*

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

Theorem 4.3 (Fundamental Theorem of Arithmetic). *Every positive integer n can be written in a unique way as a product of primes:*

$$n = p_1 \cdot p_2 \cdots p_j \qquad (p_1 \leq p_2 \leq \ldots \leq p_j)$$

Proof. We must prove two things: (1) every positive integer can be expressed as a product of primes, and (2) this expression is unique.

First, we use strong induction to prove that every positive integer n is a product of primes. As a base case, $n = 1$ is the product of the empty set of primes. For the inductive step, suppose that every $k < n$ is a product of primes. We must show that n is also a product of primes. If n is itself prime, then this is true trivially. Otherwise, $n = ab$ for some $a, b < n$. By the induction assumption, a and b are both products of primes. Therefore, $a \cdot b = n$ is also a product of primes. Thus, the claim is proved by induction.

Second, we use the well-ordering principle to prove that every positive integer can be written as a product of primes in a unique way. 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

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

be two of the (possibly many) ways to write n as a product of primes. Then $p_1 \mid n$ and so $p_1 \mid q_1 q_2 \cdots q_k$. Lemma 4.2 implies that p_1 divides one of the primes q_i. But since q_i is a prime, it must be that $p_1 = q_i$. Deleting p_1 from the first product and q_i from the second, we find that n/p_1 is a positive integer *smaller* than n that can also be written as a product of primes in two distinct ways. But this contradicts the definition of n as the smallest such positive integer. □

5 Alan Turing

The man pictured above 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 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 ingenius the programmer. Turing's paper is all the more remarkable because he wrote it in 1936, a full decade before any 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.

6 Turing's Code

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 forsaw 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 digital cash systems. Every time you buy a book from Amazon, check your grades on WebSIS, or use a PayPal account, you are relying on number theoretic algorithms. 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 possibilities.

6.1 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:

$$\begin{array}{cccccccc} & \text{"v} & \text{i} & \text{c} & \text{t} & \text{o} & \text{r} & \text{y"} \\ \rightarrow & 22 & 09 & 03 & 20 & 15 & 18 & 25 \end{array}$$

Turing's code requires the message to be a prime number, so we may need to pad the result with a few more digits to make a prime. In this case, appending the digits 13 gives the number 2209032015182513, which is prime.

Now 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:

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

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 questions that one might naturally 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 reasonably good primality tests were known even in Turing's time. In 2002, Manindra Agrawal, Neeraj Kayal, and Nitin Saxena announced a primality test that is guaranteed to work on a number n in about $(\log n)^{12}$ steps. This definitively placed primality testing in the class of "easy" computational problems at last. Amazingly, the description of their breakthrough algorithm was only thirteen lines long!

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, though a breakthrough someday is not impossible. 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.

6.2 Breaking Turing's Code

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:

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

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 read recover the secret key and read *every* message!

It is difficult to believe a mathematician as brilliant as Turing could overlook such a glaring problem. One possible explanation is that he had a slightly different system in mind, one based on *modular* arithmetic.

7 Modular Arithmetic

On page 1 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 if $n \mid (a - b)$. This is denoted $a \equiv b \pmod{n}$. For example:

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

Intuitively, the $\equiv$ symbol is sort of like an $=$ sign, and the mod 7 describes the specific sense in which 29 is equal-ish to 15. Thus, even though (mod 7) appears over on the right side, it is in no sense more strongly associated with the 15 than the 29; in fact, it actually defines the meaning of the $\equiv$ sign.

Here's another way to think about congruences: *congruence modulo n 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:

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

Now integers in the same set are all congruent modulo 3. For example, 6 and -3 are both in the first set, and they're congruent because their difference, $6 - (-3) = 9$, is a multiple of 3. Similarly, 11 and 5 are both in the last set, because $11 - 5 = 6$ is a multiple of 3. On the other hand, numbers in different sets are not congruent. For example, 9 is in the first set and 11 in the last set, and they're not congruent because $11 - 9 = 2$ is not a multiple of 3. The upshot is that when arithmetic is done modulo n there are only n really different kinds of number to worry about. In this sense, modular arithmetic is a simplification of ordinary arithmetic and thus is a good reasoning tool.

There are many useful facts about congruences, some of which are listed in the lemma below. The overall theme is that *congruences work a lot like equations*, though there are a couple exceptions.

