Discrete Mathematics Lecture Notes

Brent Yorgey

April 29, 2025

These are my lecture notes for MATH 240, Discrete Mathematics, at Hendrix College.

1 Introduction (Wednesday 22 January)

1.1 Setup

- Go around, greet everyone and learn their names.

1.2 First day activities

Introduce myself (with slide show). Explain what "Discrete Math" is all about: math is a web of connections, not a linear chain of topics. Discrete vs continuous. Math needed to be a good programmer. Big themes: (1) formal notation is a superpower! (2) Thinking and expressing ourselves using rigorous logic!

1.3 Syllabus

Show class website. Go over syllabus. Homework, quizzes. Engagement points, learning goals.

Homework: fill out survey, install Disco.

1.4 Introduction to Disco

S'25: Show Disco. Get them all to bring it up, play around with it, talk to neighbors, share insights. Go over some of the basics.
This went well in S'23 but can't do it in S'25 since replit doesn't work any more. Have to get them to install it for homework. Maybe mention that I'm looking for someone to work on building a web UI over the summer.

With whatever time is left, do a basic introduction to Disco. Get them to tell me things to try!

2 Introduction to Disco (Friday 24 January)

2.1 Setup

- Go over syllabus (if not already done).
- Show them how to make an office hours appointment. Encourage them to email or message me on Teams.
- Note first HW assignment is now posted on website.

2.2 Disco Arithmetic and Functions

Everything in Disco has a *type* which tells us what kind of value it is (or will be). We can ask Disco what type something has with the `:type` command. When we define a variable or function, we write

```
name : type
```

to say what type it has. Every definition must have a type.

If we ask for the type of a number like `3`, Disco tells us it has type $\mathbb{N}$:

```
Disco> :type 3
3 : ℕ
```

$\mathbb{N}$ stands for *natural numbers*, and denotes one of the non-negative counting numbers $0, 1, 2, 3, \ldots$ and so on. (Note this fancy style of writing capital letters with a double line in the middle is called "blackboard bold", and is the standard way mathematicians write $\mathbb{N}$ and other similar types we will meet soon. However, you do not have to use fancy symbols when writing Disco code. For example, $\mathbb{N}$ can instead be written `N`.)

A collection of numbers is *closed* under a certain operation if doing that operation on two of the numbers always gives another such number. For example, the natural numbers are closed under addition (that is, adding any two natural numbers gives another natural number) and under multiplication (multipltying any two natural numbers gives another natural number).

```
Disco> :type 3 + 5
3 + 5 : ℕ
Disco> :type 3 * 5
3 * 5 : ℕ
```

However, the natural numbers are *not* closed under subtraction, because subtracting two natural numbers might not always result in a natural number. For example, $2 - 5$ is not a natural number. If we want to be able to do subtraction, we must include all negative numbers along with positive. This bigger set of numbers $\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots$ is called the *integers*.

```
Disco> :type 3 - 5
3 - 5 : ℤ
```

Note it does not matter whether the result would *actually* be negative or not. Disco does not actually simplify/evaluate things to figure out what type they are. In fact, the whole point is to be able to tell what kind of value we will get *before* actually trying to evaluate an expression.

```
Disco> :type 3 - 1
3 - 1 : ℤ
```

Similarly, ℕ is not closed under division, and Disco has other types to handle that, just like it has ℤ to handle subtraction, but we'll talk about them later.

We can define our own functions in Disco. First, we give the name of the function and its *type*, which will be something like `A -> B` where `A` is the type of the function's inputs, and `B` is the type of the outputs. Then we define what output the function should give for each input.

```
double : N -> N
double(n) = 2n
```

If we write the above definition of `double` in a `.disco` file, we can load it at the Disco prompt using the `:load` command, then try it on some inputs:

```
Disco> :load double.disco
Loading double.disco...
Loaded.
Disco> double(2)
4
```

We can also attach *tests* to our functions.

```
!!! double(0) == 0
!!! double(2) == 4
!!! double(7) == 14
!!! forall n : N. double(n) >= 0
double : N -> N
double(n) = 2n
```

Tests can either be simple true or false tests, or they can have a `forall` to say that something should be true for every value of a certain type. When we `:load` this file, Disco will run the tests and report whether they succeed.

```
Disco> :load double.disco
Loading double.disco...
Running tests...
  double: OK
Loaded.
```

If we try changing the `double(n) >= 0` test to `double(n) > 0`, we can see that the test fails:

```
Disco> :load double.disco
Loading double.disco...
Running tests...
  double:
  - Test is false: ∀n. double(n) > n
    Counterexample:
      n = 0
Loaded.
```

