Part I

Proofs

A: About ~~math~~ CS structures, MATH has pfs.

This text is all about ~~proofs~~ methods for constructing and understanding proofs. In fact, we could have titled the book "Proofs, Proofs, and More Proofs." We will begin in Part I with a description of basic proof techniques. We then apply these techniques to establish some very important facts about numbers; facts that form the underpinnings of the world's most widely used cryptographic systems. In Chapter 4

~~Mathematical~~ Proofs

A proof is a method of establishing truth. ~~What~~ constitutes a proof differs among fields.

Simply put, a proof is a method of establishing truth.

For example, in the judicial system, legal truth is decided by a jury based on allowable evidence presented at the trial.

Like beauty, "truth" sometimes depends on the eye of the beholder, however, and so it should not be surprising that what ~~is~~ In the business world,

Authoritative truth is specified by a trusted person or organization, or maybe just your boss.

Scientific truth[1] is confirmed by experiment. In statistics, In fields such as physics or biology,

Probable truth is established by statistical analysis of sample data.

Fold into paragraph above.

Philosophical proof involves careful exposition and persuasion typically based on a series of small, plausible arguments. The best example begins with "Cogito ergo sum," a Latin sentence that translates as "I think, therefore I

[1] Actually, only scientific *falsehood* can be demonstrated by an experiment —when the experiment really fails to behave as predicted. But no amount of experiment can confirm that the *next* experiment won't fail. For this reason, scientists rarely speak of truth, but rather of *theories* that accurately predict past, and anticipated future, experiments.

put in paragraph form not bullet/indent

am." It comes from the beginning of a 17th century essay by the mathematician/philospher, René Descartes, and it is one of the most famous quotes in the world: do a web search on the phrase and you will be flooded with hits. Deducing your existence from the fact that you're thinking about your existence is a pretty cool and persuasive-sounding ~~first axiom~~ idea. However, with just a few more lines of argument in this vein, Descartes goes on to conclude that there is an infinitely beneficent God. Whether or not you believe in a beneficent God, you'll probably agree that any very short proof of God's existence is bound to be far-fetched. So even in masterful hands, this approach is not reliable.

Mathematics ~~also~~ has its own specific notion of "proof."

Definition. A *formal proof* of a *proposition* is a chain of *logical deductions* leading to the proposition from a base set of *axioms*.

The three key ideas in this definition are highlighted: proposition, logical deduction, and axiom. These three ideas are explained ~~in Chapter 1, and Chapter 2~~ in the following chapters, beginning with propositions in chapter 1. We will then ~~describe the two most common templates~~ provide lots of examples of proofs

We hope this is helpful as an explanation, but we don't really want to call it a "proof." The problem is that with something as basic as (1.6), it's hard to see what more elementary axioms are ok to use in proving it. What the explanation above did was translate the logical formula (1.6) into English and then appeal to the meaning, in English, of "for all" and "there exists" as justification. So this wasn't a proof, just an explanation that once you understand what (1.6) means, it becomes obvious.

In contrast to (1.6), the formula

$$\forall y \exists x.\ P(x,y) \text{ IMPLIES } \exists x \forall y.\ P(x,y). \tag{1.9}$$

is *not* valid. We can prove this by describing an interpretation where the hypothesis, $\forall y \exists x.\ P(x,y)$, is true but the conclusion, $\exists x \forall y.\ P(x,y)$, is not true. For example, let the domain be the integers and $P(x,y)$ mean $x > y$. Then the hypothesis would be true because, given a value, n, for y we could choose the value of x to be $n+1$. But under this interpretation the conclusion asserts that there is an integer that is bigger than all integers, which is certainly false. An

interpretation like this which falsifies an assertion is called a *counter model* to the assertion.

1.5.9 Problems

Class Problems

Homework Problems

Location A10
(the satisfiability section goes here.)

1.6 Problems

Albert will supply these and they will be organized according to sections.

describes some basic ways of organizing ~~proofs~~.

A: why

0.0.1 Problems

Class Problems

and even some examples of "~~false proofs~~" (i.e., arguments that ~~look~~ look like ~~a~~ a proof but that contain mis-steps, or deductions that aren't so logical when examined closely).

A: Beg students to expand (later?) complex plausible easily fooled step-by-step. illustrate use of false pfs

Albert - we gave lots of "propositions" that are not mathematical in the later sections, so we should remove "mathematical" from the definition.

Chapter 1

Propositions

Definition. A *proposition* is a mathematical statement that is either true or false.

Being true or false doesn't sound like much of a limitation, but it does exclude statements such as, "Wherefore art thou Romeo?" and "Give me an *A*!".

Being "mathematical" is a more serious restriction. For example, "Albert's wife's name is 'Irene' " is a true statement, and you could prove it by presenting legal documents and the testimony of their children. But it isn't a proposition because it is not a *mathematical* statement. There is no mathematical definition of Al-

move to position A1

bert or Irene, and statements about them are not part of mathematics. Propositions must be about well-defined mathematical objects like numbers, sets, functions, relations, etc., and they must be stated using mathematically precise language. For example, this with a few examples.

Proposition 1.0.1. $2 + 3 = 5$.

This is a true proposition.

Proposition 1.0.2. *The binary representation of every nonnegative integer starts with a 1.*

This is a false proposition. It could be fixed by ruling out the nonnegative integer zero. So the following proposition is true:

Proposition 1.0.3. *The binary representation of every positive integer starts with a 1.*

Unfortunately, it is not always easy to decide if a proposition is true or false, or even what the proposition means. In part, this is because the English language is riddled with ambiguities. For example, here

1.1 Compound Propositions

It is amazing that people manage to cope with all the ambiguities in the English language. Here are some sentences that illustrate the issue:

statements

1. "You may have cake, or you may have ice cream."

2. "If pigs can fly, then you can understand the Chebyshev bound."

3. "If you can solve any problem we come up with, then you get an *A* for the course."

4. "Every American has a dream."

What *precisely* do these sentences mean? Can you have both cake and ice cream or must you choose just one dessert? If the second sentence is true, then is the Chebyshev bound incomprehensible? If you can solve some problems we come up with but not all, then do you get an *A* for the course? And can you still get an *A* even if you can't solve any of the problems? Does the last sentence imply that all Americans have the same dream or might some of them have different dreams?

Some uncertainty is tolerable in normal conversation. But when we need to formulate ideas precisely —as in mathematics and programming —the ambiguities inherent in everyday language can be a real problem. We can't hope to make an exact argument if we're not sure exactly what the statements mean. So before we start into mathematics, we need to investigate the problem of how to talk about mathematics.

To get around the ambiguity of English, mathematicians have devised a special mini-language for talking about logical relationships. This language mostly uses ordinary English words and phrases such as "or", "implies", and "for all". But mathematicians endow these words with definitions more precise than those found in an ordinary dictionary. Without knowing these definitions, you might sometimes get the gist of statements in this language, but you would regularly get misled about what they really meant.

Surprisingly, in the midst of learning the language of logic, we'll come across the most important open problem in computer science —a problem whose solution

could change the world.

1.2 Propositions from Propositions

In English, we can modify, combine, and relate propositions with words such as "not", "and", "or", "implies", and "if-then". For example, we can combine three propositions into one like this:

If all humans are mortal **and** all Greeks are human, **then** all Greeks are mortal.

For the next while, we won't be much concerned with the internals of propositions —whether they involve mathematics or Greek mortality —but rather with how propositions are combined and related. So we'll frequently use variables such as P and Q in place of specific propositions such as "All humans are mortal" and "$2 + 3 = 5$". The understanding is that these variables, like propositions, can take on only the values T (true) and F (false). Such true/false variables are sometimes called *Boolean variables* after their inventor, George —you guessed it —Boole.

these should all update to reflect we are in Section 1.1

caps no quotes

1.1.1

1.2.1 "Not", "And", and "Or"

We can precisely define these special words using *truth tables*. For example, if P denotes an arbitrary proposition, then the truth of the proposition "NOT P" is defined by the following truth table:

P	NOT P
T	F
F	T

The first row of the table indicates that when proposition P is true, the proposition "NOT P" is false. The second line indicates that when P is false, "NOT P" is true. This is probably what you would expect.

In general, a truth table indicates the true/false value of a proposition for each possible setting of the variables. For example, the truth table for the proposition "P AND Q" has four lines, since the two variables can be set in four different ways:

P	Q	P AND Q
T	T	T
T	F	F
F	T	F
F	F	F

According to this table, the proposition "P AND Q" is true only when P and Q are both true. This is probably the way you think about the word "and."

There is a subtlety in the truth table for "P OR Q":

P	Q	P OR Q
T	T	T
T	F	T
F	T	T
F	F	F

The third row of this table says that "P OR Q" is true even if *both* P and Q are true. This isn't always the intended meaning of "or" in everyday speech, but this is the standard definition in mathematical writing. So if a mathematician says, "You may have cake, or you may have ice cream," he means that you *could* have both.

If you want to exclude the possibility of both having and eating, you should use "exclusive-or" (XOR):

P	Q	P XOR Q
T	T	F
T	F	T
F	T	T
F	F	F

1.1.2

1.2.2 "Implies"

caps no quotes

The least intuitive connecting word is "implies." Here is its truth table, with the lines labeled so we can refer to them later.

P	Q	P IMPLIES Q	
T	T	T	(tt)
T	F	F	(tf)
F	T	T	(ft)
F	F	T	(ff)

A nonsensical self-evident

Let's experiment with this definition. For example, is the following proposition true or false?

"If the Riemann Hypothesis is true, then $x^2 \geq 0$ for every real number x."

The Riemann Hypothesis is a famous unresolved ~~open question~~ conjecture in mathematics (i.e., no one knows if it is true or false).

~~Now, we told you before that no one knows whether Goldbach's Conjecture is true or false.~~ But that doesn't prevent you from answering the question! This proposition has the form $P \longrightarrow Q$ where the *hypothesis*, P, is "~~Goldbach's Conjecture~~ the Riemann Hypothesis is true" and the *conclusion*, Q, is "$x^2 \geq 0$ for every real number x". Since the conclusion is definitely true, we're on either line (tt) or line (ft) of the truth table. Either way, the proposition as a whole is *true*!

One of our original examples demonstrates an even stranger side of implica-

tions.

"If pigs can fly, then you can understand the Chebyshev bound."

Don't take this as an insult; we just need to figure out whether this proposition is true or false. Curiously, the answer has *nothing* to do with whether or not you can understand the Chebyshev bound. Pigs cannot fly, so we're on either line (ft) or line (ff) of the truth table. In both cases, the proposition is *true*!

In contrast, here's an example of a false implication:

"If the moon shines white, then the moon is made of white cheddar."

Yes, the moon shines white. But, no, the moon is not made of white cheddar cheese. So we're on line (tf) of the truth table, and the proposition is false.

The truth table for implications can be summarized in words as follows:

An implication is true exactly when the if-part is false or the then-part is true.

This sentence is worth remembering; a large fraction of all mathematical statements are of the if-then form!

1.1.3 IFF

caps / no quotes

1.2.3 "If and Only If"

Mathematicians commonly join propositions in one additional way that doesn't arise in ordinary speech. The proposition "P if and only if Q" asserts that P and Q are logically equivalent; that is, either both are true or both are false.

P	Q	P IFF Q
T	T	T
T	F	F
F	T	F
F	F	T

For example, the

The following if-and-only-if statement is true for every real number x:

$$x^2 - 4 \geq 0 \quad \text{iff} \quad |x| \geq 2$$

For some values of x, *both* inequalities are true. For other values of x, *neither* inequality is true . In every case, however, the proposition as a whole is true.

1.2.4 Problems

Class Problems

Homework Problems

position A2 is here as section 1.1.4

position A3 is here as section 1.1.5

1.2

1.3 Propositional Logic in Computer Programs

Propositions and logical connectives arise all the time in computer programs. For example, consider the following snippet, which could be either C, C++, or Java:

Daniel: we need to put "or" and "and" etc in CAPS to be consistent.

```
if ( x > 0 || (x <= 0 && y > 100) )
    ⋮
(further instructions)
```

CAPS

The symbol `||` denotes "or", and the symbol `&&` denotes "and". The *further instructions* are carried out only if the proposition following the word `if` is true. On closer inspection, this big expression is built from two simpler propositions. Let A be the proposition that `x > 0`, and let B be the proposition that `y > 100`. Then

we can rewrite the condition this way:

$$A \text{ or } ((\text{not } A) \text{ and } B) \tag{1.1}$$

A truth table reveals that this complicated expression is logically equivalent to "A or B."

(1.2)

A	B	A or ((not A) and B)	A or B
T	T	T	T
T	F	T	T
F	T	T	T
F	F	F	F

This means that we can simplify the code snippet without changing the program's behavior:

```
if ( x > 0 || y > 100 )
    ⋮
(further instructions)
```

The equivalence of (1.1) and (1.2) can also be confirmed reasoning by cases:

A is T. Then an expression of the form (A or anything) will have truth value T. Since both expressions are of this form, both have the same truth value in this case, namely, T.

A is F. Then (A or P) will have the same truth value as P for any proposition, P. So (1.2) has the same truth value as B. Similarly, (1.1) has the same truth value as ((not F) and B), which also has the same value as B. So in this case, both expressions will have the same truth value, namely, the value of B.

Rewriting a logical expression involving many variables in the simplest form is both difficult and important. Simplifying expressions in software can slightly increase the speed of your program. But more significantly, chip designers face essentially the same challenge. However, instead of minimizing && and || symbols in a program, their job is to minimize the number of analogous physical devices on a chip. The payoff is potentially enormous: a chip with fewer devices is smaller, consumes less power, has a lower defect rate, and is cheaper to manufacture.

1.3.1 Cryptic Notation

Programming languages use symbols like && and || in place of words like "and" and "not". Mathematicians have devised their own cryptic symbols to represent

these words, which are summarized in the table below.

English	Cryptic Notation
not P	$\neg P$ (alternatively, $\overline{P}$)
P and Q	$P \wedge Q$
P or Q	$P \vee Q$
P implies Q	$P \longrightarrow Q$
if P then Q	$P \longrightarrow Q$
P iff Q	$P \longleftrightarrow Q$

For example, using this notation, "If P and not Q, then R" would be written:

$$(P \wedge \overline{Q}) \longrightarrow R$$

This symbolic language is helpful for writing complicated logical expressions compactly. But words such as "OR" and "IMPLIES," generally serve just as well as the cryptic symbols $\vee$ and $\longrightarrow$, and their meaning is easy to remember. So we'll use the cryptic notation sparingly, and we advise you to do the same.

1.3.2 Logically Equivalent Implications

Do these two sentences say the same thing?

If I am hungry, then I am grumpy.

If I am not grumpy, then I am not hungry.

We can settle the issue by recasting both sentences in terms of propositional logic. Let P be the proposition "I am hungry", and let Q be "I am grumpy". The first sentence says "P implies Q" and the second says "(not Q) implies (not P)". We can compare these two statements in a truth table:

P	Q	P IMPLIES Q	$\overline{Q}$ IMPLIES $\overline{P}$
T	T	T	T
T	F	F	F
F	T	T	T
F	F	T	T

Sure enough, the columns of truth values under these two statements are the same, which precisely means they are equivalent. In general, "(NOT Q) IMPLIES (NOT P)" is called the *contrapositive* of the implication "P IMPLIES Q." And, as we've just shown, the two are just different ways of saying the same thing.

In contrast, the *converse* of "P IMPLIES Q" is the statement "Q IMPLIES P". In terms of our example, the converse is:

If I am grumpy, then I am hungry.

This sounds like a rather different contention, and a truth table confirms this suspicion:

P	Q	P IMPLIES Q	Q IMPLIES P
T	T	T	T
T	F	F	T
F	T	T	F
F	F	T	T

Thus, an implication *is* logically equivalent to its contrapositive but is *not* equivalent to its converse.

One final relationship: an implication and its converse together are equivalent to an iff statement, specifically, to these two statements together. For example,

If I am grumpy, then I am hungry, and
if I am hungry, then I am grumpy.

are equivalent to the single statement:

I am grumpy iff I am hungry.

Once again, we can verify this with a truth table:

P	Q	$(P$ IMPLIES $Q)$	AND	$(Q$ IMPLIES $P)$	Q IFF P
T	T	T	T	T	T
T	F	F	F	T	F
F	T	T	F	F	F
F	F	T	T	T	T

$(P \Rightarrow Q) \wedge (Q \Rightarrow P)$

$Q \Leftrightarrow P$

T F F T

A: segway w/ explanation cryptic it won't fit

Albert: This was a confusing

The underlined operators have the same column of truth values, proving that the corresponding formulas are equivalent.

1.3.3 Problems

Class Problems

Homework Problems

move this to location A10 as the last section of chapter 1

A: 1 Prop 2 Quant 3 SAT

1.4 Satisfiability

1.5

A proposition is **satisfiable** if some setting of the variables makes the proposition true. For example, P AND $\overline{Q}$ is satisfiable because the expression is true when P is true and Q is false. On the other hand, P AND $\overline{P}$ is not satisfiable because the expression as a whole is false for both settings of P. But determining whether or not a more complicated proposition is satisfiable is not so easy. How about this one?

$$(P \text{ OR } Q \text{ OR } R) \text{ AND } (\overline{P} \text{ OR } \overline{Q}) \text{ AND } (\overline{P} \text{ OR } \overline{R}) \text{ AND } (\overline{R} \text{ OR } \overline{Q})$$

The general problem of deciding whether a proposition is satisfiable is called *SAT*. One approach to SAT is to construct a truth table and check whether or not a T ever appears. But this approach is not very efficient; a proposition with n variables has a truth table with 2^n lines, so the effort required to decide about a proposition grows exponentially with the number of variables. For a proposition with just 30 variables, that's already over a billion lines to check!

Is there a more *efficient* solution to SAT? In particular, is there some, presumably very ingenious, procedure that determines in a number of steps that grows *polynomially* —like n^2 or n^{14} —instead of exponentially, whether any given proposition is satifiable or not? No one knows. And an awful lot hangs on the answer. An efficient solution to SAT would immediately imply efficient solutions to many, many other important problems involving packing, scheduling, routing, and circuit verification, among other things. This would be wonderful, but there would also be worldwide chaos. Decrypting coded messages would also become an easy task (for most codes). Online financial transactions would be insecure and secret

communications could be read by everyone.

Recently there has been exciting progress on *sat-solvers* for practical applications like digital circuit verification. These programs find satisfying assignments with amazing efficiency even for formulas with millions of variables. Unfortunately, it's hard to predict which kind of formulas are amenable to sat-solver methods, and for formulas that are NOT satisfiable, sat-solvers generally take exponential time to verify that.

So no one has a good idea how to solve SAT in polynomial time or else to prove that it can't be done —researchers are completely stuck. The problem of determining whether or not SAT has a polynomial time solution is known as the "**P** vs. **NP**" problem. It is the outstanding unanswered question in theoretical computer science. It is also one of the seven Millenium Problems: the Clay Institute will award you $1,000,000 if you solve the **P** vs. **NP** problem.

1.4.1 Problems

Class Problems

1.3

1.5 Predicates and Quantifiers

1.3.1

1.5.1 Some More Propositions

A *prime* is an integer greater than one that is not divisible by any integer greater than 1 besides itself. For example, 2, 3, 5, 7, 11 are prime (but 4, 6, and 9 are composite).

Proposition 1.5.1. *For every nonnegative integer, n, the value of $n^2 + n + 41$ is prime.*

Let's try some numerical experimentation to check this proposition. Let[1]

$$p(n) ::= n^2 + n + 41 \tag{1.3}$$

We begin with $p(0) = 41$ which is prime. $p(1) = 43$ which is prime. $p(2) = 47$, which is prime, $p(3) = 53$ which is prime, ... , $p(20) = 461$ which is prime. Hmmm... It is starting to look like a plausible claim. In fact we can keep checking through $n = 39$

[1] The symbol ::= means "equal by definition." It's always ok to simply write "=" instead of ::=, but reminding the reader that an equality holds by definition can be helpful.