Lemma 7.1 (Facts About Congruences). *The following hold for $n \geq 1$:*

1. $a \equiv a \pmod{n}$
2. $a \equiv b \pmod{n}$ *implies* $b \equiv a \pmod{n}$
3. $a \equiv b \pmod{n}$ *and* $b \equiv c \pmod{n}$ *implies* $a \equiv c \pmod{n}$
4. $a \equiv b \pmod{n}$ *implies* $a + c \equiv b + c \pmod{n}$
5. $a \equiv b \pmod{n}$ *implies* $ac \equiv bc \pmod{n}$
6. $a \equiv b \pmod{n}$ *and* $c \equiv d \pmod{n}$ *imply* $a + c \equiv b + d \pmod{n}$
7. $a \equiv b \pmod{n}$ *and* $c \equiv d \pmod{n}$ *imply* $ac \equiv bd \pmod{n}$

Proof. We prove only parts 1 and 7; the other parts are proved similarly.

(part 1) Every integer divides 0, so $n \mid (a - a)$, which means $a \equiv a \pmod{n}$.

(part 7) The assumption $a \equiv b \pmod{n}$ implies that $ac \equiv bc \pmod{n}$ by part 5. Similarly, the assumption $c \equiv d \pmod{n}$ implies $bc \equiv bd \pmod{n}$. Therefore, $ac \equiv bd \pmod{n}$ by part 3. □

There is a close connection between modular arithmetic and the remainder operation, which we looked at last time. To clarify this link, let's reconsider the partition of the integers defined by congruence modulo 3:

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

Notice that two numbers are in the same set if and only if they leave the same remainder when divided by 3. The numbers in the first set all leave a remainder of 0 when divided by 3, the numbers in the second set leave a remainder of 1, and the numbers in the third leave a remainder of 2. Furthermore, notice that each number is in the same set as its own remainder. For example, 11 and $11 \text{ rem } 3 = 2$ are both in the same set. Let's bundle all this happy goodness into a lemma.

Lemma 7.2 (Congruences and Remainders). *The following assertions hold:*

1. $a \equiv (a \text{ rem } n) \pmod{n}$
2. $a \equiv b \pmod{n}$ *if and only if* $(a \text{ rem } n) = (b \text{ rem } n)$

Proof. (of part 2) By the division algorithm, there exist unique pairs of integers q_1, r_1 and q_2, r_2 such that:

$$\begin{aligned} a &= q_1 n + r_1 && \text{(where } 0 \leq r_1 < n) \\ b &= q_2 n + r_2 && \text{(where } 0 \leq r_2 < n) \end{aligned}$$

In these terms, $(a \text{ rem } n) = r_1$ and $(b \text{ rem } n) = r_2$. Subtracting the second equation from the first gives:

$$a - b = (q_1 - q_2)n + (r_1 - r_2) \qquad \text{(where } -n < r_1 - r_2 < n)$$

Now $a \equiv b \pmod{n}$ if and only if n divides the left side. 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 $r_1 = r_2$, which is equivalent to $(a \text{ rem } n) = (b \text{ rem } n)$. □

8 Turing's Code (Version 2.0)

In 1940 France had fallen before Hitler's army, and Britain alone stood 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-boat" 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.

But a critical reason behind Germany's loss was made public only in 1974: the British had broken Germany's naval code, Enigma. Through much of the war, the Allies were able to route convoys around German submarines by listening into German communications. The British government didn't explain *how* Enigma was broken until 1996. When the analysis was finally released (by the US), the author was none other than Alan Turing. In 1939 he had joined the secret British codebreaking effort at Bletchley Park. There, he played a central role in cracking the German's Enigma code and thus in preventing Britain from falling into Hitler's hands.

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.

Let's 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* arithemtic. Maybe this is what Turing meant:

Beforehand The sender and receiver agree on a large prime p, which may be made public. (This will be the modulus for all our arithemtic.) They also agree on a secret key $k \in \{1, 2, \ldots, p-1\}$.

Encryption The message m can be any integer in the set $\{0, 1, 2, \ldots, p-1\}$; in particular, the message is no longer required to be a prime. The sender encrypts the message m to produce m^* by computing:

$$m^* = mk \text{ rem } p \tag{$*$}$$

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 p. So dividing m^* by k might not even give us an integer!

This decoding difficulty can be overcome with a better understanding of arithmetic modulo a prime.

8.1 Multiplicative Inverses

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

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

Generally, multiplicative inverses exist over the real numbers. For example, the multiplicative inverse of 3 is $1/3$ since:

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

The sole exception is that 0 does not have an inverse.

On the other hand, inverses generally do not exist over the integers. For example, 7 can not be multiplied by another integer to give 1.

Surprisingly, multiplicative inverses do exist when we're working *modulo a prime number*. For example, if we're working modulo 5, then 3 is a multiplicative inverse of 7, since:

$$7 \cdot 3 \equiv 1 \pmod 5$$

(All numbers congruent to 3 modulo 5 are also multiplicative inverses of 7; for example, $7 \cdot 8 \equiv 1 \pmod 5$ as well.) The only exception is that numbers congruent to 0 modulo 5 (that is, the multiples of 5) do not have inverses, much as 0 does not have an inverse over the real numbers. Let's prove this.

Lemma 8.1. *If p is prime and k is not a multiple of p, then k has a multiplicative inverse.*

Proof. Since p is prime, it has only two divisors: 1 and p. And since k is not a multiple of p, we must have $\gcd(p, k) = 1$. Therefore, there is a linear combination of p and k equal to 1:

$$sp + tk = 1$$

Rearranging terms gives:

$$sp = 1 - tk$$

This implies that $p \mid (1 - tk)$ by the definition of divisibility, and therefore $tk \equiv 1 \pmod p$ by the definition of congruence. Thus, t is a multiplicative inverse of k. □

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 *inverse* of the key:

$$\begin{aligned} m^* \cdot k^{-1} &\equiv (mk \text{ rem } p) \cdot k^{-1} \pmod{p} \\ &\equiv mkk^{-1} \pmod{p} \\ &\equiv m \pmod{p} \end{aligned}$$

This shows that $m^* k^{-1}$ is congruent to the original message m. Since the m was in the range $0, 1, \ldots, p-1$, we can recover it exactly taking a remainder:

$$m = m^* k^{-1} \text{ rem } p$$

So now we can decrypt!

8.2 Cancellation

Another sense in which real number are nice is that one can cancel multiplicative terms. In other words, if we know that $m_1 k = m_2 k$, then can cancel the k's and conclude that $m_1 = m_2$, provided $k \neq 0$. In general, cancellation is *not* valid in modular arithmetic. For example, this congruence is correct:

$$2 \cdot 3 \equiv 4 \cdot 3 \pmod{6}$$

But if we cancel the 3's, we reach a false conclusion:

$$2 \equiv 4 \pmod{6}$$

The fact that multiplicative terms can not be cancelled is the most significant sense in which congruences differ from ordinary equations. However, this difference goes away if we're working modulo a *prime*; then cancellation is valid.

Lemma 8.2. *Suppose p is a prime and k is not a multiple of p. Then*

$$ak \equiv bk \pmod{p} \qquad \textit{implies} \qquad a \equiv b \pmod{p}$$

Proof. Multiply both sides of the congruence by k^{-1}. □

We can use this lemma to get a bit more insight into how Turing's code works. In particular, the encryption operation in Turing's code *permutes the space of messages*. This is stated more precisely in the following corollary.

Corollary 8.3. *Suppose p is a prime and k is not a multiple of p. Then the sequence:*

$$(0 \cdot k) \text{ rem } p, \quad (1 \cdot k) \text{ rem } p, \quad (2 \cdot k) \text{ rem } p, \quad \ldots, \quad ((p-1) \cdot k) \text{ rem } p$$

is a permutation of the sequence:

$$0, \quad 1, \quad 2, \quad \ldots, \quad (p-1)$$

This remains true if the first term is deleted from each sequence.