INSERT B1

Most of the examples of propositions that we have considered thus far have been nice in the sense that it has been relatively easy to determine if they are true or false.

→ At worst, there were only a few cases to check in a truth table. Unfortunately, not all propositions are so easy to check. That is because some propositions may involve a large or infinite number of possible cases. For example, consider the following proposition involving prime numbers. (A prime is an integer greater than 1 that is divisible only by itself and 1. For example, 2, 3, 5, 7, and 11 are primes, but 4, 6 and 9 are not. A number greater than 1 that is not prime is said to be composite.)

INSERT B2

It is not immediately clear whether this proposition is true or false. In such circumstances, it is tempting to try to determine its veracity by computing the value of [1] (Footnote)

INSERT B3 ~~goes to section 4.6~~

for several values to n and then checking to see if they are ~~indeed~~ prime. If any of the computed values is not prime, then we will know that the proposition is false. If all the computed values are indeed prime, then we might be tempted to conclude that the proposition is true.

and confirm that $p(39) = 1601$ is prime. The proposition certainly does seem to be true.

But $p(40) = 40^2 + 40 + 41 = 41 \cdot 41$, which is not prime. So it's not true that the expression is prime *for all* nonnegative integers, and thus the proposition is false. The point is that in general you can't check a claim about an infinite set by checking a finite set of its elements, no matter how large the finite set.

By the way, propositions like this about *all* numbers or other things are so common that involve them. For example, that there is a special notation for it. With this notation, Proposition 1.5.1 can also be written as would be

$$\forall n \in \mathbb{N}.\ p(n) \text{ is prime.} \tag{1.4}$$

Here the symbol $\forall$ is read "for all". The symbol $\mathbb{N}$ stands for the set of *nonnegative integers*, namely, 0, 1, 2, 3, ... (ask your instructor for the complete list). The symbol "$\in$" is read as "is a member of," or "belongs to," or simply as "is in". The period after the $\mathbb{N}$ is just a separator between phrases.

Here are two even more extreme examples of propositions that, at first, seem to be true but which turn out to be false. is another example of a

Proposition 1.5.2. *$a^4 + b^4 + c^4 = d^4$ has no solution when a, b, c, d are positive integers.*

INSERT B4 goes to location A7

Although surprising, this example is not as ~~rare as~~ contrived or rare as you might suspect.

~~We will~~ As we will soon see, there are many examples of propositions that ~~are~~ seem to be true when you check a few cases, but which turn out to be false. The key to remember is that you

Euler (pronounced "oiler") conjectured this in 1769. But the proposition was proven false 218 years later by Noam Elkies at a liberal arts school up Mass Ave.

The solution he found was $a = 95800, b = 217519, c = 414560, d = 422481$.

In logical notation, Proposition 1.5.2 could be written,

$$\forall a \in \mathbb{Z}^+ \forall b \in \mathbb{Z}^+ \forall c \in \mathbb{Z}^+ \forall d \in \mathbb{Z}^+ . \; a^4 + b^4 + c^4 \neq d^4.$$

Here, $\mathbb{Z}^+$ is a symbol for the positive integers. Strings of $\forall$'s like this are usually abbreviated for easier reading, as follows:

$$\forall a, b, c, d \in \mathbb{Z}^+ . \; a^4 + b^4 + c^4 \neq d^4.$$

Proposition 1.5.3. *$313(x^3 + y^3) = z^3$ has no solution when $x, y, z \in \mathbb{Z}^+$.*

This proposition is also false, but the smallest counterexample has more than 1000 digits!

Proposition 1.5.4. *Every map can be colored with 4 colors so that adjacent[2] regions have different colors.*

[2] Two regions are adjacent only when they share a boundary segment of positive length. They are not considered to be adjacent if their boundaries meet only at a few points.

Insert BS ~~(Goes in to Section 4.8)~~

you may be wondering why anyone would care whether or not there is a solution to $313(x^3+y^3) = z^3$ where x, y and z are positive integers.

~~In fact, note~~

It turns out that find solutions to such equations is important in the field of elliptic curves, which turns out to be important to the ~~field~~ study of factoring large integers, which turns out (as we will see in chapter 4) to be important ~~for~~ in cracking ~~the~~ commonly used cryptosystems ~~which~~. which is why ~~a~~ mathematicians went to the effort to find the solution with thousands of digits.

that have infinitely many cases ~~to~~ to check turn out to be false. The following proposition (known as the "Four-Color Theorem") turns out to be true.

A The proof of this proposition is difficult and took ~~many etc~~ over a century to perfect. Along the way, many incorrect proofs were proposed,

~~This proposition is true and is known as the "Four-Color Theorem". However,~~ there have been many ~~incorrect~~ proofs, including one that stood for 10 years in the late 19th century before the mistake was found. An extremely laborious proof was finally found in 1976 by mathematicians Appel and Haken, who used a complex computer program to categorize the four-colorable maps; the program left a few thousand maps uncategorized, and these were checked by hand by Haken and his assistants—including his 15-year-old daughter. There was a lot of debate about whether this was a legitimate proof: the proof was too big to be checked without a computer, and no one could guarantee that the computer calculated correctly, nor did anyone have the energy to recheck the four-colorings of thousands of maps that were done by hand. Within the past decade, a mostly intelligible proof of the Four-Color Theorem was found, though a computer is still needed to check the colorability of several hundred special maps.[3]

[3]See http://www.math.gatech.edu/~thomas/FC/fourcolor.html

The story of the Four-Color Proof is told in a well-reviewed popular (non-technical) book: "Four Colors Suffice. How the Map Problem was Solved." *Robin Wilson*. Princeton Univ. Press, 2003, 276pp.

Proposition 1.5.5 (Goldbach). *Every even integer greater than 2 is the sum of two primes.*

No one knows whether this proposition is true or false. It is known as *Goldbach's Conjecture*, and dates back to 1742.

For a computer scientist, some of the most important things to prove are the "correctness" of programs and systems, i.e. whether a program or system does what it's supposed to. Programs are notoriously buggy, and there's a growing community of researchers and practitioners trying to find ways to prove program correctness. These efforts have been successful enough in the case of CPU chips that they are now routinely used by leading chip manufacturers to prove chip correctness and avoid mistakes like the notorious Intel division bug in the 1990's.

Developing mathematical methods to verify programs and systems remains an active research area. We'll consider some of these methods later in the course.

ISBN 0-691-11533-8.

Insert B6 (put in location A9)

In some cases, we do not know whether or not a proposition is true. For example, the following simple proposition (known as Goldbach's Conjecture) has been heavily studied for over since 1742 but we still do not know if it is true. Of course, it has been checked by computer for many values of n, but as we have seen, that is not enough sufficient to conclude that it is true.

1,3.2

1.5.2 Predicates

A *predicate* is a proposition whose truth depends on the value of one or more variables. Most of the propositions above were defined in terms of predicates. For example,

"n is a perfect square"

is a predicate whose truth depends on the value of n. The predicate is true for $n = 4$ since four is a perfect square, but false for $n = 5$ since five is not a perfect square.

Like other propositions, predicates are often named with a letter. Furthermore, a function-like notation is used to denote a predicate supplied with specific variable values. For example, we might name our earlier predicate P:

$$P(n) ::= \text{"}n \text{ is a perfect square"}$$

Now $P(4)$ is true, and $P(5)$ is false.

This notation for predicates is confusingly similar to ordinary function notation. If P is a predicate, then $P(n)$ is either *true* or *false*, depending on the value

of n. On the other hand, if p is an ordinary function, like $n^2 + 1$, then $p(n)$ is a *numerical quantity*. **Don't confuse these two!**

1.3.3

1.5.3 Quantifiers

There are a couple of assertions commonly made about a predicate: that it is *sometimes* true and that it is *always* true. For example, the predicate

$$\text{“}x^2 \geq 0\text{”}$$

is always true when x is a real number. On the other hand, the predicate

$$\text{“}5x^2 - 7 = 0\text{”}$$

is only sometimes true; specifically, when $x = \pm\sqrt{7/5}$.

There are several ways to express the notions of "always true" and "sometimes true" in English. The table below gives some general formats on the left and specific examples using those formats on the right. You can expect to see such phrases hundreds of times in mathematical writing!

Always True

For all n, $P(n)$ is true.	For all $x \in \mathbb{R}$, $x^2 \geq 0$.
$P(n)$ is true for every n.	$x^2 \geq 0$ for every $x \in \mathbb{R}$.

Sometimes True

There exists an n such that $P(n)$ is true.	There exists an $x \in \mathbb{R}$ such that $5x^2 - 7 = 0$.
$P(n)$ is true for some n.	$5x^2 - 7 = 0$ for some $x \in \mathbb{R}$.
$P(n)$ is true for at least one n.	$5x^2 - 7 = 0$ for at least one $x \in \mathbb{R}$.

All these sentences quantify how often the predicate is true. Specifically, an assertion that a predicate is always true is called a *universal* quantification, and an assertion that a predicate is sometimes true is an *existential* quantification. Sometimes the English sentences are unclear with respect to quantification:

"If you can solve any problem we come up with, then you get an A for the course."

The phrase "you can solve any problem we can come up with" could reasonably be interpreted as either a universal or existential quantification:

"you can solve *every* problem we come up with,"

or maybe

"you can solve *at least one* problem we come up with."

In the preceding example, the quantified phrase appears inside a larger if-then state-

ment. This is quite normal; quantified statements are themselves propositions and can be combined with and, or, implies, etc., just like any other proposition.

1.5.4 More Cryptic Notation

There are symbols to represent universal and existential quantification, just as there are symbols for "and" ($\wedge$), "implies" ($\longrightarrow$), and so forth. In particular, to say that a predicate, P, is true for all values of x in some set, D, one writes:

$$\forall x \in D.\ P(x)$$

The symbol $\forall$ is read "for all", so this whole expression is read "for all x in D, $P(x)$ is true". To say that a predicate $P(x)$ is true for at least one value of x in D, one writes:

$$\exists x \in D.\ P(x)$$

The backward-E, $\exists$, is read "there exists". So this expression would be read, "There exists an x in D such that $P(x)$ is true." The symbols $\forall$ and $\exists$ are always followed by a variable —usually with an indication of the set the variable ranges over —and

then a predicate, as in the two examples above.

As an example, let Probs be the set of problems we come up with, Solves(x) be the predicate "You can solve problem x", and G be the proposition, "You get an A for the course." Then the two different interpretations of

> "If you can solve any problem we come up with, then you get an A for the course."

can be written as follows:

$$(\forall x \in \text{Probs. Solves}(x)) \text{ IMPLIES } G,$$

or maybe

$$(\exists x \in \text{Probs. Solves}(x)) \text{ IMPLIES } G.$$

1.3.5

1.5.5 Mixing Quantifiers

Many mathematical statements involve several quantifiers. For example, *Goldbach's Conjecture* states:

> "Every even integer greater than 2 is the sum of two primes."

Let's write this more verbosely to make the use of quantification clearer:

> For every even integer n greater than 2, there exist primes p and q such that $n = p + q$.

Let Evens be the set of even integers greater than 2, and let Primes be the set of primes. Then we can write Goldbach's Conjecture in logic notation as follows:

$$\underbrace{\forall n \in \text{Evens}}_{\text{for every even integer } n > 2} \underbrace{\exists p, q \in \text{Primes}}_{\text{there exist primes } p \text{ and } q \text{ such that}} . \; n = p + q.$$

OK as it was

More simply: The proposition can also be written more simply as $\forall n \in \text{Evens}\ \exists p, q \in \text{Primes}.\ p + q = n.$

1.5.6 Order of Quantifiers

1.3.6

Swapping the order of different kinds of quantifiers (existential or universal) usually changes the meaning of a proposition. For example, let's return to one of our initial, confusing statements:

"Every American has a dream."

This sentence is ambiguous because the order of quantifiers is unclear. Let A be the set of Americans, let D be the set of dreams, and define the predicate $H(a, d)$ to be "American a has dream d." Now the sentence could mean that there is a single

dream that every American shares:

$$\exists d \in D\ \forall a \in A.\ H(a, d)$$

For example, it might be that every American shares the dream of owning their own home.

Or it could mean that every American has a personal dream:

$$\forall a \in A\ \exists d \in D.\ H(a, d)$$

For example, some Americans may dream of a peaceful retirement, while others dream of continuing practicing their profession as long as they live, and still others may dream ~~of being so rich they needn't think at all about work.~~ ~~about finishing their problem set~~ OK as it was.

Swapping quantifiers in Goldbach's Conjecture creates a patently false statement that every even number ≥ 2 is the sum of *the same* two primes: ; namely

$$\underbrace{\exists p \in \text{Primes}\ \exists q \in \text{Primes}}_{\text{there exist primes } p \text{ and } q \text{ such that}}\ \underbrace{\forall n \in \text{Evens},}_{\text{for every even integer } n > 2}\ n = p + q.$$

make this be a subsection

1.3.7

Variables Over One Domain

When all the variables in a formula are understood to take values from the same nonempty set, D, it's conventional to omit mention of D. For example, instead of $\forall x \in D\ \exists y \in D.\ Q(x, y)$ we'd write $\forall x \exists y.\ Q(x, y)$. The unnamed nonempty set that x and y range over is called the *domain of discourse*, or just plain *domain*, of the formula.

It's easy to arrange for all the variables to range over one domain. For example, Goldbach's Conjecture could be expressed with all variables ranging over the domain $\mathbb{N}$ as

$$\forall n.\ n \in \text{Evens IMPLIES } (\exists p \exists q.\ p \in \text{Primes} \wedge q \in \text{Primes} \wedge n = p + q).$$

1.3.8

1.5.7 Negating Quantifiers

There is a simple relationship between the two kinds of quantifiers. The following two sentences mean the same thing:

It is not the case that everyone likes to snowboard.

There exists someone who does not like to snowboard.

In terms of logic notation, this follows from a general property of predicate formulas:

$$\text{NOT}\,\forall x.\ P(x) \text{ is equivalent to } \exists x.\ \text{NOT}\,P(x).$$

Similarly, these sentences mean the same thing:

There does not exist anyone who likes skiing over magma.

Everyone dislikes skiing over magma.

We can express the equivalence in logic notation this way:

$$(\text{NOT}\,\exists x.\ P(x)) \text{ IFF } \forall x.\ \text{NOT}\,P(x). \tag{1.5}$$

The general principle is that *moving a "not" across a quantifier changes the kind of quantifier.*

1.5.8 Validity

A propositional formula is called *valid* when it evaluates to T no matter what truth values are assigned to the individual propositional variables. For example, the

propositional version of the Distributive Law is that P AND $(Q$ OR $R)$ is equivalent to $(P$ AND $Q)$ OR $(P$ AND $R)$. This is the same as saying that

$$[P \text{ AND } (Q \text{ OR } R)] \text{ IFF } [(P \text{ AND } Q) \text{ OR } (P \text{ AND } R)]$$

is valid.

The same idea extends to predicate formulas, but to be valid, a formula now must evaluate to true no matter what values its variables may take over any unspecified domain, and no matter what interpretation a predicate variable may be given. For example, we already observed that the rule for negating a quantifier is captured by the valid assertion (1.5).

Another useful example of a valid assertion is

$$\exists x \forall y.\ P(x,y) \text{ IMPLIES } \forall y \exists x.\ P(x,y). \tag{1.6}$$

Here's an explanation why this is valid:

Let D be the domain for the variables and P_0 be some binary predicate[4]

[4]That is, a predicate that depends on two variables.

on D. We need to show that if

$$\exists x \in D\, \forall y \in D.\ P_0(x, y) \tag{1.7}$$

holds under this interpretation, then so does

$$\forall y \in D\, \exists x \in D.\ P_0(x, y). \tag{1.8}$$

So suppose (1.7) is true. Then by definition of $\exists$, this means that some element $d_0 \in D$ has the property that

$$\forall y \in D.\, P_0(d_0, y).$$

By definition of $\forall$, this means that

$$P_0(d_0, d)$$

is true for all $d \in D$. So given any $d \in D$, there is an element in D, namely, d_0, such that $P_0(d_0, d)$ is true. But that's exactly what (1.8) means, so we've proved that (1.8) holds under this interpretation, as required.

We hope this is helpful as an explanation, but we don't really want to call it a "proof." The problem is that with something as basic as (1.6), it's hard to see what more elementary axioms are ok to use in proving it. What the explanation above did was translate the logical formula (1.6) into English and then appeal to the meaning, in English, of "for all" and "there exists" as justification. So this wasn't a proof, just an explanation that once you understand what (1.6) means, it becomes obvious.

In contrast to (1.6), the formula

$$\forall y \exists x.\ P(x,y) \text{ IMPLIES } \exists x \forall y.\ P(x,y). \tag{1.9}$$

is *not* valid. We can prove this just by describing an interpretation where the hypothesis, $\forall y \exists x.\ P(x,y)$, is true but the conclusion, $\exists x \forall y.\ P(x,y)$, is not true. For example, let the domain be the integers and $P(x,y)$ mean $x > y$. Then the hypothesis would be true because, given a value, n, for y we could choose the value of x to be $n+1$, for example. But under this interpretation the conclusion asserts that there is an integer that is bigger than all integers, which is certainly false. An

interpretation like this which falsifies an assertion is called a *counter model* to the assertion.

1.5.9 Problems

Class Problems

Homework Problems

Location A10
(the satisfiability section goes here.)

1.6 Problems

Albert will supply these and they will be organized according to sections.

We hope this is helpful as an explanation, but we don't really want to call it a "proof." The problem is that with something as basic as (1.6), it's hard to see what more elementary axioms are ok to use in proving it. What the explanation above did was translate the logical formula (1.6) into English and then appeal to the meaning, in English, of "for all" and "there exists" as justification. So this wasn't a proof, just an explanation that once you understand what (1.6) means, it becomes obvious.

In contrast to (1.6), the formula

$$\forall y \exists x.\ P(x,y) \text{ IMPLIES } \exists x \forall y.\ P(x,y). \tag{1.9}$$

is *not* valid. We can prove this by describing an interpretation where the hypothesis, $\forall y \exists x.\ P(x,y)$, is true but the conclusion, $\exists x \forall y.\ P(x,y)$, is not true. For example, let the domain be the integers and $P(x,y)$ mean $x > y$. Then the hypothesis would be true because, given a value, n, for y we could choose the value of x to be $n+1$. But under this interpretation the conclusion asserts that there is an integer that is bigger than all integers, which is certainly false. An

interpretation like this which falsifies an assertion is called a *counter model* to the assertion.

1.5.9 Problems

Class Problems

Homework Problems

Location A10
(the satisfiability section goes here.)

1.6 Problems

Albert will supply these and they will be organized according to sections.

Chapter 2

Patterns of Proof

~~Proof Templates~~

2.1 The Axiomatic Method

The standard procedure for establishing truth in mathematics was invented by Euclid, a mathematician working in Alexandria, Egypt around 300 BC. His idea was to begin with five *assumptions* about geometry, which seemed undeniable based on direct experience. For example, "There is a straight line segment between every pair of points." Propositions like these that are simply accepted as true are called

axioms.

Starting from these axioms, Euclid established the truth of many additional propositions by providing "proofs". A *proof* is a sequence of logical deductions from axioms and previously-proved statements that concludes with the proposition in question. You probably wrote many proofs in high school geometry class, and you'll see a lot more in this course.

There are several common terms for a proposition that has been proved. The different terms hint at the role of the proposition within a larger body of work.

- Important propositions are called *theorems*.
- A *lemma* is a preliminary proposition useful for proving later propositions.
- A *corollary* is a proposition that follows in just a few logical steps from a theorem.