Proof. The first sequence contains p numbers, which are all in the range 0 to $p-1$ by the definition of remainder. Furthermore, the numbers in the first sequence are all different; by Lemma 8.2, $ik \equiv jk \pmod{p}$ if and only if $i \equiv j \pmod{p}$, and no two numbers in the range 0, 1, . . . , p - 1 are congruent modulo p. Thus, the first sequence must contain *all* of the numbers from 0 to $p-1$ in some order. The claim remains true if the first terms are deleted, because both sequences begin with 0. □

For example, suppose $p = 5$ and $k = 3$. Then the sequence:

$$\underbrace{(0 \cdot 3) \text{ rem } 5}_{=0}, \quad \underbrace{(1 \cdot 3) \text{ rem } 5}_{=3}, \quad \underbrace{(2 \cdot 3) \text{ rem } 5}_{=1}, \quad \underbrace{(3 \cdot 3) \text{ rem } 5}_{=4}, \quad \underbrace{(4 \cdot 3) \text{ rem } 5}_{=2}$$

is a permutation of 0, 1, 2, 3, 4 and the last four terms are a permutation of 1, 2, 3, 4. As long as the Nazis don't know the secret key k, they don't know how the message space is permuted by the process of encryption and thus can't read encoded messages.

8.3 Fermat's Theorem

A remaining challenge in using Turing's code is that decryption requires the inverse of the secret key k. But how can we find an inverse of k? One approach is to rely on Fermat's Theorem, which is much easier than his famous Last Theorem— and more useful.

Theorem 8.4 (Fermat's Theorem). *Suppose p is a prime and k is not a multiple of p. Then:*

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

Proof. We reason as follows:

$$\begin{aligned} 1 \cdot 2 \cdot 3 \cdots (p-1) &\equiv (k \text{ rem } p) \cdot (2k \text{ rem } p) \cdot (3k \text{ rem } p) \cdots ((p-1)k \text{ rem } p) \pmod{p} \\ &\equiv k \cdot 2k \cdot 3k \cdots (p-1)k \pmod{p} \\ &\equiv (p-1)! \cdot k^{p-1} \pmod{p} \end{aligned}$$

The expressions on the first line are actually equal, by Corollary 8.3, so they are certainly congruent modulo p. The second step uses part 1 of Lemma 7.2. In the third step, we rearrange terms in the product.

Now $(p-1)!$ can not be a multiple of p, because the prime factorizations of $1, 2, \ldots, (p-1)$ contain only primes smaller than p. Therefore, we can cancel $(p-1)!$ from the first expression and the last by Lemma 8.2, which proves the claim. □

Here is how we can find inverses using Fermat's Theorem. Suppose p is a prime and k is not a multiple of p. Then, by Fermat's Theorem, we know that:

$$k^{p-2} \cdot k \equiv 1 \pmod{p}$$

Therefore, k^{p-2} must be a multiplicative inverse of k. For example, suppose that we want the multiplicative inverse of 6 modulo 17. Then we need to compute $6^{15} \text{ rem } 17$, which we can do by successive squaring. All the congruences below hold modulo 17.

$$\begin{aligned}
6^2 &\equiv 36 \equiv 2 \\
6^4 &\equiv (6^2)^2 \equiv 2^2 \equiv 4 \\
6^8 &\equiv (6^4)^2 \equiv 4^2 \equiv 16 \\
6^{15} &\equiv 6^8 \cdot 6^4 \cdot 6^2 \cdot 6 \equiv 16 \cdot 4 \cdot 2 \cdot 6 \equiv 3
\end{aligned}$$

Therefore, $6^{15} \text{ rem } 17 = 3$. Sure enough, 3 is the multiplicative inverse of 6 modulo 17, since:

$$3 \cdot 6 \equiv 1 \pmod{17}$$

In general, if we were working modulo a prime p, finding a multiplicative inverse by trying every value between 1 and $p - 1$ would require about p operations. However, the approach above requires only about $\log p$ operations, which is far better when p is large.

8.4 Breaking Turing's Code— Again

German weather reports were *not* encrypted 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 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 m and m^* where:

$$m^* \equiv mk \pmod{p}$$

Now they can compute:

$$\begin{aligned}
m^{p-2} \cdot m^* &\equiv m^{p-2} \cdot (mk \text{ rem } p) \pmod{p} && \text{(def. of } m^*) \\
&\equiv m^{p-2} \cdot mk \pmod{p} && \text{(part 2 of Lemma 7.2)} \\
&\equiv m^{p-1} \cdot k \pmod{p} && \text{(simplification)} \\
&\equiv k \pmod{p} && \text{(Fermat's Theorem)}
\end{aligned}$$

Now the Nazis have the secret key k and can decrypt any message!

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

9 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, at that time, homosexuality was a crime in Britain, punishable by up to two years in prison. Turing was sentenced to a humiliating 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 devoutly religious 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.

10 Arithmetic with an Arbitrary Modulus

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 at MIT, Ronald Rivest, Adi Shamir, and Leonard Adleman proposed a highly secure cryptosystem (called **RSA**) based on number theory. Despite decades of attack, no significant weakness has been found. 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 ***secret key***, which she guards closely, and a ***public key***, which she distributes as widely as possible. To send her a message, one encrypts using her widely-distributed public key. Then she decrypts the message using her 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 scheme may have, but rather modulo the product of *two* large primes. Thus, we'll need to know a bit about how arithmetic works modulo a composite number in order to understand RSA. Arithmetic modulo an arbitrary positive integer is really only a little more painful than working modulo a prime, in the same sense that a doctor says "This is only going to hurt a little" before he jams a big needle in your arm.

10.1 Relative Primality and Phi

First, we need a new definition. Integers a and b are ***relatively prime*** if $\gcd(a, b) = 1$. For example, 8 and 15 are relatively prime, since $\gcd(8, 15) = 1$. Note that every integer is relatively prime to a genuine prime number p, except for multiples of p.

We'll also need a certain function that is defined using relative primality. Let n be a positive integer. Then $\phi(n)$ denotes the number of integers in $\{1, 2, \ldots, n-1\}$ that are relatively prime to

The Riemann Hypothesis

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 Berhard Riemann in 1859 and is now one of the most famous unsolved problem in mathematics. The formula for the sum of an infinite geometric series says:

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

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:

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

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:

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

Researchers continue to work intensely to settle this conjecture, as they have for over a century. A proof would immediately imply, among other things, a strong form of the Prime Number Theorem— and earn the prover a $1 million prize! (We're not sure what the cash would be for a counter-example, but the discoverer would be wildly applauded by mathematics everywhere.)

n. For example, $\phi(7) = 6$, since 1, 2, 3, 4, 5, and 6 are all relatively prime to 7. Similarly, $\phi(12) = 4$, since only 1, 5, 7, and 11 are relatively prime to 12. If you know the prime factorization of n, then computing $\phi(n)$ is a piece of cake, thanks to the following theorem.

Theorem 10.1. *The function ϕ obeys the following relationships:*

1. *If a and b are relatively prime, then $\phi(ab) = \phi(a)\phi(b)$.*
2. *If p is a prime, then $\phi(p^k) = p^k - p^{k-1}$ for $k \geq 1$.*

This is not a terribly difficult theorem, but we'll hold off on the proof for a few weeks. In the meanwhile, here's an example of how we might use Theorem 10.1 to compute $\phi(300)$:

$$\begin{aligned}\phi(300) &= \phi(2^2 \cdot 3 \cdot 5^2)\\ &= \phi(2^2) \cdot \phi(3) \cdot \phi(5^2)\\ &= (2^2 - 2^1)(3^1 - 3^0)(5^2 - 5^1)\\ &= 80\end{aligned}$$

We factor 300 in the first step, use part (1) of Theorem 10.1 twice in the second step, use part (2) in the third step, and then simplify.

10.2 Generalizing to an Arbitrary Modulus

Let's generalize what we know about arithmetic modulo a prime. Now, instead of working modulo a prime p, we'll work modulo an arbitrary positive integer n. The basic theme is that arithmetic modulo n may be complicated, but the integers *relatively prime* to n remain fairly well-behaved. For example, if k is relatively prime to n, then k has a multiplicative inverse modulo n:

Lemma 10.2. *Let n be a positive integer. If k is relatively prime to n, then there exists an integer k^{-1} such that:*

$$k \cdot k^{-1} \equiv 1 \pmod{n}$$

As a consequence of this lemma, we can cancel a multiplicative term from both sides of a congruence if that term is relatively prime to the modulus:

Corollary 10.3. *Suppose n is a positive integer and k is relatively prime to n. If*

$$ak \equiv bk \pmod{n}$$

then

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

This holds because we can multiply both sides of the first congruence by k^{-1} and simplify to obtain the second.

10.3 Euler's Theorem

RSA essentially relies on Euler's Theorem, a generalization of Fermat's Theorem to an arbitrary modulus. The proof is much like the proof of Fermat's Theorem, except that we focus on integers relatively prime to the modulus. Let's start with a lemma.