The definitions are not precise. In fact, sometimes a good lemma turns out to be far more important than the theorem it was originally used to prove.

Euclid's axiom-and-proof approach, now called the *axiomatic method*, is the

foundation for mathematics today. In fact, just a handful of axioms, called the axioms Zermelo-Frankel with Choice (*ZFC*), together with a few logical deduction rules, appear to be sufficient to derive essentially all of mathematics. We'll examine these in Chapter 3.

This should be a subsection

2.1.1

2.2 Our Axioms

The ZFC axioms are important in studying and justifying the foundations of mathematics, but for practical purposes, they are much too primitive. Proving theorems in ZFC is a little like writing programs in byte code instead of a full-fledged programming language —by one reckoning, a formal proof in ZFC that $2 + 2 = 4$ requires more than 20,000 steps! So instead of starting with ZFC, we're going to take a *huge* set of axioms as our foundation: we'll accept all familiar facts from high school math!

This will give us a quick launch, but you may find this imprecise specification of the axioms troubling at times. For example, in the midst of a proof, you may

find yourself wondering, "Must I prove this little fact or can I take it as an axiom?" Feel free to ask for guidance, but really there is no absolute answer. Just be up front about what you're assuming, and don't try to evade homework and exam problems by declaring everything an axiom!

2.1.2 Next subsection

~~2.2.1~~ Logical Deductions

Logical deductions or *inference rules* are used to prove new propositions using previously proved ones.

A fundamental inference rule is *modus ponens*. This rule says that a proof of P together with a proof that P IMPLIES Q is a proof of Q.

Inference rules are sometimes written in a funny notation. For example, *modus ponens* is written:

Rule.

$$\frac{P, \quad P \text{ IMPLIES } Q}{Q}$$

When the statements above the line, called the *antecedents*, are proved, then we

can consider the statement below the line, called the *conclusion* or *consequent*, to also be proved.

A key requirement of an inference rule is that it must be *sound*: any assignment of truth values that makes all the antecedents true must also make the consequent true. So if we start off with true axioms and apply sound inference rules, everything we prove will also be true.

Put insert @C1 here

There are many other natural, sound inference rules, for example:

Rule.

$$\frac{P \text{ IMPLIES } Q, \quad Q \text{ IMPLIES } R}{P \text{ IMPLIES } R}$$

Rule.

$$\frac{\text{NOT}(P) \text{ IMPLIES NOT}(Q)}{Q \text{ IMPLIES } P}$$

On the other hand,

Insert C1 ~~Eqocs.~~

~~The following proposition is even~~

~~easier. It is false but the smallest~~

You can see why modus ponens is a sound inference rule by checking the truth table for ~~it~~ there is P IMPLIES Q. ~~and Q.~~ only one case where P and P IMPLIES Q are both true, and in that case, Q is also true.

P	Q	P IMPLIES Q
F	F	T
F	T	T
T	F	F
T	T	T

~~Leave T~~

Here's the good news: many proofs follow one of a handful of standard templates. Each proof has it own details, of course, but these templates ~~at least~~ provide you with an outline to fill in. ~~We'll~~ In the remainder of this chapter, we'll go through several of these standard patterns, pointing out the basic idea and common pitfalls and giving some examples. Many of these templates fit together; one may give you a top-level outline while others help you at the next level of detail. And we'll show you other, more sophisticated proof techniques ~~later on~~ in chapter 3,

The recipes ~~below~~ that follow are very specific at times, telling you exactly which words to write down on your piece of paper. You're certainly free to say things your own way instead; we're just giving you something you *could* say so that you're never at a complete loss.

move the section "Proof by cases" here

2.3 Proving an Implication

make subsection 2.2.2

Propositions of the form "If P, then Q" are called *implications*. This implication is often rephrased as "P IMPLIES Q", or "$P \Rightarrow Q$".

Here are some examples of implications.

- (Quadratic Formula) If $ax^2 + bx + c = 0$ and $a \neq 0$, then

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

- (Goldbach's Conjecture) If n is an even integer greater than 2, then n is a sum of two primes.

- If $0 \leq x \leq 2$, then $-x^3 + 4x + 1 > 0$.

There are a couple of standard methods for proving an implication.

subsubsection

2.3.1 Method #1 – Assume P is True.

INSERT C5

In order to prove that P IMPLIES Q:

1. Write, "Assume P."

2. Show that Q logically follows.

For example, we will use this method to prove

Example

Theorem 2.3.1. *If $0 \leq x \leq 2$, then $-x^3 + 4x + 1 > 0$.*

INSERT C5

~~This is because~~ This method is really an example of proof by cases in disguise.

In particular, when proving P IMPLIES Q, there are to consider: two cases A P is ~~false~~ false and P is true. ~~false~~. The case when P is false is easy since, by definition, F IMPLIES Q is true no matter what Q is. This case is so easy that we usually just forget about it and start right off by assuming that P is true when proving an implication, since this is the only case that ~~really matters.~~ is interesting.

Hence, in

Before we write a proof of this theorem, we have to do some scratchwork to figure out why it is true.

The inequality certainly holds for $x = 0$; then the left side is equal to 1 and $1 > 0$. As x grows, the $4x$ term (which is positive) initially seems to have greater magnitude than $-x^3$ (which is negative). For example, when $x = 1$, we have $4x = 4$, but $-x^3 = -1$ only. In fact, it looks like $-x^3$ doesn't begin to dominate until $x > 2$. So it seems the $-x^3 + 4x$ part should be nonnegative for all x between 0 and 2, which would imply that $-x^3 + 4x + 1$ is positive.

So far, so good. But we still have to replace all those "seems like" phrases with solid, logical arguments. We can get a better handle on the critical $-x^3 + 4x$ part by factoring it, which is not too hard:

$$-x^3 + 4x = x(2 - x)(2 + x)$$

Aha! For x between 0 and 2, all of the terms on the right side are nonnegative. And a product of nonnegative terms is also nonnegative. Let's organize this blizzard of observations into a clean proof.

Proof. Assume $0 \leq x \leq 2$. Then x, $2 - x$, and $2 + x$ are all nonnegative. Therefore, the product of these terms is also nonnegative. Adding 1 to this product gives a positive number, so:

$$x(2 - x)(2 + x) + 1 > 0$$

Multiplying out on the left side proves that

$$-x^3 + 4x + 1 > 0$$

as claimed. ■

There are a couple points here that apply to all proofs:

- You'll often need to do some scratchwork while you're trying to figure out the logical steps of a proof. Your scratchwork can be as disorganized as you like— full of dead-ends, strange diagrams, obscene words, whatever. But keep your scratchwork separate from your final proof, which should be clear and concise.

- Proofs typically begin with the word "Proof" and end with some sort of

doohickey like □ or "q.e.d". The only purpose for these conventions is to clarify where proofs begin and end.

Insert C4 goes here

Subsubsection

2.3.2 Method #2 - Prove the Contrapositive

We have already seen that an implication ("P IMPLIES Q") is logically equivalent to its *contrapositive*

NOT(Q) IMPLIES NOT(P).

Proving one is as good as proving the other, and proving the contrapositive is sometimes easier than proving the original statement. Hence, you can proceed as follows:

1. Write, "We prove the contrapositive:" and then state the contrapositive.
2. Proceed as in Method #1.

For example, we can use this approach to prove

Example

Theorem 2.3.2. *If r is irrational, then $\sqrt{r}$ is also irrational.*

Recall that rational numbers are equal to a ratio of integers and irrational num-

~~Warning~~ ← subsubsection

Insert C4

Pitfall When using this method to prove an implication, be sure to not get confused and assume that P is true after the proof of the implication is completed. For example, in the proof of Theorem 2.3.1, it is ok to assume (i.e., that P is true) since this ~~is the only case when proving~~ ~~$(0 \le x \le 2)$ IMPLIES $(-x^3+4x+1>0)$~~ is a nontrivial case when proving $(0 \le x \le 2)$ IMPLIES $(-x^3+4x+1>0)$, but it need not be true in general. Indeed, if you were then going on to prove another result using the variable x, it could be disastrous to have a step where you assume that $0 \le x \le 2$ just because it as part of you assumed the proof of Theorem 2.3.1.

bers are not. So we must show that if r is *not* a ratio of integers, then $\sqrt{r}$ is also *not* a ratio of integers. That's pretty convoluted! We can eliminate both *not*'s and make the proof straightforward by considering the contrapositive instead.

remove this space

Proof. We prove the contrapositive: if $\sqrt{r}$ is rational, then r is rational.

Assume that $\sqrt{r}$ is rational. Then there exist integers a and b such that:

$$\sqrt{r} = \frac{a}{b}$$

Squaring both sides gives:

$$r = \frac{a^2}{b^2}$$

Since a^2 and b^2 are integers, r is also rational. ■

~~2.3.3 Problems~~

~~Homework Problems~~

subsection

~~2.3~~ ~~2.4~~ 2.2.3 **Proving an "If and Only If"**

Many mathematical theorems assert that two statements are logically equivalent; that is, one holds if and only if the other does. Here is an example that has been known for several thousand years:

> Two triangles have the same side lengths if and only if two side lengths and the angle between those sides are the same in each triangle

The phrase "if and only if" comes up so often that it is often abbreviated "iff".

subsub section ~~2.4.1~~

2.4.1 Method #1: Prove Each Statement Implies the Other

The statement "P IFF Q" is equivalent to the two statements "P IMPLIES Q" and "Q IMPLIES P". So you can prove an "iff" by proving *two* implications:

1. Write, "We prove P implies Q and vice-versa."

2. Write, "First, we show P implies Q." Do this by one of the methods in Section 2.3.

3. Write, "Now, we show Q implies P." Again, do this by one of the methods in Section 2.3.

2.4.2 Method #2: Construct a Chain of Iffs

In order to prove that P is true iff Q is true:

1. Write, "We construct a chain of if-and-only-if implications."

2. Prove P is equivalent to a second statement which is equivalent to a third statement and so forth until you reach Q.

This method sometimes requires more ingenuity than the first, but the result can be a short, elegant proof, as we see in the following example.

Example

Definition:

The *standard deviation* of a sequence of values $x_1, x_2, \ldots, x_n$ is defined to be:

$$\sqrt{\frac{(x_1-\mu)^2+(x_2-\mu)^2+\cdots+(x_n-\mu)^2}{n}} \tag{2.1}$$

where μ is the *mean* of the values:

$$\mu ::= \frac{x_1+x_2+\cdots+x_n}{n}$$

Theorem 2.4.1. *The standard deviation of a sequence of values $x_1, \ldots, x_n$ is zero iff all the values are equal to the mean.*

Again

Theorem 2.4.1 says that

For example, the standard deviation of test scores is zero if and only if everyone scored exactly the class average. (we will talk alot more about means and standard deviations in Part IV of the book.)

Proof. We construct a chain of "iff" implications, starting with the statement that the standard deviation (2.1) is zero:

$$\sqrt{\frac{(x_1-\mu)^2+(x_2-\mu)^2+\cdots+(x_n-\mu)^2}{n}} = 0. \tag{2.2}$$

Now since zero is the only number whose square root is zero, equation (2.2) holds

iff

$$(x_1 - \mu)^2 + (x_2 - \mu)^2 + \cdots + (x_n - \mu)^2 = 0. \tag{2.3}$$

Now squares of real numbers are always nonnegative, so every term on the left hand side of equation (2.3) is nonnegative. This means that (2.3) holds iff

Every term on the left hand side of (2.3) is zero. (2.4)

But a term $(x_i - \mu)^2$ is zero iff $x_i = \mu$, so (2.4) is true iff

Every x_i equals the mean.

■

2.5 Proof by Cases

Breaking a complicated proof into cases and proving each case separately is a useful common proof strategy. Here's an amusing example.

Let's agree that given any two people, either they have met or not. If every pair of people in a group has met, we'll call the group a *club*. If every pair of people in

Insert C2

In fact, we have implicitly already used this strategy when we used truth tables to show that certain propositions (in Section 1.1.5) were true or valid. For example, we showed that an implication $P \to Q$ is equivalent to its contrapositive $\neg Q \to \neg P$ by considering all possible assignments of T or F to P and Q. In each of the four cases, we showed that $P \to Q$ was true if and only if $\neg Q \to \neg P$ was true. (For example, if $P = T$ and $Q = F$, then both $P \to Q$ and $\neg Q \to \neg P$ are false, thereby establishing that $(P \to Q) \leftrightarrow (\neg Q \to \neg P)$ is true for this case.) Hence we could conclude that $P \to Q$ and $\neg Q \to \neg P$ are equivalent.

Proof by cases works in much more general environments than propositions involving Boolean variables. In what follows, we will use this approach

C2 (cont) ; C2-1

To prove ~~as Fatou's Lemma~~ a simple fact
As background,
about aquaintences. ~~For the pro~~

we will assume that for any pair of people,

a group has not met, we'll call it a group of *strangers*.

Theorem. *Every collection of 6 people includes a club of 3 people or a group of 3 strangers.*

Proof. The proof is by case analysis[1]. Let x denote one of the six people. There are two cases:

1. Among the 5 other people besides x, at least 3 have met x.
2. Among the 5 other people, at least 3 have not met x.

Now we have to be sure that at least one of these two cases must hold,[2] but that's easy: we've split the 5 people into two groups, those who have shaken hands with x and those who have not, so one the groups must have at least half the people.

Case 1: Suppose that at least 3 people have met x.

This case splits into two subcases:

[1] Describing your approach at the outset helps orient the reader. Try to remember to always do this

[2] Part of a case analysis argument is showing that you've covered all the cases. Often this is obvious, because the two cases are of the form "P" and "not P". However, the situation above is not stated quite so simply.

(i.e., those that have met x) have

Case 1.1: No pair among those people met each other. Then these people are a group of at least 3 strangers. So the Theorem holds in this subcase.

Case 1.2: Some pair among those people have met each other. Then that pair, together with x, form a club of 3 people. So the Theorem holds in this subcase.

This implies that the Theorem holds in Case 1.

Case 2: Suppose that at least 3 people did not meet x.

This case also splits into two subcases:

(i.e., those that have not met x) have

Case 2.1: Every pair among those people met each other. Then these people are a club of at least 3 people. So the Theorem holds in this subcase.

Case 2.2: Some pair among those people have not met each other. Then that pair, together with x, form a group of at least 3 strangers. So the Theorem holds in this subcase.

This implies that the Theorem also holds in Case 2, and therefore holds in all cases.

■

2.5.1 Problems

Class Problems

Homework Problems

2.5

2.6 Proof by Contradiction

In a *proof by contradiction* or *indirect proof*, you show that if a proposition were false, then some false fact would be true. Since a false fact can't be true, the proposition had better not be false. That is, the proposition really must be true.

Proof by contradiction is *always* a viable approach. However, as the name suggests, indirect proofs can be a little convoluted. So direct proofs are generally preferable as a matter of clarity.

Method: In order to prove a proposition P by contradiction:

1. Write, "We use proof by contradiction."

2. Write, "Suppose P is false."

3. Deduce something known to be false (a logical contradiction).

4. Write, "This is a contradiction. Therefore, P must be true."

Example

As an example, we will use proof by contradiction to prove that $\sqrt{2}$ is irrational. Recall

Remember that a number is *rational* if it is equal to a ratio of integers. For example,

$3.5 = 7/2$ and $0.1111\cdots = 1/9$ are rational numbers. On the other hand, we'll prove by contradiction that $\sqrt{2}$ is irrational.

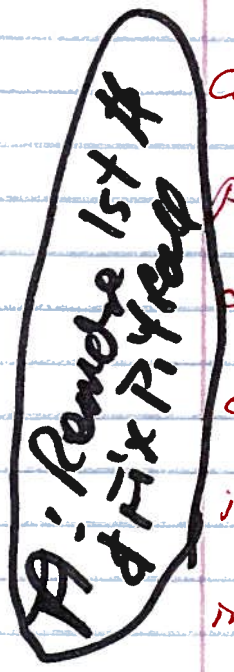
Theorem 2.6.1. *$\sqrt{2}$ is irrational.*

Proof. We use proof by contradiction. Suppose the claim is false; that is, $\sqrt{2}$ is rational. Then we can write $\sqrt{2}$ as a fraction n/d in lowest terms (i.e., so that there is no number greater than 1 that divides both n and d), where n and d are positive integers. Furthermore, let's take n and d so that n/d is

Squaring both sides gives $2 = n^2/d^2$ and so $2d^2 = n^2$. This implies that n is a multiple of 2. Therefore n^2 must be a multiple of 4. But since $2d^2 = n^2$, we know $2d^2$ is a multiple of 4 and so d^2 is a multiple of 2. This implies that d is a multiple

INSERT CB-1

A proof of a proposition P by contradiction is really the same as proving the implication NOT P IMPLIES F by contrapositive. Indeed, the contrapositive of NOT P IMPLIES F is NOT (F) IMPLIES P. As we saw in Section 2.3#, such a proof would begin by assuming NOT (F) in an effort to derive a falsehood, just as you do in a proof by contradiction.



↓ subsubsection

Pitfall

Now matter how you think about it, it is important to remember that when you start by assuming NOT (P), you will derive conclusions along the way that are not necessarily true. (Indeed, the whole point of the method is to derive a

falsehood.) This means that you cannot rely on such ~~results~~ intermediate results ~~on use results derived along the way in other~~ after the proof is completed.

~~When doing a proof by contradiction,~~ it is important to remember that you ~~have started by assuming that~~ that P is false when you really believe that P (and will ultimately prove) is that P is true. ~~This means that~~ you will derive conclusions along the way that are not necessarily (e.g., that n is even in the proof of Theorem 2.6.1 ~~(e.g. $\neg P$)~~ so be careful to not rely on them after the proof by contradiction is completed. There was not much risk of that happening in the proof of Theorem 2.6.1, but when you are doing more complicated proofs that build up from several lemmas, some of which use a proof by contradiction, it will be important to keep track of which

C3 (cont) C3 – Ex 3

follow from an (closed) assumption in a proof by contradiction,

of 2.

So the numerator and denominator have 2 as a common factor, which contradicts the fact that n/d is in lowest terms. So $\sqrt{2}$ must be irrational. ■

move this entire section to become section 5.1

INSERT C3 goes here

~~2.7~~ 5.1 Sets

Propositions of the sort we've considered so far are good for reasoning about individual statements, but not so good for reasoning about a collection of objects. Let's first review a couple mathematical tools for grouping objects and then extend our logical language to cope with such collections.

Informally, a *set* is a bunch of objects, which are called the *elements* of the set. The elements of a set can be just about anything: numbers, points in space, or even other sets. The conventional way to write down a set is to list the elements inside curly-braces. For example, here are some sets:

A	=	$\{\text{Alex}, \text{Tippy}, \text{Shells}, \text{Shadow}\}$	dead pets
B	=	$\{\text{red}, \text{blue}, \text{yellow}\}$	primary colors
C	=	$\{\{a, b\}, \{a, c\}, \{b, c\}\}$	a set of sets

This works fine for small finite sets. Other sets might be defined by indicating how to generate a list of them:

$$D = \{1, 2, 4, 8, 16, \dots\} \qquad \text{the powers of 2}$$

The order of elements is not significant, so $\{x, y\}$ and $\{y, x\}$ are the same set written two different ways. Also, any object is, or is not, an element of a given set —there is no notion of an element appearing more than once in a set.[3] So writing $\{x, x\}$ is just indicating the same thing twice, namely, that x is in the set. In particular, $\{x, x\} = \{x\}$.

The expression $e \in S$ asserts that e is an element of set S. For example, $32 \in D$ and blue $\in B$, but Tailspin $\notin A$ —yet.

Sets are simple, flexible, and everywhere. You'll find some set mentioned in nearly every section of this text.

[3] It's not hard to develop a notion of *multisets* in which elements can occur more than once, but multisets are not ordinary sets.

2.7.1 Some Popular Sets

Mathematicians have devised special symbols to represent some common sets.

symbol	set	elements
$\emptyset$	the empty set	none
$\mathbb{N}$	nonnegative integers	$\{0, 1, 2, 3, \ldots\}$
$\mathbb{Z}$	integers	$\{\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots\}$
$\mathbb{Q}$	rational numbers	$\frac{1}{2}$, $-\frac{5}{3}$, 16, etc.
$\mathbb{R}$	real numbers	π, e, -9, $\sqrt{2}$, etc.
$\mathbb{C}$	complex numbers	i, $\frac{19}{2}$, $\sqrt{2} - 2i$, etc.

A superscript "$+$" restricts a set to its positive elements; for example, $\mathbb{R}^+$ denotes the set of positive real numbers. Similarly, $\mathbb{R}^-$ denotes the set of negative reals.

2.7.2 Comparing and Combining Sets

The expression $S \subseteq T$ indicates that set S is a *subset* of set T, which means that every element of S is also an element of T (it could be that $S = T$). For example, $\mathbb{N} \subseteq \mathbb{Z}$ and $\mathbb{Q} \subseteq \mathbb{R}$ (every rational number is a real number), but $\mathbb{C} \not\subseteq \mathbb{Z}$ (not every complex number is an integer).

As a memory trick, notice that the $\subseteq$ points to the smaller set, just like a $\leq$ sign points to the smaller number. Actually, this connection goes a little further: there is a symbol $\subset$ analogous to $<$. Thus, $S \subset T$ means that S is a subset of T, but the

two are *not* equal. So $A \subseteq A$, but $A \not\subset A$, for every set A.

There are several ways to combine sets. Let's define a couple of sets for use in examples:

$$X ::= \{1, 2, 3\}$$

$$Y ::= \{2, 3, 4\}$$

- The *union* of sets X and Y (denoted $X \cup Y$) contains all elements appearing in X or Y or both. Thus, $X \cup Y = \{1, 2, 3, 4\}$.

- The *intersection* of X and Y (denoted $X \cap Y$) consists of all elements that appear in *both* X and Y. So $X \cap Y = \{2, 3\}$.

- The *set difference* of X and Y (denoted $X - Y$) consists of all elements that are in X, but not in Y. Therefore, $X - Y = \{1\}$ and $Y - X = \{4\}$.

2.7.3 Complement of a Set

Sometimes we are focused on a particular domain, D. Then for any subset, A, of D, we define $\overline{A}$ to be the set of all elements of D *not* in A. That is, $\overline{A} ::= D - A$. The

set $\overline{A}$ is called the *complement* of A.

For example, when the domain we're working with is the real numbers, the complement of the positive real numbers is the set of negative real numbers together with zero. That is,

$$\overline{\mathbb{R}^+} = \mathbb{R}^- \cup \{0\}\,.$$

It can be helpful to rephrase properties of sets using complements. For example, two sets, A and B, are said to be *disjoint* iff they have no elements in common, that is, $A \cap B = \emptyset$. This is the same as saying that A is a subset of the complement of B, that is, $A \subseteq \overline{B}$.

2.7.4 Power Set

The set of all the subsets of a set, A, is called the *power set*, $\mathcal{P}(A)$, of A. So $B \in \mathcal{P}(A)$ iff $B \subseteq A$. For example, the elements of $\mathcal{P}(\{1,2\})$ are $\emptyset$, $\{1\}$, $\{2\}$ and $\{1,2\}$.

More generally, if A has n elements, then there are 2^n sets in $\mathcal{P}(A)$. For this reason, some authors use the notation 2^A instead of $\mathcal{P}(A)$.

2.7.5 Set Builder Notation

An important use of predicates is in *set builder notation*. We'll often want to talk about sets that cannot be described very well by listing the elements explicitly or by taking unions, intersections, etc., of easily-described sets. Set builder notation often comes to the rescue. The idea is to define a *set* using a *predicate;* in particular, the set consists of all values that make the predicate true. Here are some examples of set builder notation:

$$A ::= \{n \in \mathbb{N} \mid n \text{ is a prime and } n = 4k + 1 \text{ for some integer } k\}$$

$$B ::= \{x \in \mathbb{R} \mid x^3 - 3x + 1 > 0\}$$

$$C ::= \{a + bi \in \mathbb{C} \mid a^2 + 2b^2 \leq 1\}$$

The set A consists of all nonnegative integers n for which the predicate

"n is a prime and $n = 4k + 1$ for some integer k"

is true. Thus, the smallest elements of A are:

$$5, 13, 17, 29, 37, 41, 53, 57, 61, 73, \ldots.$$

Trying to indicate the set A by listing these first few elements wouldn't work very well; even after ten terms, the pattern is not obvious! Similarly, the set B consists of all real numbers x for which the predicate

$$x^3 - 3x + 1 > 0$$

is true. In this case, an explicit description of the set B in terms of intervals would require solving a cubic equation. Finally, set C consists of all complex numbers $a + bi$ such that:

$$a^2 + 2b^2 \leq 1$$

This is an oval-shaped region around the origin in the complex plane.

2.7.6 Proving Set Equalities

Two sets are defined to be equal if they contain the same elements. That is, $X = Y$ means that $z \in X$ if and only if $z \in Y$, for all elements, z. (This is actually the

first of the ZFC axioms.) So set equalities can be formulated and proved as "iff" theorems. For example:

Theorem 2.7.1 (*Distributive Law* for Sets). *Let A, B, and C be sets. Then:*

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \tag{2.5}$$

Proof. The equality (2.5) is equivalent to the assertion that

$$z \in A \cap (B \cup C) \quad \text{iff} \quad z \in (A \cap B) \cup (A \cap C) \tag{2.6}$$

for all z. Now we'll prove (2.6) by a chain of iff's.

First we need a rule for distributing a propositional AND operation over an OR operation. It's easy to verify by truth-table that

Lemma 2.7.2. *The propositional formula*

$$P \text{ AND } (Q \text{ OR } R)$$

and

$$(P \text{ AND } Q) \text{ OR } (P \text{ AND } R)$$

are equivalent.

Now we have

$z \in A \cap (B \cup C)$

iff $(z \in A)$ AND $(z \in B \cup C)$ (def of $\cap$)

iff $(z \in A)$ AND $(z \in B$ OR $z \in C)$ (def of $\cup$)

iff $(z \in A$ AND $z \in B)$ OR $(z \in A$ AND $z \in C)$ (Lemma 2.7.2)

iff $(z \in A \cap B)$ OR $(z \in A \cap C)$ (def of $\cap$)

iff $z \in (A \cap B) \cup (A \cap C)$ (def of $\cup$)

■

2.7.7 Glossary of Symbols

symbol	meaning
$::=$	is defined to be
$\wedge$	and
$\vee$	or
$\longrightarrow$	implies
$\neg$	not
$\neg P$	not P
$\overline{P}$	not P
$\longleftrightarrow$	iff
$\longleftrightarrow$	equivalent
$\oplus$	xor
$\exists$	exists
$\forall$	for all
$\in$	is a member of
$\subseteq$	is a subset of
$\subset$	is a proper subset of
$\cup$	set union
$\cap$	set intersection
$\overline{A}$	complement of a set, A
$\mathcal{P}(A)$	powerset of a set, A
$\emptyset$	the empty set, $\{\}$

2.7.8 Problems

Homework Problems

2.8 *Good* Proofs in Practice

One purpose of a proof is to establish the truth of an assertion with absolute certainty. Mechanically checkable proofs of enormous length or complexity can accomplish this. But humanly intelligible proofs are the only ones that help someone

understand the subject. Mathematicians generally agree that important mathematical results can't be fully understood until their proofs are understood. That is why proofs are an important part of the curriculum.

To be understandable and helpful, more is required of a proof than just logical correctness: a good proof must also be clear. Correctness and clarity usually go together; a well-written proof is more likely to be a correct proof, since mistakes are harder to hide.

In practice, the notion of proof is a moving target. Proofs in a professional research journal are generally unintelligible to all but a few experts who know all the terminology and prior results used in the proof. Conversely, proofs in the first weeks of a beginning course like 6.042 would be regarded as tediously long-winded by a professional mathematician. In fact, what we accept as a good proof later in the term will be different from what we consider good proofs in the first couple of weeks of 6.042. But even so, we can offer some general tips on writing good proofs:

State your game plan. A good proof begins by explaining the general line of reasoning, for example, "We use case analysis" or "We argue by contradiction."

Keep a linear flow. Sometimes proofs are written like mathematical mosaics, with juicy tidbits of independent reasoning sprinkled throughout. This is not good. The steps of an argument should follow one another in an intelligible order.

A proof is an essay, not a calculation. Many students initially write proofs the way they compute integrals. The result is a long sequence of expressions without explanation, making it very hard to follow. This is bad. A good proof usually looks like an essay with some equations thrown in. Use complete sentences.

Avoid excessive symbolism. Your reader is probably good at understanding words, but much less skilled at reading arcane mathematical symbols. So use words where you reasonably can.

Revise and simplify. Your readers will be grateful.

Introduce notation thoughtfully. Sometimes an argument can be greatly simplified by introducing a variable, devising a special notation, or defining a new term. But do this sparingly since you're requiring the reader to remember all that new stuff. And remember to actually *define* the meanings of new variables, terms, or notations; don't just start using them!

Structure long proofs. Long programs are usually broken into a hierarchy of smaller procedures. Long proofs are much the same. Facts needed in your proof that are easily stated, but not readily proved are best pulled out and proved in preliminary lemmas. Also, if you are repeating essentially the same argument over and over, try to capture that argument in a general lemma, which you can cite repeatedly instead.

Be wary of the "obvious". When familiar or truly obvious facts are needed in a proof, it's OK to label them as such and to not prove them. But remember that what's obvious to you, may not be —and typically is not —obvious to your reader.

Most especially, don't use phrases like "clearly" or "obviously" in an attempt to bully the reader into accepting something you're having trouble proving. Also, go on the alert whenever you see one of these phrases in someone else's proof.

Finish. At some point in a proof, you'll have established all the essential facts you need. Resist the temptation to quit and leave the reader to draw the "obvious" conclusion. Instead, tie everything together yourself and explain why the original claim follows.

The analogy between good proofs and good programs extends beyond structure. The same rigorous thinking needed for proofs is essential in the design of critical computer systems. When algorithms and protocols only "mostly work" due to reliance on hand-waving arguments, the results can range from problematic to catastrophic. An early example was the Therac 25, a machine that provided radiation therapy to cancer victims, but occasionally killed them with massive overdoses due to a software race condition. A more recent (August 2004) exam-

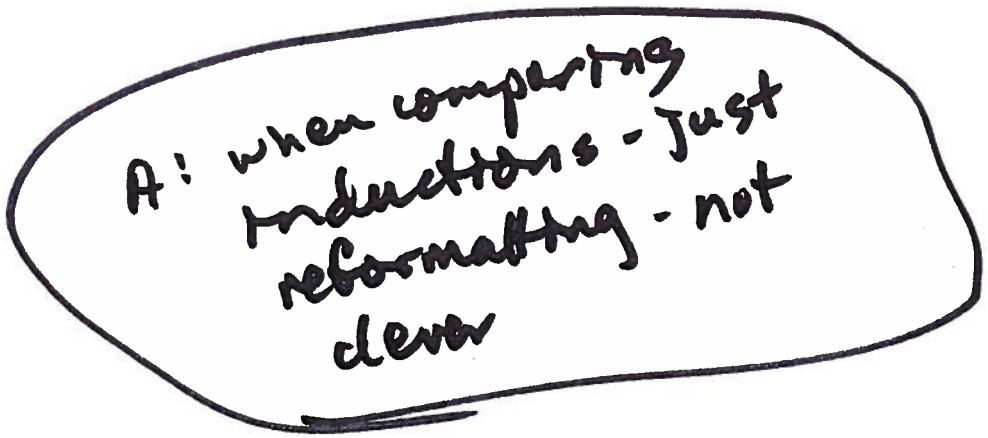
ple involved a single faulty command to a computer system used by United and American Airlines that grounded the entire fleet of both companies— and all their passengers!

It is a certainty that we'll all one day be at the mercy of critical computer systems designed by you and your classmates. So we really hope that you'll develop the ability to formulate rock-solid logical arguments that a system actually does what you think it does!

2.8.1 Problems

Class Problems

Homework Problems



Chapter 3

Induction and the well ordering Principle

insert DI goes here

3.1 The Well Ordering Principle

Every *nonempty* set of *nonnegative integers* has a *smallest* element.

This statement is known as The *Well Ordering Principle*. Do you believe it? Seems sort of obvious, right? But notice how tight it is: it requires a *nonempty* set —it's false for the empty set which has *no* smallest element because it has no

Insert D1

~~In this chapter~~

Now that you understand the basics of how to prove that a proposition is true, ~~we~~ it is time to equip you with ~~some very~~ ~~the most powerful tools~~ the most powerful methods we have for establishing truth: ~~induction~~ the Well-ordering Principle, ~~the~~ Induction Rule, and strong induction. These methods are especially useful when you need to prove that a predicate is true for all natural numbers.

Although the three methods ~~sound~~ look and feel different, it turns out that ~~we will find shown at the end of the chapter that that when you look more closely, they~~ they are equivalent in the sense that whatever you can prove using one of the methods, you can also ~~can~~ prove using either of the others. The choice

~~DT~~ D1 (cont) D1-7

of which method to use ~~depends to~~ depends on whichever seems to be easiest or most natural for the problem at hand.

elements at all! And it requires a set of *nonnegative* integers —it's false for the set of *negative* integers and also false for some sets of nonnegative *rationals* —for example, the set of positive rationals. So, the Well Ordering Principle captures something special about the nonnegative integers.

3.1.1 Well Ordering Proofs

While the Well Ordering Principle may seem obvious, it's hard to see offhand why it is useful. But in fact, it provides one of the most important proof rules in discrete mathematics.

In fact, looking back, we took the Well Ordering Principle for granted in proving that $\sqrt{2}$ is irrational. That proof assumed that for any positive integers m and n, the fraction m/n can be written in *lowest terms*, that is, in the form m'/n' where m' and n' are positive integers with no common factors. How do we know this is always possible?

Suppose to the contrary[1] that there were $m, n \in \mathbb{Z}^+$ such that the fraction m/n cannot be written in lowest terms. Now let C be the set of positive integers that are

[1] This means that you are about to see an informal proof by contradiction.

numerators of such fractions. Then $m \in C$, so C is nonempty. Therefore, by Well Ordering, there must be a smallest integer, $m_0 \in C$. So by definition of C, there is an integer $n_0 > 0$ such that

$$\text{the fraction } \frac{m_0}{n_0} \text{ cannot be written in lowest terms.}$$

This means that m_0 and n_0 must have a common factor, $p > 1$. But

$$\frac{m_0/p}{n_0/p} = \frac{m_0}{n_0},$$

so any way of expressing the left hand fraction in lowest terms would also work for m_0/n_0, which implies

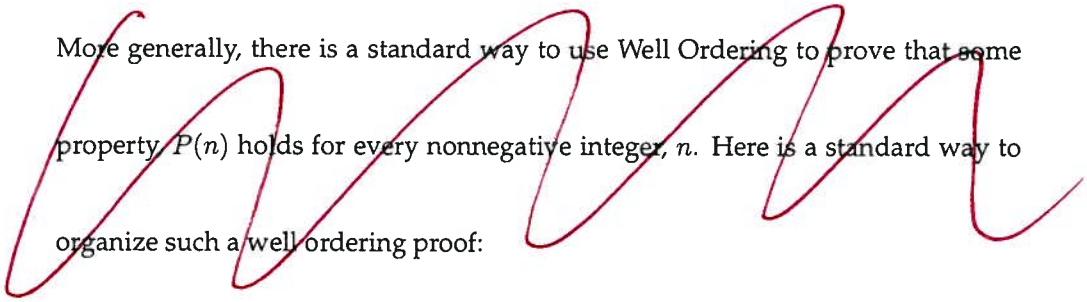
$$\text{the fraction } \frac{m_0/p}{n_0/p} \text{ cannot be in written in lowest terms either.}$$

So by definition of C, the numerator, m_0/p, is in C. But $m_0/p < m_0$, which contradicts the fact that m_0 is the smallest element of C.

Since the assumption that C is nonempty leads to a contradiction, it follows that C must be empty. That is, that there are no numerators of fractions that can't be written in lowest terms, and hence there are no such fractions at all.

We've been using the Well Ordering Principle on the sly from early on!

3.1.2 Template for Well Ordering Proofs



More generally, there is a standard way to use Well Ordering to prove that some property $P(n)$ holds for every nonnegative integer, n. Here is a standard way to organize such a well ordering proof:

more generally, to

To prove that "$P(n)$ is true for all $n \in \mathbb{N}$" using the Well Ordering Principle, you can take the following steps:

- Define the set, C, of *counterexamples* to P being true. Namely, define[a]

$$C ::= \{n \in \mathbb{N} \mid P(n) \text{ is false}\}.$$

- Assume for proof by contradiction that C is nonempty.

- By the Well Ordering Principle, there will be a smallest element, n, in C.

- Reach a contradiction (somehow) —often by showing how to use n to find another member of C that is smaller than n. (This is the open-ended part of the proof task.)

- Conclude that C must be empty, that is, no counterexamples exist. QED

[a]The notation $\{n \mid P(n)\}$ means "the set of all elements n, for which $P(n)$ is true."

Take out of box and use bullets

use a

and assume

is false

false.

3.1.3 Problems

Class Problems

Homework Problems

3.1.3

3.1.4 Summing the Integers Examples

Let's use this this template to prove

Theorem.

$$1+2+3+\cdots+n = n(n+1)/2 \tag{3.1}$$

for all nonnegative integers, n.

First, we better address of a couple of ambiguous special cases before they trip us up:

- If $n = 1$, then there is only one term in the summation, and so $1+2+3+\cdots+n$ is just the term 1. Don't be misled by the appearance of 2 and 3 and the suggestion that 1 and n are distinct terms!

- If $n \leq 0$, then there are no terms at all in the summation. By convention, the

sum in this case is 0.

So while the dots notation is convenient, you have to watch out for these special cases where the notation is misleading! (In fact, whenever you see the dots, you should be on the lookout to be sure you understand the pattern, watching out for the beginning and the end.)

We could have eliminated the need for guessing by rewriting the left side of (3.1) with *summation notation*:

$$\sum_{i=1}^{n} i \quad \text{or} \quad \sum_{1 \le i \le n} i.$$

Both of these expressions denote the sum of all values taken by the expression to the right of the sigma as the variable, i, ranges from 1 to n. Both expressions make it clear what (3.1) means when $n = 1$. The second expression makes it clear that when $n = 0$, there are no terms in the sum, though you still have to know the convention that a sum of no numbers equals 0 (the *product* of no numbers is 1, by the way).

OK, back to the proof:

and use of the well ordering Principle.

Proof. By contradiction. Assume that the theorem is *false*. Then, some nonnegative integers serve as *counterexamples* to it. Let's collect them in a set:

$$C ::= \left\{ n \in \mathbb{N} \mid 1+2+3+\cdots+n \neq \frac{n(n+1)}{2} \right\}.$$

By our assumption that the theorem admits counterexamples, C is a nonempty set of nonnegative integers. So, by the Well Ordering Principle, C has a minimum element, call it c. That is, c is the *smallest counterexample* to the theorem.