Lemma 10.4. *Suppose n is a positive integer and k is relatively prime to n. Let $k_1, \ldots, k_r$ denote all the integers relatively prime to n in the range $0 \leq k_i < n$. Then the sequence:*

$$(k_1 \cdot k) \text{ rem } n, \quad (k_2 \cdot k) \text{ rem } n, \quad (k_3 \cdot k) \text{ rem } n, \quad \ldots, \quad (k_r \cdot k) \text{ rem } n$$

is a permutation of the sequence:

$$k_1, \quad k_2, \quad \ldots, \quad k_r$$

Proof. We will show that the numbers in the first sequence are all distinct and all appear in the second sequence. Since the two sequences have the same length, the first must be a permutation of the second.

First, we show that the numbers in the first sequence are all distinct. Suppose that $k_i k \text{ rem } n = k_j k \text{ rem } n$. This is equivalent to $k_i k \equiv k_j k \pmod{n}$, which implies $k_i \equiv k_j \pmod{n}$ by Corollary 10.3. This, in turn, means that $k_i = k_j$ since both are between 1 and $n-1$. Thus, a term in the first sequence is only equal to itself.

Next, we show that each number in the first sequence appears in the second. By assumption, $\gcd(k_i, n) = 1$ and $\gcd(k, n) = 1$, which means that

$$\gcd(k_i k, n) = \gcd(k_i k \text{ rem } n, n) = 1$$

by part (3) of Lemma 3.4. Therefore, $k_i k \text{ rem } n$ is relatively prime to n and is in the range from 0 to $n-1$ by the definition of rem. The second sequence is defined to consist of all such integers. □

We can now prove Euler's Theorem:

Theorem 10.5 (Euler's Theorem). *Suppose n is a positive integer and k is relatively prime to n. Then:*

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

Proof. Let $k_1, \ldots, k_r$ denote all integers relatively prime to n such that $0 \leq k_i < n$. Then $r = \phi(n)$, by the definition of the function ϕ. Now we can reason as follows:

$$\begin{aligned}
& k_1 \cdot k_2 \cdot k_3 \cdots k_r \\
&\qquad \equiv \quad \big(k_1 \cdot k \text{ rem } n\big) \cdot \big(k_2 \cdot k \text{ rem } n\big) \cdot \big(k_3 \cdot k \text{ rem } n\big) \cdots \big(k_r \cdot k \text{ rem } n\big) \pmod{n} \\
&\qquad \equiv \quad (k_1 \cdot k) \cdot (k_2 \cdot k) \cdot (k_3 \cdot k) \cdots (k_r \cdot k) \pmod{n} \\
&\qquad \equiv \quad (k_1 \cdot k_2 \cdot k_3 \cdots k_r) \cdot k^r \pmod{p}
\end{aligned}$$

The first two expressions are actually equal by Lemma 10.4; therefore, they are certainly congruent modulo n. The second step uses a property of mod and rem that we proved earlier. In the third step, we rearrange terms in the product.

Part (3) of Lemma 3.4 implies that $k_1 \cdot k_2 \cdot k_3 \cdots k_r$ is prime relative to n. Therefore, we can cancel this product from the first expression and the last by Corollary 10.3. This proves the claim. □

We can find multiplicative inverses using Euler's theorem as we did with Fermat's theorem: if k is relatively prime to n, then $k^{\phi(n)-1}$ is a multiplicative inverse of k modulo n. However, this approach requires computing $\phi(n)$. Our best method for doing so requires factoring n, which can be quite difficult in general. Fortunately, when we know how to factor n, we can use Theorem 10.1 to compute $\phi(n)$ efficiently!

10.4 RSA

Finally, we are ready to see how the RSA public key encryption scheme works:

RSA Public Key Encryption

Beforehand The receiver creates a public key and a secret key as follows.

1. Generate two distinct primes, p and q.
2. Let $n = pq$.
3. Select an integer e such that $\gcd(e, (p-1)(q-1)) = 1$.
   The *public key* is the pair (e, n). This should be distributed widely.
4. Compute d such that $de \equiv 1 \pmod{(p-1)(q-1)}$.
   The *secret key* is the pair (d, n). This should be kept hidden!

Encoding The sender encrypts message m to produce m' using the public key:

$$m' = m^e \text{ rem } n.$$

Decoding The receiver decrypts message m' back to message m using the secret key:

$$m = (m')^d \text{ rem } n.$$

We'll explain in class why this way of Decoding works!