Since c is the smallest counterexample, we know that (3.1) is false for $n = c$ but true for all nonnegative integers $n < c$. But (3.1) is true for $n = 0$, so $c > 0$. This means $c-1$ is a nonnegative integer, and since it is less than c, equation (3.1) is true for $c-1$. That is,

$$1+2+3+\cdots+(c-1) = \frac{(c-1)c}{2}.$$

But then, adding c to both sides we get

$$1+2+3+\cdots+(c-1)+c = \frac{(c-1)c}{2} + c = \frac{c^2-c+2c}{2} = \frac{c(c+1)}{2},$$

which means that (3.1) does hold for c, after all! This is a contradiction, and we are done. ∎

3.1.5 Problems

Class Problems

3.1.6 Factoring into Primes

We've previously taken for granted the *Prime Factorization Theorem* that every integer greater than one has a unique[1] expression as a product of prime numbers. This is another of those familiar mathematical facts which are not really obvious. We'll prove the uniqueness of prime factorization in a later chapter, but well ordering gives an easy proof that every integer greater than one can be expressed as *some* product of primes.

Theorem 3.1.1. *Every natural number can be factored as a product of primes.*

Proof. The proof is by Well Ordering. Let C be the set of all integers greater than one that cannot be factored as a product of primes. We assume C is not empty and derive a contradiction.

If C is not empty, there is a least element, $n \in C$, by Well Ordering. The n can't

[1] ...unique up to the order in which the prime factors appear

be prime, because a prime by itself is considered a (length one) product of primes and no such products are in C.

So n must be a product of two integers a and b where $1 < a, b < n$. Since a and b are smaller than the smallest element in C, we know that $a, b \notin C$. In other words, a can be written as a product of primes $p_1 p_2 \cdots p_k$ and b as a product of primes $q_1 \cdots q_l$. Therefore, $n = p_1 \cdots p_k q_1 \cdots q_l$ can be written as a product of primes, contradicting the claim that $n \in C$. Our assumption that $C \neq \emptyset$ must therefore be false. ■

3.2 Induction

Induction is by far the most powerful and commonly-used proof technique in discrete mathematics and computer science. In fact, the use of induction is a defining characteristic of *discrete* —as opposed to *continuous* —mathematics. To understand how it works, suppose there is a professor who brings to class a bottomless bag of assorted miniature candy bars. She offers to share the candy in the following way.

First, she lines the students up in order. Next she states two rules:

1. The student at the beginning of the line gets a candy bar.

2. If a student gets a candy bar, then the following student in line also gets a candy bar.

Let's number the students by their order in line, starting the count with 0, as usual in Computer Science. Now we can understand the second rule as a short description of a whole sequence of statements:

- If student 0 gets a candy bar, then student 1 also gets one.
- If student 1 gets a candy bar, then student 2 also gets one.
- If student 2 gets a candy bar, then student 3 also gets one.

⋮

Of course this sequence has a more concise mathematical description:

If student n gets a candy bar, then student $n+1$ gets a candy bar, for all nonnegative integers n.

So suppose you are student 17. By these rules, are you entitled to a miniature candy bar? Well, student 0 gets a candy bar by the first rule. Therefore, by the second rule, student 1 also gets one, which means student 2 gets one, which means student 3 gets one as well, and so on. By 17 applications of the professor's second rule, you get your candy bar! Of course the rules actually guarantee a candy bar to *every* student, no matter how far back in line they may be.

This should be a subsection

3.2.1 A Rule for

~~3.3~~ Ordinary Induction

The reasoning that led us to conclude that every student gets a candy bar is essentially all there is to induction.

The Principle of Induction.

Let $P(n)$ be a predicate. If

- $P(0)$ is true, and
- $P(n)$ IMPLIES $P(n+1)$ for all nonnegative integers, n,

then

- $P(m)$ is true for all nonnegative integers, m.

Since we're going to consider several useful variants of induction in later sections, we'll refer to the induction method described above as *ordinary induction* when we need to distinguish it. Formulated as a proof rule, this would be

Rule. ***Induction Rule***

$$\frac{P(0), \quad \forall n \in \mathbb{N}. P(n) \text{ IMPLIES } P(n+1)}{\forall m \in \mathbb{N}. P(m)}$$

This general induction rule works for the same intuitive reason that all the students get candy bars, and we hope the explanation using candy bars makes it clear

why the soundness of the ordinary induction can be taken for granted. In fact, the rule is so obvious that it's hard to see what more basic principle could be used to justify it.[2] What's not so obvious is how much mileage we get by using it.

3.3.1 Using Ordinary Induction

Ordinary induction often works directly in proving that some statement about nonnegative integers holds for all of them. For example, here is the formula for the sum of the nonnegative integer that we already proved (equation (3.1)) using the Well Ordering Principle:

Theorem 3.3.1. *For all $n \in \mathbb{N}$,*

$$1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2} \tag{3.2}$$

This time, let's use the Induction Principle to prove Theorem 3.3.1.

Suppose that we define predicate $P(n)$ to be the equation (3.2). Recast in terms of this predicate, the theorem claims that $P(n)$ is true for all $n \in \mathbb{N}$. This is great,

[2] But see section 3.4.

because the induction principle lets us reach precisely that conclusion, provided we establish two simpler facts:

- $P(0)$ is true.

- For all $n \in \mathbb{N}$, $P(n)$ IMPLIES $P(n+1)$.

So now our job is reduced to proving these two statements. The first is true because $P(0)$ asserts that a sum of zero terms is equal to $0(0+1)/2 = 0$, which is true by definition. The second statement is more complicated. But remember the basic plan for proving the validity of any implication: *assume* the statement on the left and then *prove* the statement on the right. In this case, we assume $P(n)$ in order to prove $P(n+1)$, which is the equation

$$1 + 2 + 3 + \cdots + n + (n+1) = \frac{(n+1)(n+2)}{2}. \tag{3.3}$$

These two equations are quite similar; in fact, adding $(n+1)$ to both sides of equa-

tion (3.2) and simplifying the right side gives the equation (3.3):

$$1+2+3+\cdots+n+(n+1) = \frac{n(n+1)}{2} + (n+1)$$

$$= \frac{(n+2)(n+1)}{2}$$

Thus, if $P(n)$ is true, then so is $P(n+1)$. This argument is valid for every nonnegative integer n, so this establishes the second fact required by the induction principle. Therefore, the induction principle says that the predicate $P(m)$ is true for all nonnegative integers, m, so the theorem is proved.

3.2.3

3.3.2 A Template for Induction Proofs

The proof of Theorem 3.3.1 was relatively simple, but even the most complicated induction proof follows exactly the same template. There are five components:

1. **State that the proof uses induction.** This immediately conveys the overall structure of the proof, which helps the reader understand your argument.

2. **Define an appropriate predicate $P(n)$.** The eventual conclusion of the induction argument will be that $P(n)$ is true for all nonnegative n. Thus, you

should define the predicate $P(n)$ so that your theorem is equivalent to (or follows from) this conclusion. Often the predicate can be lifted straight from the proposition that you are trying to prove, ~~claim~~, as in the example above. The predicate $P(n)$ is called the *induction hypothesis*. Sometimes the induction hypothesis will involve several variables, in which case you should indicate which variable serves as n.

3. **Prove that $P(0)$ is true.** This is usually easy, as in the example above. This part of the proof is called the *base case* or *basis step*.

4. **Prove that $P(n)$ implies $P(n+1)$ for every nonnegative integer n.** This is called the *inductive step*. The basic plan is always the same: assume that $P(n)$ is true and then use this assumption to prove that $P(n+1)$ is true. These two statements should be fairly similar, but bridging the gap may require some ingenuity. Whatever argument you give must be valid for every nonnegative integer n, since the goal is to prove the implications $P(0) \rightarrow P(1)$, $P(1) \rightarrow P(2)$, $P(2) \rightarrow P(3)$, etc. all at once.

5. **Invoke induction.** Given these facts, the induction principle allows you to

conclude that $P(n)$ is true for all nonnegative n. This is the logical capstone to the whole argument, but it is so standard that it's usual not to mention it explicitly.

Always be sure to explicitly label the base case and inductive step. It will make your proofs clearer, and it will decrease the chance that you forget a key step (such as checking the base case).

~~3.3.3~~ 3.2.4 A Clean Writeup

The proof of Theorem 3.3.1 given above is perfectly valid; however, it contains a lot of extraneous explanation that you won't usually see in induction proofs. The writeup below is closer to what you might see in print and should be prepared to produce yourself.

Proof of Theorem 3.3.1. We use induction. The induction hypothesis, $P(n)$, will be equation (3.2).

Base case: $P(0)$ is true, because both sides of equation (3.2) equal zero when $n = 0$.

Inductive step: Assume that $P(n)$ is true, where n is any nonnegative integer.

Then

$$1+2+3+\cdots+n+(n+1) = \frac{n(n+1)}{2} + (n+1) \quad \text{(by induction hypothesis)}$$

$$= \frac{(n+1)(n+2)}{2} \quad \text{(by simple algebra)}$$

which proves $P(n+1)$.

So it follows by induction that $P(n)$ is true for all nonnegative n. ■

Induction was helpful for *proving the correctness* of this summation formula, but not helpful for *discovering* it in the first place. Tricks and methods for finding such formulas will appear in a later chapter. be covered in Part III of the text.

3.3.4 Courtyard Tiling

3.2.5 A more challenging Example

During the development of MIT's famous Stata Center, costs rose further and further as budget, and there were some radical fundraising ideas. One rumored plan was to install a big courtyard with dimensions $2^n \times 2^n$ (as shown in Figure 1 for the case where $n = 3$) and to have one

(who we will refer to as "Bill", for the purposes of preserving anonymity).

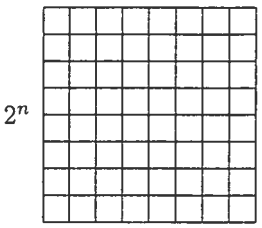


Figure 1: A $2^n \times 2^n$ courtyard for $n = 3$.

continued from prior page

One of the central squares would be occupied by a statue of a wealthy potential donor. Let's call him "Bill". In the special case $n = 0$, the whole courtyard consists of a single central square; otherwise, there are four central squares. A complication was that the building's unconventional architect, Frank Gehry, was alleged to require that only special L-shaped tiles be used (shown in Figure 2) for the courtyards.

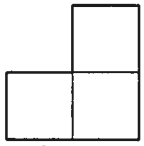


Figure 2: The special L-shaped tile.

It was quickly determined that a courtyard meeting these constraints exists, at least for $n = 2$ (see Figure 3.) But what about for large values of n? Is

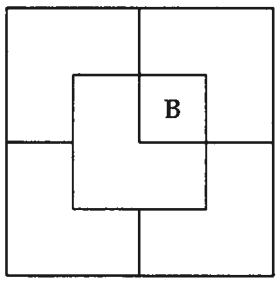


Figure 3: A tiling using L-shaped tiles for $n = 2$ with Bill in a central square.

For larger values of n, is there a way to tile a $2^n \times 2^n$ courtyard with L-shaped tiles and a statue in the center? Let's try to prove that this is so.

Theorem 3.3.2. *For all $n \geq 0$ there exists a tiling of a $2^n \times 2^n$ courtyard with Bill in a central square.*

Proof. (doomed attempt) The proof is by induction. Let $P(n)$ be the proposition that there exists a tiling of a $2^n \times 2^n$ courtyard with Bill in the center.

Base case: $P(0)$ is true because Bill fills the whole courtyard.

Inductive step: Assume that there is a tiling of a $2^n \times 2^n$ courtyard with Bill in the center for some $n \geq 0$. We must prove that there is a way to tile a $2^{n+1} \times 2^{n+1}$ courtyard with Bill in the center ....

Now we're in trouble! The ability to tile a smaller courtyard with Bill in the center isn't much help in tiling a larger courtyard with Bill in the center. We haven't figured out how to bridge the gap between $P(n)$ and $P(n+1)$.

So if we're going to prove Theorem 3.3.2 by induction, we're going to need some *other* induction hypothesis than simply the statement about n that we're try-

ing to prove.

When this happens, your first fallback should be to look for a *stronger* induction hypothesis; that is, one which implies your previous hypothesis. For example, we could make $P(n)$ the proposition that for *every* location of Bill in a $2^n \times 2^n$ courtyard, there exists a tiling of the remainder.

This advice may sound bizarre: "If you can't prove something, try to prove something grander!" But for induction arguments, this makes sense. In the inductive step, where you have to prove $P(n)$ IMPLIES $P(n+1)$, you're in better shape because you can *assume* $P(n)$, which is now a more powerful statement. Let's see how this plays out in the case of courtyard tiling.

Proof. (successful attempt) The proof is by induction. Let $P(n)$ be the proposition that for every location of Bill in a $2^n \times 2^n$ courtyard, there exists a tiling of the remainder.

Base case: $P(0)$ is true because Bill fills the whole courtyard.

Inductive step: Assume that $P(n)$ is true for some $n \geq 0$; that is, for every

location of Bill in a $2^n \times 2^n$ courtyard, there exists a tiling of the remainder. Divide the $2^{n+1} \times 2^{n+1}$ courtyard into four quadrants, each $2^n \times 2^n$. One quadrant contains Bill (**B** in the diagram below). Place a temporary Bill (**X** in the diagram) in each of the three central squares lying outside this quadrant as shown in Figure 4.

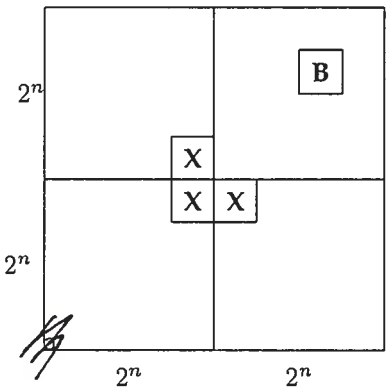


Figure 4: using a stronger inductive hypothesis to prove Theorem 3.3.2.

Now we can tile each of the four quadrants by the induction assumption. Replacing the three temporary Bills with a single L-shaped tile completes the job. This proves that $P(n)$ implies $P(n+1)$ for all $n \geq 0$. Thus $P(n)$ is true for all $n \in \mathbb{N}$. The theorem follows as a special case where we put Bill in a central square. ■

This proof has two nice properties. First, not only does the argument guarantee that a tiling exists, but also it gives an algorithm for finding such a tiling. Second,

we have a stronger result: if Bill wanted a statue on the edge of the courtyard, away from the pigeons, we could accommodate him!

Strengthening the induction hypothesis is often a good move when an induction proof won't go through. But keep in mind that the stronger assertion must actually be *true*; otherwise, there isn't much hope of constructing a valid proof! Sometimes finding just the right induction hypothesis requires trial, error, and insight. For example, mathematicians spent almost twenty years trying to prove or disprove the conjecture that "Every planar graph is 5-choosable"[3]. Then, in 1994, Carsten Thomassen gave an induction proof simple enough to explain on a napkin. The key turned out to be finding an extremely clever induction hypothesis; with that in hand, completing the argument was easy!

[3]5-choosability is a slight generalization of 5-colorability. Although every planar graph is 4-colorable and therefore 5-colorable, not every planar graph is 4-choosable. If this all sounds like nonsense, don't panic. We'll discuss graphs, planarity, and coloring in Part II of the text.

3.2.6

3.3.5 A Faulty Induction Proof

Insert D3 goes here

False Theorem. *All horses are the same color.*

Notice that no n is mentioned in this assertion, so we're going to have to reformulate it in a way that makes an n explicit. In particular, we'll (falsely) prove that

False Theorem 3.3.3. *In every set of $n \geq 1$ horses, all are the same color.*

This a statement about all integers $n \geq 1$ rather ≥ 0, so it's natural to use a slight variation on induction: prove $P(1)$ in the base case and then prove that $P(n)$ implies $P(n+1)$ for all $n \geq 1$ in the inductive step. This is a perfectly valid variant of induction and is *not* the problem with the proof below.

False proof. The proof is by induction on n. The induction hypothesis, $P(n)$, will be

$$\text{In every set of } n \text{ horses, all are the same color.} \tag{3.4}$$

Base case: ($n = 1$). $P(1)$ is true, because in a set of horses of size 1, there's only

Insert D3

~~When you are~~

If we have done a good job in writing this text, right about now, you should be thinking "Hey, this induction stuff isn't so hard after all – just show P(0) is true and that P(n) ~~→~~ implies P(n+1) for any ~~integer~~ number n." ~~End of~~ ~~it is actually~~ And, you would be right, although sometimes when you start doing induction proofs on your own, you can run into trouble. For example, we will now attempt to ruin your day by using induction to "prove" that all horses are the same color. And just when you thought it was safe to ~~go back~~ skip class and ~~head to the pub~~ party work on your ~~PSO~~ robot program instead. Bummer!

one horse, and this horse is definitely the same color as itself.

Inductive step: Assume that $P(n)$ is true for some $n \geq 1$. That is, assume that in every set of n horses, all are the same color. Now consider a set of $n+1$ horses:

$$h_1,\ h_2,\ \ldots,\ h_n,\ h_{n+1}$$

By our assumption, the first n horses are the same color:

$$\underbrace{h_1,\ h_2,\ \ldots,\ h_n,}_{\text{same color}}\ h_{n+1}$$

Also by our assumption, the last n horses are the same color:

$$h_1,\ \underbrace{h_2,\ \ldots,\ h_n,\ h_{n+1}}_{\text{same color}}$$

So h_1 is the same color as the remaining horses besides h_{n+1}, and likewise h_{n+1} is the same color as the remaining horses besides h_1. So h_1 and h_{n+1} are the same color. That is, horses $h_1, h_2, \ldots, h_{n+1}$ must all be the same color, and so $P(n+1)$ is true. Thus, $P(n)$ implies $P(n+1)$.

By the principle of induction, $P(n)$ is true for all $n \geq 1$. ■

We've proved something false! Is math broken? Should we all become poets?

No, this proof has a mistake.

INSERT D6 goes here

The error in this argument is in the sentence that begins, "So h_1 and h_{n+1} are the same color." The "..." notation creates the impression that there are some remaining horses besides h_1 and h_{n+1}. However, this is not true when $n = 1$. In that case, the first set is just h_1 and the second is h_2, and there are no remaining horses besides them. So h_1 and h_2 need not be the same color!

This mistake knocks a critical link out of our induction argument. We proved $P(1)$ and we *correctly* proved $P(2) \longrightarrow P(3)$, $P(3) \longrightarrow P(4)$, etc. But we failed to prove $P(1) \longrightarrow P(2)$, and so everything falls apart: we can not conclude that $P(2)$, $P(3)$, etc., are true. And, of course, these propositions are all false; there are horses of a different color.

Students sometimes claim that the mistake in the proof is because $P(n)$ is false for $n \geq 2$, and the proof assumes something false, namely, $P(n)$, in order to prove $P(n+1)$. You should think about how to explain to such a student why this claim would get no credit on a 6.042 exam.

Insert D6

The first error in this argument is in the sentence that begins "So h_1 is the same color as the remaining horses besides h_{n+1} (i.e., $h_2, \ldots, h_n$)..."

3.3.6 Problems

Class Problems

Homework Problems

3.2.7

make this be a subsection

3.4 Induction versus Well Ordering

The Induction Rule looks nothing like the Well Ordering Principle, but these two proof methods are closely related. In fact, as the examples above suggest, we can take any Well Ordering proof and reformat it into an Induction proof. Conversely, it's equally easy to take any Induction proof and reformat it into a Well Ordering proof.

So what's the difference? Well, sometimes induction proofs are clearer because they resemble recursive procedures that reduce handling an input of size $n+1$ to handling one of size n. On the other hand, Well Ordering proofs sometimes seem more natural, and also come out slightly shorter. The choice of method is really a matter of style and is up to you.

3.3

3.5 Strong Induction

insert DS goes here

Strong Induction is a variation of ordinary Induction

A useful variant of induction is called *strong induction*. Strong Induction and Ordinary Induction that cannot is very useful for a number of proofs

nary Induction are used for exactly the same things: proving that a predicate $P(n)$ is true for all $n \in \mathbb{N}$.

new subsection

3.3.1 A Rule for Strong Induction

Principle of Strong Induction. Let $P(n)$ be a predicate. If

- $P(0)$ is true, and
- for all $n \in \mathbb{N}$, $P(0)$, $P(1)$, ..., $P(n)$ *together* imply $P(n+1)$,

then $P(n)$ is true for all $n \in \mathbb{N}$.

The only change from the ordinary induction principle is that strong induction allows you to assume more stuff in the inductive step of your proof! In an ordinary induction argument, you assume that $P(n)$ is true and try to prove that $P(n+1)$ is also true. In a strong induction argument, you may assume that $P(0)$, $P(1)$, ..., and $P(n)$ are *all* true when you go to prove $P(n+1)$. These extra assumptions can

Insert D 8

Strong induction is a variation of ordinary induction that is useful when the predicate P(n+1) naturally depends on P(a) for values of $a < n$. As with ordinary induction, strong induction is used to prove

only make your job easier. Hence the name : strong induction.

INSERT D9 goes here

~~3.3.1 An Example~~

3.5.1 ~~Products of Primes~~ 3.3.2 Some Examples: Products of Primes

subsubsection w/o number

As a first example, we'll use strong induction to re-prove Theorem 3.1.1 which we previously proved using Well Ordering.

Lemma 3.5.1. *Every integer greater than 1 is a product of primes.*

Proof. We will prove Lemma 3.5.1 by strong induction, letting the induction hypothesis, $P(n)$, be

$$n \text{ is a product of primes.}$$

So Lemma 3.5.1 will follow if we prove that $P(n)$ holds for all $n \geq 2$.

Base Case: ($n = 2$) $P(2)$ is true because 2 is prime, and so it is a length one product of primes by convention.

Inductive step: Suppose that $n \geq 2$ and that i is a product of primes for every integer i where $2 \leq i < n + 1$. We must show that $P(n + 1)$ holds, namely, that $n + 1$ is also a product of primes. We argue by cases:

INSERT D9

Formulated as a proof rule, strong induction ~~would be~~ is:

Rule: Strong Induction Rule

~~$P(0), \forall n \in \mathbb{N}. (P(0) \wedge P(1) \wedge \ldots \wedge P(n)) \Rightarrow P(n+1)$~~

$$P(0),\ \forall n \in \mathbb{N}.\ (P(0) \wedge P(1) \wedge \ldots \wedge P(n)) \Rightarrow P(n+1)$$

$$\forall m \in \mathbb{N},\ P(m)$$

The template for strong induction proofs is identical to the template given in Section 3.2.3 except for two things:

- you should state that your proof is by strong induction, and
- you can assume that $P(0), P(1), \ldots, P(n)$ are all true instead of only $P(n)$ during the inductive step.

If $n+1$ is itself prime, then it is a length one product of primes by convention, and so $P(n+1)$ holds in this case.

Otherwise, $n+1$ is not prime, which by definition means $n+1 = km$ for some integers k, m such that $2 \leq k, m < n+1$. Now by the strong induction hypothesis, we know that k is a product of primes. Likewise, m is a product of primes. It follows immediately that $km = n$ is also a product of primes. Therefore, $P(n+1)$ holds in this case as well.

So $P(n+1)$ holds in any case, which completes the proof by strong induction that $P(n)$ holds for all ~~nonnegative integers, n~~ $n \geq 2$.

■

make it be a subsubsection w/o a number

3.5.2 Making Change

The country Inductia, whose unit of currency is the Strong, has coins worth 3Sg (3 Strongs) and 5Sg. Although the Inductians have some trouble making small change like 4Sg or 7Sg, it turns out that they can collect coins to make change for any number that is at least 8 Strongs.

Strong induction makes this easy to prove for $n+1 \geq 11$, because then $(n+1)-3 \geq 8$, so by strong induction the Inductians can make change for exactly $(n+1)-3$ Strongs, and then they can add a 3Sg coin to get $(n+1)$Sg. So the only thing to do is check that they can make change for all the amounts from 8 to 10Sg, which is not too hard to do.

Here's a detailed writeup using the official format:

Proof. We prove by strong induction that the Inductians can make change for any amount of at least 8Sg. The induction hypothesis, $P(n)$ will be:

If $n \geq 8$, then there is a collection of coins whose value is n Strongs.

Notice that $P(n)$ is an implication. When the hypothesis of an implication is false, we know the whole implication is true. In this situation, the implication is said to be *vacuously* true. So $P(n)$ will be vacuously true whenever $n < 8$.[4]

[4] Another approach that avoids these vacuous cases is to define

$$Q(n) ::= \text{there is a collection of coins whose value is } n + 8\text{Sg},$$

and prove that $Q(n)$ holds for all $n \geq 0$.

We now proceed with the induction proof:

Base case: $P(0)$ is vacuously true because a 3Sg coin together with an 5Sg coin makes 8Sg.

Inductive step: We assume $P(m)$ holds for all $m \leq n$, and prove that $P(n+1)$ holds. We argue by cases:

Case $(n+1 < 8)$: $P(n+1)$ is vacuously true in this case.

Case $(n+1 = 8)$: $P(8)$ holds because the Inductians can use one 3Sg coin and one 5Sg coin.

Case $(n+1 = 9)$: Use three 3Sg coins. we have to make $n+1+8 = 9$ Sg. We can do this using

Case $(n+1 = 10)$: Use two 5Sg coins. we have to make $n+1+8 = 10$ Sg. We can do this using

Case $(n+1 \geq 11)$: Then $0 \leq n-2 \leq n$, so by the strong induction hypothesis, the Inductians can make change for $n-2$ Strong. Now by adding a 3Sg coin, they can make change for $(n+1)$Sg.

So in any case, $P(n+1)$ is true, and we conclude by strong induction that for all $n \geq 0$, the Inductians can make change for $n+8$ Strong. That is, they can make change for any number of 8 or more Strong. ■

make into a sub sub section without a number

3.5.3 The Stacking Game

Here is another exciting 6.042 game that's surely about to sweep the nation!

You begin with a stack of n boxes. Then you make a sequence of moves. In each move, you divide one stack of boxes into two nonempty stacks. The game ends when you have n stacks, each containing a single box. You earn points for each move; in particular, if you divide one stack of height $a + b$ into two stacks with heights a and b, then you score ab points for that move. Your overall score is the sum of the points that you earn for each move. What strategy should you use to maximize your total score?

As an example, suppose that we begin with a stack of $n = 10$ boxes. Then the game might proceed as follows: shown in Figure A. Can you find a better strategy?

Stack Heights										Score
<u>10</u>										
5	<u>5</u>									25 points
<u>5</u>	3	2								6
<u>4</u>	3	2	1							4
2	<u>3</u>	2	1	2						4
<u>2</u>	2	2	1	2	1					2
1	<u>2</u>	2	1	2	1	1				1
1	1	<u>2</u>	1	2	1	1	1			1
1	1	1	1	<u>2</u>	1	1	1	1		1
1	1	1	1	1	1	1	1	1	1	1
								Total Score	=	45 points

Figure A: An example of the stacking game with $n = 10$ boxes. On each line, the underlined stack is divided in the next step.

On each line, the underlined stack is divided in the next step. Can you find a better strategy?

Analyzing the Game

Let's use strong induction to analyze the unstacking game. We'll prove that your score is determined entirely by the number of boxes —your strategy is irrelevant!

Theorem 3.5.2. *Every way of unstacking n blocks gives a score of $n(n-1)/2$ points.*

There are a couple technical points to notice in the proof:

- The template for a strong induction proof is exactly the same as for ordinary induction.

- As with ordinary induction, we have some freedom to adjust indices. In this case, we prove $P(1)$ in the base case and prove that $P(1), \ldots, P(n)$ imply $P(n+1)$ for all $n \geq 1$ in the inductive step.

Proof. The proof is by strong induction. Let $P(n)$ be the proposition that every way of unstacking n blocks gives a score of $n(n-1)/2$.

Base case: If $n = 1$, then there is only one block. No moves are possible, and so the total score for the game is $1(1-1)/2 = 0$. Therefore, $P(1)$ is true.

Inductive step: Now we must show that $P(1), \ldots, P(n)$ imply $P(n+1)$ for all $n \geq 1$. So assume that $P(1), \ldots, P(n)$ are all true and that we have a stack of $n+1$ blocks. The first move must split this stack into substacks with positive sizes a and b where $a + b = n + 1$ and $0 < a, b \leq n$. Now the total score for the game is the sum of points for this first move plus points obtained by unstacking the two resulting substacks:

$$
\begin{aligned}
\text{total score} &= (\text{score for 1st move}) \\
&\quad + (\text{score for unstacking } a \text{ blocks}) \\
&\quad + (\text{score for unstacking } b \text{ blocks}) \\
&= ab + \frac{a(a-1)}{2} + \frac{b(b-1)}{2} && \text{by } P(a) \text{ and } P(b) \\
&= \frac{(a+b)^2 - (a+b)}{2} = \frac{(a+b)((a+b)-1)}{2} \\
&= \frac{(n+1)n}{2}
\end{aligned}
$$

This shows that $P(1), P(2), \ldots, P(n)$ imply $P(n+1)$.

New subsection

Therefore, the claim is true by strong induction. ■

3.3.3 Strong Induction versus Ordinary Induction

Despite the name, strong induction is technically no more powerful than ordinary induction, though it makes some proofs easier to follow. But any theorem that can be proved with strong induction could also be proved with ordinary induction (using a slightly more complicated induction hypothesis). On the other hand, announcing that a proof uses ordinary rather than strong induction highlights the fact that $P(n+1)$ follows directly from $P(n)$, which is generally good to know.

3.5.4 Problems

INSERT EO goes here then followed by INSERT E1 which is a new section:

3.4 Proof by Invariants

Then it is followed by

3.5 Problems

Class Problems

INSERT

Is strong induction really stronger than ordinary induction? It certainly looks that way. After all, you can assume alot more when proving the induction step. But actually, ~~there is no difference~~ anything that can be proved ~~whatever you can prove~~ with strong induction can also be proved with ordinary induction — you just need to use a "stronger" induction hypothesis.

Which method should you use? ~~Use~~ whichever you find easier. But ~~be sure~~ whichever method you choose, be sure to state the method ~~to be say which method you are using~~ up front so that the reader can understand and more easily verify your proof.

INSERT E1

3.4 ~~Proof by~~ Invariants

One of the most important uses of induction in computer science ~~is to~~ involves proving that a program or process preserves ~~maintains~~ ~~a property for set of properties~~ desirable as it one or more desirable properties ~~of~~ A proceeds. A property that is preserved ~~by~~ through ~~any~~ a series of operations or steps is known as an invariant. ~~An~~ Examples of desirable invariants could include properties ~~propositions~~ such as "a variable never exceeding a certain value or becoming negative, the altitude of a plane never dropping below 1,000 feet without the wing flaps and landing gear being deployed, and the temperature of a nuclear reactor never exceeding ~~entering~~ ~~500 degrees~~ the threshold for a meltdown.

E1-2

We typically use induction to prove that a proposition is an invariant. In particular, we show that the proposition is true at the beginning (this is the base case) and that if it is true after t steps have been taken, it will also be true after step $t+1$ (this is the inductive step). We can then use the induction principle to conclude that the proposition is indeed an invariant, i.e., that it will always hold.

3.4.1 A Simple Example - the Diagonally Moving Robot

~~Not surprisingly, we will be seeing 3.4.1 Establishing that a Proposition is Lots of an Invariant~~

Invariants are useful in systems that have a start state or configuration

and a well-defined series of steps during which the system ~~can~~ can change state.[1] ~~Such systems can~~ For example, suppose that you have a robot that can walk across diagonals on an infinite 2-dimension grid. The robot starts at ~~position~~ position (0,0) and at each step it moves up ~~or down~~ down (but not both) by 1 unit vertically and left or right (but not both) by 1 unit horizontally. To be clear, ~~Note that~~ the robot must move by exactly 1 unit in each dimension ~~each~~ during each step since it can only traverse diagonals.

---

[1] Such systems are known as state machines and we will ~~discuss~~ study them in greater ~~depth~~ detail in Chapter ??.

In this example, the state of the robot ~~after~~ at any time can be specified by a coordinate pair (x, y) that denotes the robot's position. The start state is $(0, 0)$ since it is given that the robot ~~is~~ starts at that position. After the first step, the robot could be in states $(1, 1)$, $(1, -1)$, $(-1, 1)$ or $(-1, -1)$. After two steps, there are 9 possible states for the robot, including $(0, 0)$.

Can the robot ever reach position $(1, 0)$?

After playing around with the robot for a bit, it will become apparent that the robot will never be able to reach

position (1, 0). This is because the robot can only reach positions (x, y) for which x+y is even. This crucial observation quickly leads to the formulation of a predicate

P(t) :: if the robot is in state (x, y) after t steps, then x+y is even,

which can be proved to be an invariant by induction.

Theorem: The sum of robot's coordinates is always even.

Proof: We will prove that P(t) is an invariant by induction.

P(0) is true since the robot starts at (0,0) and 0+0 is even.

E1-6

Assume that P(t) is true for the

~~to show P(t) IMPLIES P(t+1)~~

inductive step. Let (x, y) be the position of the robot after t steps. Since P(t) is assumed to be true, we know that x+y is even. There are four cases to consider for step t+1 depending on which direction the robot moves.

Case 1: The robot moves to (x+1, y+1). Then the sum of the coordinates is x+y+2, which is even, and so P(t+1) is true.

Case 2: The robot move to (x+1, y-1). Then the sum of the coordinates is x+y, which is even, and so P(t+1) is true.

Case 3: The robot moves to $(x-1, y+1)$. Then the sum of the coordinates is $x+y$, as with Case 2, and $P(t+1)$ is true.

Case 4: The robot moves to $(x-1, y-1)$. Then the sum of the coordinates is $x+y-2$, which is even, and so $P(t+1)$ is true.

In every case, $P(t+1)$ is true and so we have proved $P(t)$ IMPLIES $P(t+1)$ and so by induction, we know that $P(t)$ is true for all $t \geq 0$. #

Corollary 1: The robot can never reach position $(1, 0)$.

Proof: By Theorem T1, we know the robot can only reach positions

with coordinates that sum to an even number, and thus it cannot reach position (1,0). ∎

Since this was the first time we ~~did a proof by~~ predicate ~~proved~~ proved that a ~~property~~ was an invariant, we were careful to go through all four cases in gory detail. As you become more experienced with such proofs, you will likely become more brief as well. ~~In this case,~~ Indeed, if we were going through this proof again at a later point in the text, we might simply note that ~~the parity of the sum of the coordinates~~ the sum of the coordinates after ~~steps is x+y~~ step t+1 can be only x+y, x+y+2 or x+y-2 and therefore that it is even.

3.4.2 The Invariant Method

In summary, if you would like to prove that some property NICE holds for every step of a process, then it is often helpful to use the following method:

- Define P(t) to be the predicate that NICE holds immediately after step t.
- Show that P(0) is true, namely that NICE holds for the start state.
- Show that

$$\forall t \in \mathbb{N}.\ P(t) \text{ IMPLIES } P(t+1),$$

namely that for any $t \geq 0$, if NICE holds immediately after step t, then it must also hold after the following step.

E1-10

more Challenging

3.4.3 A ~~Harder~~ Example — the 15-Puzzle

In the late 19th Century, Noyes Chapman, ~~the~~ a postmaster in Canastota, New York, invented the 15-puzzle[1], ~~which~~ which consisted of a 4×4 numbered grid containing 15 blocks ~~labelled 1 to 15~~ numbered ~~1 to 15~~ in which the 14-block and the 15-block were out of order. The objective was to move the blocks one at a time into an adjacent hole in the grid so as to eventually get all 15 blocks into their natural order. A picture of the 15-puzzle is shown in Figure F1 along with the configuration after the 12-block is moved into the hole below. The ~~final~~ desired final configuration

1 Actually, there is a dispute about who really invented the 15-puzzle. Sam Lloyd, a well known puzzle designer, claimed to be the inventor, but this has since been discredited.

is shown in Figure F2.

E1-11

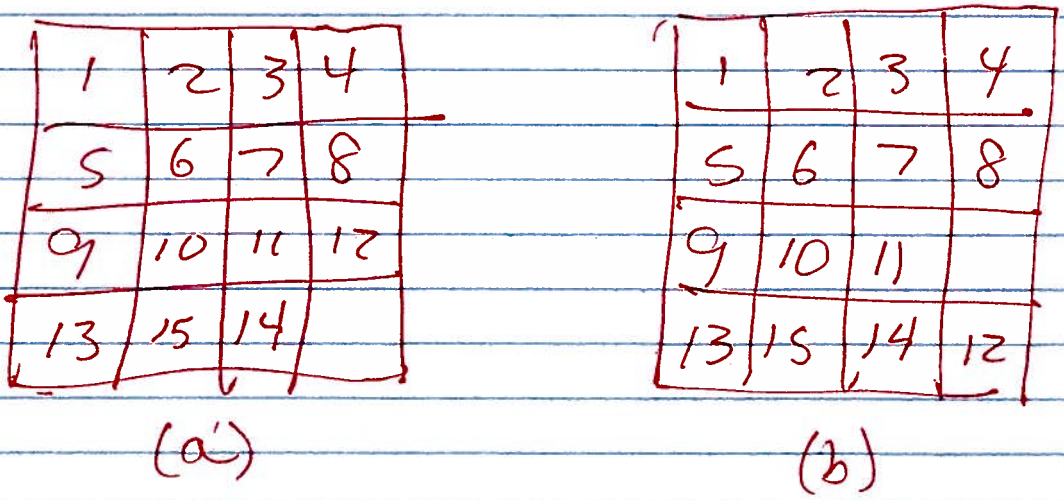


Figure F1 The 15-puzzle in its starting configuration (a) and after the 12-block is moved into the hole below (b).

~~Exercise~~

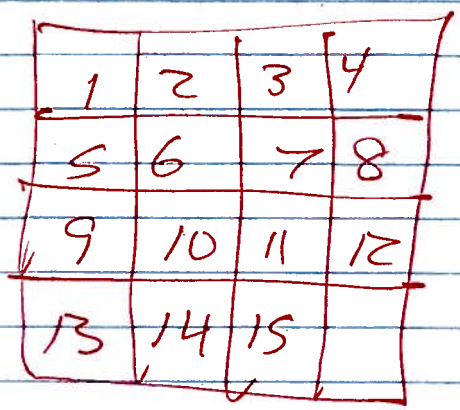


Figure F2: The desired final configuration for the 15-puzzle. Can it be achieved by only moving one block at a time into an adjacent hole?

The 15-puzzle became very popular in North America and Europe and is still sold in game and puzzle shops today. Prizes were offered for its solution, but it is doubtful that they were ever awarded, since it is impossible to get from the configuration in Figure F1 to the configuration in Figure F2 by only moving one block at a time into an adjacent hole. The proof of this fact is a little tricky so we have left it for you to figure out on your own. Instead, we will prove that the analogous task for the much easier 8-puzzle cannot be performed. Both proofs, of course, make use of the invariant method.

Structure long proofs. Long program are usually broken into a heirarchy of smaller procedures. Long proofs are much the same. Facts needed in your proof that are easily stated, but not readily proved are best pulled out and proved in a preliminary lemmas. Also, if you are repeating essentially the same argument over and over, try to capture that argument in a general lemma and then repeatedly cite that instead.

Don't bully. Words such as "clearly" and "obviously" serve no logical function. Rather, they almost always signal an attempt to bully the reader into accepting something which the author is having trouble justifying rigorously. Don't use these words in your own proofs and go on the alert whenever you read one.

Finish. At some point in a proof, you'll have established all the essential facts you need. Resist the temptation to quit and leave the reader to draw the right conclusions. Instead, tie everything together yourself and explain why the original claim follows.

The analogy between good proofs and good programs extends beyond structure. The same rigorous thinking needed for proofs is essential in the design of critical computer system. When algorithms and protocols only "mostly work" due to reliance on hand-waving arguments, the results can range from problematic to catastrophic. An early example was the Therac 25, a machine that provided radiation therapy to cancer victims, but occasionally killed them with massive overdoses due to a software race condition. More recently, a buggy electronic voting system credited presidential candidate Al Gore with *negative* 16,022 votes in one county. In August 2004, a single faulty command to a computer system used by United and American Airlines grounded the entire fleet of both companies— and all their passengers.

It is a certainty that we'll all one day be at the mercy of critical computer systems designed by you and your classmates. So we really hope that you'll develop the ability to formulate rock-solid logical arguments that a system actually does what you think it does.

2 The 8 Puzzle

In the 8-Puzzle, there are 8 lettered tiles and a blank square arranged in a 3 × 3 grid. Any lettered tile adjacent to the blank square can be slid into the blank. For example, a sequence of two moves is illustrated below.

A	B	C
D	E	F
H	G	

(a) →

A	B	C
D	E	F
H		G

(b) →

A	B	C
D		F
H	E	G

(c)

We'll only be considering legal moves in this lecture.

In the leftmost configuration shown above, the G and H tiles are out of order. We can find a way of swapping G and H so that they are in the right order, but then other letters

may be out of order. Can you find a sequence of moves that puts these two letters in the correct order, but returns every other tile to its original position? Some experimentation suggests that the answer is probably "no", ~~so let's try to prove that.~~ and we will prove that is so by finding an invariant (i.e.)

~~We're going to take an approach that is frequently used in the analysis of software and systems. We'll look for an *invariant*,~~ a property of the puzzle that is always maintained, no matter how you move the tiles around. If we can then show that putting the ~~G and H~~ all the tiles in the correct order would violate the invariant, then we can conclude that ~~this is impossible.~~ the puzzle cannot be solved.

~~Let's see how this game plan plays out. Here is the theorem we're trying to prove:~~

Theorem 1. *No sequence of legal moves transforms the ~~board below on the left~~ into the ~~board below on the right~~.*

configuration in Figure F4. configuration in Figure F3(a)

A	B	C
D	E	F
~~H~~	G	

A	B	C
D	E	F
G	H	

Figure F4: The desired final configuration of the 8-puzzle

We'll build up a sequence of observations, stated as lemmas. Once we achieve a critical mass, we'll assemble these observations into a complete proof of Theorem 1.

Define a ***row move*** as a move in which a tile slides horizontally and a ***column move*** as one in which a tile slides vertically. Assume that tiles are read top-to-bottom and left-to-right like English text, that is, the *natural order*, defined as follows:

1	2	3
4	5	6
7	8	9

So when we say two tiles are "out of order", we mean that the larger letter precedes the smaller letter in this natural order.

Our difficulty is that one pair of tiles (the G and H) is out of order initially. An immediate observation is that row moves alone are of little value in addressing this problem:

Lemma 2. *A row move does not change the order of the tiles.*

Proof. A row move moves a tile from cell i to cell $i + 1$ or vice versa. This tile does not change its order with respect to any other tile. Since no other tile moves, there is no change in the order of any of the other pairs of tiles. □

Let's turn to column moves. This is the more interesting case, since here the order can change. For example, the column move shown ~~below changes~~ the relative order of the pairs (G, H) and (G, E). in Figure F5 changes

A	B	C
D	F	
H	E	G

→

A	B	C
D	F	G
H	E	

Figure F5: An example of a column move in which the ~~G-block~~ G-tile is moved ~~upward~~ into the adjacent hole above. In this case, G changes order with E and H.

E1 -15

Lemma 3. *A column move changes the relative order of exactly two pairs of tiles.*

Proof. Sliding a tile down moves it after the next two tiles in the order. Sliding a tile up moves it before the previous two tiles in the order. Either way, the relative order changes between the moved tile and each of the two it crosses. The relative order between any other pair of tiles does not change. □

These observations suggest that there are limitations on how tiles can be swapped. Some such limitation may lead to the invariant we need. In order to reason about swaps more precisely, let's define a term referring to a pair of items that are out of order:

Definition 1. *A pair of letters L_1 and L_2 is an **inversion** if L_1 precedes L_2 in the alphabet, but L_1 appears after L_2 in the puzzle order.*

For example, in the puzzle below, there are three inversions: (D, F), (E , F) and (G, E).

A	*B*	*C*
F	*D*	*G*
E	*H*	

There is exactly one inversion (G,H) in the start state:

A	*B*	*C*
D	*E*	*F*
H	**G**	

There are no inversions in the end state:

A	*B*	*C*
D	*E*	*F*
G	**H**	

Let's work out the effects of row and column moves in terms of inversions.

Lemma 4. *During a move, the number of inversions can only increase by 2, decrease by 2 or remain the same.*

Proof. By Lemma 2, a row move does not change the order of the tiles; thus, in particular, a row move does not change the number of inversions.

By Lemma 3, a column move changes the relative order of exactly 2 pairs of tiles. There are three cases: If both pairs were originally in order, then the number of inversions after the move goes up by 2. If both pairs were originally inverted, then the number of inversions after the move goes down by 2. If one pair was originally inverted, and the other was originally in order, then the number of inversions stays the same (since the changing the former pair makes the number of inversions smaller by 1, and changing the latter pair makes the number of inversions larger by 1). □

We are almost there. If the number of inversions only change by 2, then what about the parity? That is, the "parity" of a number refers to whether the number is even or odd. For example, 7 and 5 have odd parity, and 18 and 0 have even parity.

Since adding or subtracting 2 from a number does not change the parity, we have the following corollary:

Corollary 5. *Neither a row nor a column move ever changes the parity of the number of inversions.*

Now we can bundle up all these observations and state an *invariant*, that is, a property of the puzzle that never changes, no matter how you slide the tiles around.

Lemma 6. *In every configuration reachable from the position shown below, the* parity of the number of inversions is odd.

Proof. We use induction. Let $P(n)$ be the proposition that after n moves from the above configuration, the parity of the number of inversions is odd.

Base case: After zero moves, exactly one pair of tiles is inverted (H and G), which is an odd number. Therefore, $P(0)$ is true.

Inductive step: Now we must prove that $P(n)$ implies $P(n+1)$ for all $n \geq 0$. So assume that $P(n)$ is true; that is, after n moves the parity of the number of inversions is odd. Consider any sequence of $n+1$ moves $m_1, \ldots, m_{n+1}$. By the induction hypothesis $P(n)$, we know that the parity after moves $m_1, \ldots, m_n$ is odd. By Corollary 5, we know that the parity does not change during m_{n+1}. Therefore, the parity of the number of inversions after moves $m_1, \ldots, m_{n+1}$ is odd, so we have that $P(n+1)$ is true.

Thus, $P(n)$ implies $P(n+1)$ for all $n \geq 0$.

By the principle of induction, $P(n)$ is true for all $n \geq 0$. □

The theorem we originally set out to prove is restated below. With this invariant in hand, the proof is simple.

Theorem. *No sequence of moves transforms the board below on the left into the board below on the right.*

A	B	C
D	E	G
H	**G**	

A	B	C
D	E	F
G	**H**	

Proof. In the target configuration on the right, the total number of inversions is zero, which is even. Therefore, by Lemma 6, the target configuration is unreachable. □

3.5 Problems

Chapter 4

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 expressed pleasure in its impracticality when 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 or-

dinary 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, check your grades on WebSIS, or use a PayPal account, you are relying on number theoretic algorithms.

4.1 Divisibility

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

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

$$a \text{ divides } b \quad \text{iff} \quad ak = b \text{ for some } k.$$

The notation, $a \mid b$, is an abbreviation for "a divides b." If $a \mid b$, then we also say that

Insert F1

Number theory also provides an environment excellent ~~[illegible]~~ for us to practice and apply ~~most all of~~ the proof techniques that we ~~have~~ developed in Chapters 2 and 3.

b is a *multiple* of a. A consequence of this definition is that every number divides zero.

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.[1] Interestingly, we'll see that computer scientists have found ways to turn some of these difficulties to their advantage.

[1] text from top of next page goes here in footnote

Don't Panic —we're going to stick to some relatively benign parts of number theory. We rarely put any of these super-hard unsolved problems ~~on exams~~ in the HW problem sets.

4.1.1 Facts About Divisibility

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

Lemma 4.1.1. *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 prove only part 2.; the other proofs are similar.

Assume a|b and b|c.

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 $(ak_1)k_2 = c$. So $a(k_1k_2) = c$, which implies that $a \mid c$. ∎

Fix spacing.

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. Because of its special properties, the number 1 is considered to be neither prime nor composite.

fit to a single page

Famous Conjectures in Number Theory

Fermat's Last Theorem There are no 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 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. 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 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

I recall that a prime is a number greater than 1 that is divisible only by itself and 1.

4.1.2 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 4.1.2 (Division Theorem). [1] *Let n and d be integers such that $d > 0$. Then there exists a unique pair of integers q and r, such that*

$$n = q \cdot d + r \text{ AND } 0 \leq r < d. \tag{4.1}$$

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, the expression "32 % 5" evaluates to 2 in Java, C, and C++. However, all these languages treat

[1] This theorem is often called the "Division Algorithm," even though it is not what we would call an algorithm. we will take this familiar result for granted without proof.

negative numbers strangely.

We'll take this familiar Division Theorem for granted without proof.

4.1.3 Die Hard

— put insert 62 here —

We've previously looked at the Die Hard water jug problem with jugs of sizes 3 and 5, and 3 and 9. A little number theory lets us solve all these silly water jug questions at once. In particular, it will be easy to figure out exactly which amounts of water can be measured out using jugs with capacities a and b.

Finding an Invariant Property

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

~~Number Theory I~~

~~2 Die Hard~~

This is insert 0-2. It comes from lecture 4 in 6.042 in Fall 08

> **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?

The preceding script is ~~This is~~ from the movie *Die Hard 3: With a Vengeance*. In the movie, 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, we think the inner message of the film is that everyone should learn at least a little number theory.

~~3 Die Once and For All~~

Although there were no water jug tests in Die Hard 4, rumor has it that the jugs will return in future sequels:

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

5

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 6: 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 7: 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.

3.1 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.

$(0,0) \to (a,0)$	fill first jug
$\to (0,a)$	pour first into second
$\to (a,a)$	fill first jug
$\to (2a-b,b)$	pour first into second
$\to (2a-b,0)$	empty second jug
$\to (0,2a-b)$	pour first into second
$\to (a,2a-b)$	fill first
$\to (3a-2b,b)$	pour first into second

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

$$s \cdot a + t \cdot b$$

for some integers s and t. An expression of this form is called a ***linear combination*** of a and b. This sounds like an invariant assertion that we might be able to prove by induction!

Lemma 3. *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.*

Proof. We use induction. Our invariant $P(n)$ is the proposition that after n steps, the amount of water in each jug is a linear combination of a and b.

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

Inductive step. Now we must show that $P(n)$ implies $P(n+1)$ for $n \geq 0$. So assume that after n steps the amount of water in each jug is a linear combination of a and b. There are two cases:

enough:

$(0,0) \to (a,0)$	fill first jug
$\to (0,a)$	pour first into second
$\to (a,a)$	fill first jug
$\to (2a-b,b)$	pour first into second (assuming $2a \geq b$)
$\to (2a-b,0)$	empty second jug
$\to (0,2a-b)$	pour first into second
$\to (a,2a-b)$	fill first
$\to (3a-2b,b)$	pour first into second (assuming $3a \geq 2b$)

What leaps out is that at every step, the amount of water in each jug is of the form

$$s \cdot a + t \cdot b \tag{4.2}$$

for some integers s and t. An expression of the form (4.2) is called an *integer linear combination* of a and b, but in this chapter we'll just call it a *linear combination*, since we're only talking integers. So we're suggesting:

Lemma 4.1.3. *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.*

Lemma 4.1.3 is easy to prove by induction on the number of pourings.

Proof. The induction hypothesis, $P(n)$, is the proposition that after n steps, 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. We assume by induction hypothesis that after n steps the amount of water in each jug is a linear combination of a and b. 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 in each jug is a linear combina-

tion of a and b before we begin pouring:

$$j_1 = s_1 \cdot a + t_1 \cdot b$$

$$j_2 = s_2 \cdot a + t_2 \cdot b$$

After pouring, one jug is either empty (contains 0 gallons) or full (contains a or b gallons). Thus, the other jug contains either $j_1 + j_2$ gallons, $j_1 + j_2 - a$, or $j_1 + j_2 - b$ gallons, all of which are linear combinations of a and b. So $P(n+1)$ holds in this case as well.

So in any case, $P(n+1)$ follows, completing the proof by induction. ■

This theorem has an important corollary:

Corollary 4.1.4. *Bruce dies.*

Proof. In Die Hard 7, Bruce has water jugs with capacities 3 and 6 and must form 4 gallons of water. However, the amount in each jug is always of the form $3s + 6t$ by Lemma 4.1.3. This is always a multiple of 3 by Lemma 4.1.1.3, so he cannot measure out 4 gallons. ■

But Lemma 4.1.3 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, namely greatest common divisors, and these will help us solve the water jug problem.

4.2 The Greatest Common Divisor

We've already examined the Euclidean Algorithm for computing ged(a, b), the greatest common divisor of a and b. This quantity turns out to be a very valuable piece of information about the relationship between a and b. We'll be making arguments about greatest common divisors all the time.

4.2.1 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 4.2.1. *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. By the Well Ordering Principle, there is a smallest positive linear combination of a and b; call it m. 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 $\gcd(a, b) \leq m$. Now any common divisor of a and b —that is, any c such that $c \mid a$ and $c \mid b$ —will divide both sa and tb, and therefore also divides $sa + tb$. The $\gcd(a, b)$ is by definition a common divisor of a and b, so

every s and t. In particular, $\gcd(a, b) \mid m$, which implies that $\gcd(a, b) \leq m$.

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. Substituting in for m gives:

$$a = q \cdot (sa + tb) + r, \qquad \text{so}$$

$$r = (1 - qs)a + (-qt)b.$$

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 $m \mid a$. ■

Corollary 4.2.2. *An integer is linear combination of a and b iff it is a multiple of* $\gcd(a, b)$.

Proof. By (4.2), 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. ■

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

Corollary 4.2.3. *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 2 gallons using 12 and 899 gallon jugs, because 2 is not a multiple of $\gcd(12, 899) = 1$.

4.2.2 Properties of the Greatest Common Divisor

We'll often make use of some basic gcd facts:

Lemma 4.2.4. *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. $\gcd(a, b) = \gcd(b, \text{rem}(a, b))$.

Here's the trick to proving these statements: translate the gcd world to the linear combination world using Theorem 4.2.1, argue about linear combinations, and then translate back using Theorem 4.2.1 again.

Proof. We prove only parts 3. and 4.

Proof of 3. The assumptions together with Theorem 4.2.1 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 $a \cdot (asu + btu + csv) + bc(tv)$. This is a linear combination of a and bc that is equal to 1, so $\gcd(a, bc) = 1$ by Theorem 4.2.1.

Proof of 4. Theorem 4.2.1 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 $\gcd(ac, bc) = c \cdot \gcd(a, b) = c \cdot 1 = c$. The first equality uses part 2 of this lemma, and the second uses the assumption that $\gcd(a, b) = 1$. ■

— INSERT #63 goes here ——————

Lemma 4.2.4.5 is the preserved invariant from Lemma 8.1.7 that we used to prove partial correctness of the Euclidean Algorithm.

But what about Die Hard 5. Is it possible to make 3 gallons using 21 and 26-gallon jugs?

for Bruce

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 3 gallons can be formed. On the other hand, we don't know if it can be done either. To resolve the matter, we will need more number theory.

This is Insert 6-3 from Lecture 4 of Fall 08 6.042 notes

Lemma 4.2.4

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):

$$\begin{aligned}
\gcd(1247, 899) &= \gcd(899, \underbrace{\text{rem}(1247, 899)}_{=248}) \\
&= \gcd(248, \underbrace{\text{rem}(899, 248)}_{=155}) \\
&= \gcd(155, \underbrace{248 \text{ rem } 155}_{=93}) \\
&= \gcd(93, \underbrace{\text{rem}(155, 93)}_{=62}) \\
&= \gcd(62, \underbrace{\text{rem}(93, 62)}_{=31}) \\
&= \gcd(31, \underbrace{\text{rem}(62, 31)}_{=0}) \\
&= \gcd(31, 0) \\
&= 31
\end{aligned}$$

process

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.

and it was discovered by the ~~Ancient~~ Greeks over 3000 years ago.

no #

The calculation, together with Corollary ~~7~~ 4.2.3, 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 ~~6~~!

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

4.3 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 6 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.

4.2.4

4.2.3 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 4.2.2 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

$$3 = s' \cdot 21 + t' \cdot 26 \tag{4.4}$$

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 then t' must be negative; otherwise, this expression would be much greater than 3.

we can

Now ~~here's how to~~ form 3 gallons using jugs with capacities 21 and 26. we simply repeat the follow steps s' times:

~~Repeat s' times:~~

1. Fill the 21-gallon jug.

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

At the end of this process, we must have 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 (4.4), 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 (4.4) 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 that approach gives:

$$
\begin{array}{lllllllll}
(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 repeat these two steps until the desired amount of water is obtained:

1. Fill the smaller jug.

   If at any time

2. Pour all the water in the smaller jug into the larger jug. ~~Whenever~~ 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 of the greatest common divisor of the jug capacities —all the quantities we can possibly produce. No ingenuity is needed at all!

4.2.5

4.2.4 The Pulverizer

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

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

INSERT 6.5 goes here

The previous subsection gives a roundabout and not very efficient method of finding such coefficients s and t. In Chapter 8.1.3 we defined and verified the "Extended Euclidean GCD algorithm," which is a much more efficient way to find these coefficients. In this section we finally explain where the obscure procedure in Chapter 8.1.3 came from by describing it in a way that dates to sixth-century India, where it was called *kuttak*, which means "The Pulverizer."

Insert 65

6.042/18.062J Mathematics for Computer Science
Tom Leighton, Marten van Dijk, and Brooke Cowan

September 17, 2008

Notes for Recitation 4

1 The Pulverizer

We saw in lecture that the greatest common divisor (GCD) of two numbers can be written as a linear combination of them.[1] That is, 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

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

unfortunately,
~~However,~~ the proof was *nonconstructive*: it didn't suggest a way for finding such s and t.

no # → That job is 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".~~ because it is so close to Euclid's Algorithm.

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

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

~~which was proved in lecture (see the notes "Number Theory I").~~ For example, we can compute the GCD of 259 and 70 as follows:

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

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

[1] In fact, we proved that among all positive linear combinations of the numbers their GCD is the smallest.

Suppose we use Euclid's Algorithm to compute the GCD of 259 and 70, for example:

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

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	$(\mathrm{rem}(x, y))$		$x - q \cdot y$
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 $\mathrm{rem}(x, y)$, 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.

4.2.5 Problems

Class Problems

4.3 The Fundamental Theorem of Arithmetic

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

Theorem 4.3.1 (Fundamental Theorem of Arithmetic). *Every positive integer n can be written in a unique way as a product of primes:*

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

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 you were in a 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 of preliminary facts.

Lemma 4.3.2. *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

David: make sure we have the digits ge right (re Albert's update)

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" mathematicians who had the misfortune of being contemporaries of Gauss.)

the only positive 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 Lemma 4.2.4. ■

A routine induction argument extends this statement to:

Lemma 4.3.3. *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 3.1.1 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

$$n = p_1 \cdot p_2 \cdots p_j$$

$$= q_1 \cdot q_2 \cdots q_k$$

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 4.3.3 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. ■

4.3.1 Problems

Class Problems

4.4 Alan Turing

Figure 42: Alan Turing.

The man pictured ~~above~~ in Figure 42 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.

4.4.1 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"
→	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 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:

$$\begin{aligned} m^* &= m \cdot k \\ &= 2209032015182513 \cdot 22801763489 \\ &= 503698255498207185946678 57 \end{aligned}$$

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, 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. Amazingly, the description of their breakthrough algorithm was only thirteen lines long!

Of course, a twelfth degree polynomial grows pretty fast, so the Agrawal, *et al.* procedure is of no practical use. Still, good ideas have a way of breeding more good ideas, so there's certainly hope that further improvements will lead to a procedure that is useful in practice. But the truth is, there's no practical need to improve it, since very efficient *probabilistic* procedures for prime-testing have been known since the early 1970's. These procedures have some probability of giving a wrong answer, but their probability of being wrong is so tiny that relying on their answers is the best bet you'll ever make.

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.

4.4.2 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:

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

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!

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.

4.5 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 iff $n \mid (a - b)$. This is written

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

For example:

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

There is a close connection between congruences and remainders:

Lemma 4.5.1 (Congruences and Remainders).

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

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

$$a = q_1 n + r_1 \qquad \text{where } 0 \leq r_1 < n,$$

$$b = q_2 n + r_2 \qquad \text{where } 0 \leq r_2 < n.$$

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 $r_1 = r_2$, that is, when $\text{rem}(a, n) = \text{rem}(b, n)$. ■

So we can also see that

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

This formulation explains why the congruence relation has properties like an equality relation. Notice that even though (mod 7) appears over on the right side the $\equiv$ symbol, it 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 stick to it.

We'll make frequent use of the following immediate Corollary of Lemma 4.5.1:

Corollary 4.5.2.

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

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:

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

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 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 of exceptions.

Lemma 4.5.3 (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. Parts 1.–3. follow immediately from Lemma 4.5.1. Part 4. follows immediately from the definition that $a \equiv b \pmod{n}$ iff $n \mid (a - b)$. Likewise, part 5. follows because if $n \mid (a - b)$ then it divides $(a - b)c = ac - bc$. To prove part 6., assume

$$a \equiv b \pmod{n} \tag{4.5}$$

and

$$c \equiv d \pmod{n}. \tag{4.6}$$

Then

$$a + c \equiv b + c \pmod{n} \qquad \text{(by part 4. and (4.5))},$$

$$c + b \equiv d + b \pmod{n} \qquad \text{(by part 4. and (4.6))}, \text{ so}$$

$$b + c \equiv b + d \pmod{n} \qquad \text{and therefore}$$

$$a + c \equiv b + d \pmod{n} \qquad \text{(by part 3.)}$$

Part 7. has a similar proof. ■

4.5.1 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-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.

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 (see `http://en.wikipedia.org/wiki/Polish_Cipher_Bureau`) 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 numbertheory and 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* arithmetic. 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 arithmetic.) 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:

$$m^* = \text{rem}(mk, p) \tag{4.7}$$

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.

4.5.2 Problems

Class Problems

4.6 Arithmetic with a Prime Modulus

4.6.1 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 4.6.1. *If p is prime and k is not a multiple of p, then k has a multiplicative inverse modulo p.*

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:

$$\begin{aligned} m^* \cdot k^{-1} &= \text{rem}(mk, p) \cdot k^{-1} && \text{(the def. (4.7) of } m^*) \\ &\equiv (mk)k^{-1} \pmod{p} && \text{(by Cor. 4.5.2)} \\ &\equiv m \pmod{p}. \end{aligned}$$

This shows that m^*k^{-1} is congruent to the original message m. Since m was in

the range $0, 1, \ldots, p-1$, we can recover it exactly by taking a remainder:

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

~~So now we can decrypt!~~ ~~provided that we can find k.~~ ~~Of course, there is still the~~

So all we need to decrypt the message is to find a value of k^{-1}. From the proof of Lemma 4.6.1, we know that t is such a value, where $s*t*k = 1$. Finding t is easy using the Pulverizer.

4.6.2 Cancellation

Another sense in which real numbers are nice is that one can cancel multiplicative terms. In other words, if we know that $m_1 k = m_2 k$, then we 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,

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

but cancelling the 3's leads to the *false* conclusion that $2 \equiv 4 \pmod{6}$. 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 4.6.2. *Suppose p is a prime and k is not a multiple of p. Then*

$$ak \equiv bk \pmod{p} \quad \text{IMPLIES} \quad a \equiv b \pmod{p}.$$

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 set of possible messages*. This is stated more precisely in the following corollary.

Corollary 4.6.3. *Suppose p is a prime and k is not a multiple of p. Then the sequence:*

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

is a permutation[2] *of the sequence:*

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

Proof. The sequence of remainders contains $p-1$ numbers. Since $i \cdot k$ is not divisible by p for $i = 1, \ldots p-1$, all these remainders are in the range 1 to $p-1$ by the definition of remainder. Furthermore, the remainders are all different: no two numbers in the range 1 to $p-1$ are congruent modulo p, and by Lemma 4.6.2, $i \cdot k \equiv j \cdot k \pmod{p}$ if and only if $i \equiv j \pmod{p}$. Thus, the sequence of remainders must contain *all* of the numbers from 1 to $p-1$ in some order. ■

reordering of the elements.

[2] A *permutation* of a sequence of elements is a sequence with the same elements (including repeats) possibly in a different order. More formally, if

$$\vec{e} ::= e_1, e_2, \ldots, e_n$$

is a length n sequence, and $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ is a bijection, then

$$e_{\pi(1)}, e_{\pi(2)}, \ldots, e_{\pi(n)},$$

is a *permutation* of $\vec{e}$.

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

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

is 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 set of possible messages are permuted by the process of encryption and thus can't read encoded messages.

4.6.3 Fermat's Little Theorem

A remaining challenge in using Turing's code is that decryption requires the inverse of the secret key k. An effective way to calculate k^{-1} follows from the proof of Lemma 4.6.1, namely

$$k^{-1} = \text{rem}(t, p)$$

where s, t are coefficients such that $sp + tk = 1$. Notice that t is easy to find using the Pulverizer.

to finding the inverse of the secret key k in Turing's code (

An alternative approach, about equally efficient and probably more memorable, is to rely on Fermat's Little Theorem, which is much easier than his famous

Last Theorem.

Theorem 4.6.4 (Fermat's Little Theorem). *Suppose p is a prime and k is not a multiple of p. Then:*

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

Proof. We reason as follows:

$$\begin{aligned}
(p-1)! ::= 1 \cdot 2 \cdots (p-1) & \\
&= \operatorname{rem}(k,p) \cdot \operatorname{rem}(2k,p) \cdots \operatorname{rem}((p-1)k,p) && \text{(by Cor 4.6.3)} \\
&\equiv k \cdot 2k \cdots (p-1)k \pmod{p} && \text{(by Cor 4.5.2)} \\
&\equiv (p-1)! \cdot k^{p-1} \pmod{p} && \text{(rearranging terms)}
\end{aligned}$$

Now $(p-1)!$ is not a multiple of p because the prime factorizations of $1, 2, \ldots, (p-1)$ contain only primes smaller than p. So by Lemma 4.6.2, we can cancel $(p-1)!$ from the first and last expressions, 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 $\text{rem}(6^{15}, 17)$, which we can do by successive squaring. All the congruences below hold modulo 17.

$$6^2 \equiv 36 \equiv 2$$

$$6^4 \equiv (6^2)^2 \equiv 2^2 \equiv 4$$

$$6^8 \equiv (6^4)^2 \equiv 4^2 \equiv 16$$

$$6^{15} \equiv 6^8 \cdot 6^4 \cdot 6^2 \cdot 6 \equiv 16 \cdot 4 \cdot 2 \cdot 6 \equiv 3$$

Therefore, $\text{rem}(6^{15}, 17) = 3$. 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 in-

verse 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.

4.6.4 Breaking Turing's Code —Again

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. Sup-

pose that the Nazis know both m and m^* where:

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

Now they can compute:

$$\begin{aligned} m^{p-2} \cdot m^* &= m^{p-2} \cdot \text{rem}(mk, p) && \text{(def. (4.7) of } m^*) \\ &\equiv m^{p-2} \cdot mk \pmod{p} && \text{(by Cor 4.5.2)} \\ &\equiv m^{p-1} \cdot k \pmod{p} \\ &\equiv k \pmod{p} && \text{(Fermat's Theorem)} \end{aligned}$$

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 deciphering of Enigma messages surely saved thousands of lives, if not the whole of Britain.

4.6.5 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 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.

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.

David: make sure you have the latest version of this box from Albert.

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:

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

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

$$1 + \frac{1}{5^s} + \frac{1}{5^{2s}} + \frac{1}{5^{3s}} + \cdots = \frac{1}{1-1/5^s}$$

etc.

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

4.6.6 Problems

Class Problems

Homework Problems

4.7 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, Ronald Rivest, Adi Shamir, and Leonard Adleman at MIT 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. The sender then encrypts his message using her widely-

distributed public key. Then she decrypts the received 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 —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.

4.7.1 Relative Primality and ϕ

First, we need a new definition. Integers a and b are *relatively prime* iff $\gcd(a, b) = 1$. For example, 8 and 15 are relatively prime, since $\gcd(8, 15) = 1$. Note that, except for multiples of p, every integer is relatively prime to a prime number 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 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. The function ϕ is known as *Euler's ϕ function*; it's also called Euler's *totient* function.

Theorem 4.7.1. *The function ϕ obeys the following relationships:*

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

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

Here's an example of using Theorem 4.7.1 to compute $\phi(300)$:

$$\begin{aligned}
\phi(300) &= \phi(2^2 \cdot 3 \cdot 5^2) \\
&= \phi(2^2) \cdot \phi(3) \cdot \phi(5^2) && \text{(by Theorem 4.7.1.(a))} \\
&= (2^2 - 2^1)(3^1 - 3^0)(5^2 - 5^1) && \text{(by Theorem 4.7.1.(b))} \\
&= 80.
\end{aligned}$$

The proof of Theorem 4.7.1.(a) requires a few more properties of modular arithmetic worked out in the next section (see Problem **??**). We'll also give another a proof in a few weeks based on rules for counting things.

To prove Theorem 4.7.1.(b), notice that every pth number among the p^k numbers in the interval from 0 to $p^k - 1$ is divisible by p, and only these are divisible by p. So $1/p$th of these numbers are divisible by p and the remaining ones are not. That is,

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

4.7.2 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, the proof of Lemma 4.6.1 of an inverse for k modulo p extends to an inverse for k relatively prime to n:

Lemma 4.7.2. *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 4.7.3. *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.

The following lemma is the natural generalization of Corollary 4.6.3.

4.7.3 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 4.7.4. *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 1 to $n-1$. Then the sequence:*

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

is a permutation of the sequence:

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

Proof. We will show that the remainders in the first sequence are all distinct and are equal to some member of the sequence of k_j's. Since the two sequences have the same length, the first must be a permutation of the second.

First, we show that the remainders in the first sequence are all distinct. Suppose that $\text{rem}(k_i k, n) = \text{rem}(k_j k, n)$. This is equivalent to $k_i k \equiv k_j k \pmod{n}$, which implies $k_i \equiv k_j \pmod{n}$ by Corollary 4.7.3. This, in turn, means that $k_i = k_j$ since both are between 1 and $n-1$. Thus, none of the remainder terms in the first sequence is equal to any other remainder term.

Next, we show that each remainder in the first sequence equals one of the k_i. By assumption, $\gcd(k_i, n) = 1$ and $\gcd(k, n) = 1$, which means that

$$\gcd(n, \text{rem}(k_i k, n)) = \gcd(k_i k, n) \qquad \text{(by Lemma 4.2.4.5)}$$

$$= 1 \qquad \text{(by Lemma 4.2.4.3)}.$$

Since rem$(k_i k, n)$ is in the range from 0 to $n-1$ by the definition of remainder, and since it is relatively prime to n, it must (by definition of the k_j's) be equal to some k_j. ■

INSERTS H1, H2 & H3 go here in that order

We can now prove Euler's Theorem:

Theorem 4.7.5 (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:

$$\begin{aligned}
& k_1 \cdot k_2 \cdots k_r \\
&= \operatorname{rem}(k_1 \cdot k, n) \cdot \operatorname{rem}(k_2 \cdot k, n) \cdots \operatorname{rem}(k_r \cdot k, n) && \text{(by Lemma 4.7.4)} \\
&\equiv (k_1 \cdot k) \cdot (k_2 \cdot k) \cdot \cdots \cdot (k_r \cdot k) \pmod{n} && \text{(by Cor 4.5.2)} \\
&\equiv (k_1 \cdot k_2 \cdots k_r) \cdot k^r \pmod{n} && \text{(rearranging terms)}
\end{aligned}$$

Lemma 4.2.4.7. implies that $k_1 \cdot k_2 \cdots k_r$ is prime relative to n. So by Corollary 4.7.3, we can cancel this product from the first and last expressions. This proves the claim. ∎

INSERT #1

4.7.2 Euler's Theorem

RSA relies heavily on a generalization of Fermat's Theorem known as Euler's Theorem. For both theorems, ~~The proof is similar to the key difference between the two theorems is that in Euler's theorem,~~ the exponent of k needed to produce an inverse of k modulo n depends on the number of integers in the set $\{1, 2, \ldots, n-1\}$ that are relatively prime to n. This value is known as Euler's ϕ function (a.k.a. Euler's totient function) and it is denoted as $\phi(n)$. 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 the only numbers

less than 12 that are relatively prime to 12.

~~For any~~

If n is prime, then $\phi(n) = n-1$ since every number ~~less than n between~~ & less than a prime is relatively prime to that prime. when n is composite, however, the ϕ function gets a little complicated. The following theorem characterizes the ϕ function for composite n. We won't prove the theorem in its full generality here, although we will give a proof for the special case when n is the product of two primes since that ~~case is of special~~ is the case that matters for RSA.

INSERT 142

is the textbook

p 207

INSERT #3

Corollary 4.6: Let $p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_j^{\alpha_j}$ be the unique prime factorization of an integer n where $p_1 < p_2 < \cdots < p_j$ for $j \geq 1$. Then

$$\phi(n) = \left(p_1^{\alpha_1} - p_1^{\alpha_1 - 1}\right)\left(p_2^{\alpha_2} - p_2^{\alpha_2 - 1}\right) \cdots \left(p_j^{\alpha_j} - p_j^{\alpha_j - 1}\right).$$

For example,

$$\phi(300) = \phi(2^2 \cdot 3 \cdot 5^2)$$
$$= (2^2 - 2^1)(3^1 - 3^0)(5^2 - 5)$$
$$= 2 \cdot 2 \cdot 20$$
$$= 80.$$

Corollary 4.7: Let $n = pq$ where p and q are different primes. Then $\phi(n) = (p-1)(q-1)$.

Proof of Corollary 4.7 follows easily from Corollary 4.6 and Theorem 4.7.1

Corollary 4.7 is important for RSA and since but since we have not provided a proof of Theorem 4.7.1, we will give a direct proof of Corollary 4.7. in what follows.

Proof of Corollary 4.7: Since p and q are prime, any number that is not relatively prime to $N = pq$ must be a multiple of p or a multiple of q. Among the numbers $1, 2, \ldots, pq-1$, there are precisely $q-1$ multiples of p and $p-1$ multiples of q, since p and q are relatively prime and since the numbers under consideration are less than pq, the $q-1$ multiples of p are different than the $p-1$ multiples of q.

Hence, $\phi(n) = (pq-1) - (q-1) - (p-1)$

$= pq - q - p + 1$

$= (p-1)(q-1)$,

as claimed. ∎

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 $\phi(n)$. Unfortunately, finding $\phi(n)$ is about as hard as factoring n, and factoring is hard in general. However, when we know how to factor n, we can use Theorem 4.7.1 to compute $\phi(n)$ efficiently. Then computing $k^{\phi(n)-1}$ to find inverses is a competitive alternative to the Pulverizer.

Coop: This can be done using Corollary 4.6 if we know the factorization of n.

INSERT 4.4 goes here

make into a section instead of subsection

~~4.7.3~~ ~~4.7.4 RSA~~

4.8 The RSA Algorithm

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

RSA Public Key Encryption

INSERT H4

$\Leftarrow$ Computing $\phi(n)$ is easy (using corollary H6) if we know the prime factorization of n. Unfortunately, finding the factors of n can be hard to do when n is large and so the Pulverizer is often the best approach to computing inverses mod n.

Remove from box & put into text

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

1. Generate two distinct primes, p and q. Since they can be used to generate the secret key, they must be kept hidden.
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 $de \equiv 1 \pmod{(p-1)(q-1)}$. This can be done using the Pulverizer.

   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: Given a message m, the sender first checks that gcd(m,n) = 1.[1] then

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

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

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

We'll explain why this way of Decoding works in Problem ??.

[1] It would be very bad if gcd(m,n) equals p or q since then it would be easy for someone to use the encoded message to compute the secret key. If gcd(m,n) = n, then the encoded message would be 0, which is clearly useless.

— INSERT H6 and H7 go here —

4.7.5 Problems

Practice Problems

Class Problems

Homework Problems

Exam Problems

4.9 Problems

This is INSERT HG (from lecture 5 Fall 08 6.042 Notes)

5.4 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 in $\{1, \ldots, n-1\}$ to produce m' using the public key:

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

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

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

Why does decoding work? We need to show that the decryption $\text{rem}\,((m')^d, n)$ is indeed equal to the sender's message m. Since the sender encrypts the message m, $m' = rem(m^e, n)$, m' is congruent to m^e modulo n by Lemma 2.1. That is,

$$m' \equiv m^e \pmod{n}.$$

By raising both sides to the power d, we obtain the congruence

$$(m')^d \equiv m^{ed} \pmod{n}. \tag{1}$$

The encryption exponent e and the decryption exponent d are chosen such that $de \equiv 1 \pmod{(p-1)(q-1)}$. So, there exists an integer r such that $ed = 1 + r(p-1)(q-1)$. By substituting $1 + r(p-1)(q-1)$ for ed in (1), we obtain

$$(m')^d \equiv m \cdot m^{r(p-1)(q-1)} \pmod{n}. \tag{2}$$

If we can show that

$$m \cdot m^{r(p-1)(q-1)} \equiv m \pmod{n}, \tag{3}$$

INSERT #7

~~Corollary #2~~
~~From Corollary #2, we know~~
By Euler's Theorem ~~we know that~~

and the assumption that $\gcd(m,n) = 1$,

we know that ~~$m^{\phi(n)} \equiv 1 \pmod{n}$~~.

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

From Corollary #2, we know that

$\phi(n) = (p-1)(q-1)$. Hence,

$$(m')^d \equiv m \cdot m^{r(p-1)(q-1)} \pmod{n}$$

$$\equiv m \cdot (1)^r \pmod{n}$$

$$\equiv m \pmod{n}.$$

~~as desired.~~ Hence, the decryption process indeed reproduces the original message m.

Is it hard for someone without the secret key to decrypt the message? No one knows for sure but it is generally

believed that if n is a very large number (say with a thousand digits), then it is difficult to reverse engineer d from e and n. Of course, it is easy to compute d if you know p and q (by using the Pulverizer) but it is not known quickly how to factor n into p and q when n is very large. Maybe with a little more studying of number theory, you will be the first to figure out how to do it. Although, we should warn you that Gauss worked on it for years without a lot to show for his efforts. And if you do figure it out, you might wind up meeting some serious-looking fellows in black suits...