However, another way to express a correct test would be as follows:

```
!!! forall n : N. (n == 0) or double(n) > 0
```

As a final example, let's implement factorial (although it is already built in). Recall that $n! = n \cdot (n-1) \cdot (n-2) \cdot \cdots \cdot 1$. Another way to write this is

$$n! = n \cdot (n-1)!$$

If we add a "base case" $0! = 1$, this becomes a perfectly viable way to *define* factorial. We could transcribe it into Disco as follows:

```
fac : N -> N
fac(0) = 1
fac(n) = n * fac (n-1)
```

Unfortunately, this is a type error. The reason is that since the input to `fac` is supposed to be a natural number, that means $n - 1$ must be a natural number, but it cannot be since it uses subtraction. We can use the `.-` operator instead, which works on natural numbers (you will explore what this operator does for homework).

```
fac : N -> N
fac(0) = 1
fac(n) = n * fac (n .- 1)
```

3 Introduction to Propositional Logic (Monday 27 January)

- Write two problems on the board:
  - "30 people each buy a ticket to see the heffalump. The owner pays $12 for food and ends up with $33 in profit. How much does one ticket cost?"
  - "In what circumstances does the following loop *stop*?"

```
while (tickets < 30 or (!heffalumps.isEmpty() and not(count > 5))):
  ...
```

Have them work on solving the problems with their neighbors. How did you solve the first one? (algebra.) How did you solve the second? (talking, natural language.) Which was easier? (the first one). Formal notation is a superpower! People used to solve algebra problems using natural language too. Our goal: allow you to solve the second problem using a formal language too.

Every programming language has *Boolean* true/false values. (Named for George Boole, English mathematician, 1815–1864, "The Laws of Thought" (1854).) Where have we seen them in Disco? We're going to learn how to manipulate them according to formal algebraic rules.

3.1 Propositions

Definition 3.1. A *proposition* is a declarative statement that can be true or false.

Example.
- "Washington, D.C. is the capital of the United States." (true)
- "Toronto is the capital of Canada." (false proposition)
- "10 is divisible by 3." (false)
- "10 is even." (true)

Example. Some non-examples of propositions:

- 6
- "Please hold this."

We will use p, q, r, s, P, Q, R, S to stand for arbitrary propositions. Our goal is to develop a language for writing down formal, precise propositions and manipulating them. Formal, algebraic notation is a superpower!

3.2 Propositional logic connectives

Our goal is to see ways to build more complex propositions from simpler ones.

Negation

Definition 3.2. Let p be a proposition. The *negation* of p is written $\neg p$ (or $\overline{p}$) and read "not p" or "It is not the case that p."

We can make a *truth table* showing the truth value of $\neg p$ for each possible truth value of p:

p	$\neg p$
T	F
F	T

We can also get Disco to show us the truth table for $\neg$ using the `:table` command:

```
Disco> :table not
F  T
T  F
```

Conjunction

Definition 3.3. The *conjunction* of propositions p, q is written $p \wedge q$ ("p and q"). It is true when both p and q are true, and false otherwise.

Let's make a truth table. Notice that the truth table for $\neg p$ needed two rows, one for T and one for F. The truth table for $p \wedge q$ will need *four* rows, one for each possible combination of truth values for p and q.

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

Example. Conjunction works just like we expect in natural language. Consider: "It is not raining today and I had eggs for breakfast."

This is true if both parts are true. If it's raining, the statement is false. If I ate something other than eggs, it's false. Of course it's definitely false when it's raining and I ate something other than eggs.

If we let $r =$ "It is raining today" and $e =$ "I had eggs for breakfast", then we can translate this sentence into propositional logic as

$$\neg r \wedge e.$$

Note that negation has "higher precedence" than conjunction (it has "stickier glue") so this is unambiguous. It does *not* mean $\neg(r \wedge e)$, which would be different. If in doubt, just use parentheses: $(\neg r) \wedge e$.

Disjunction

Definition 3.4. The *disjunction* of p, q is written $p \vee q$ ("p or q"). It is false when both are false, and true otherwise.

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

The symbols $\wedge$ and $\vee$ are easy to mix up. Just remember that $\wedge$ looks like a capital "A" for "And", $\vee$ looks like a capital "V" for "Vote" (a vote is when you choose between options).

There's a good reason the symbols are upside down versions of each other: they are "opposite" (the mathy word is "dual") in the sense that if we consider an opposite world where everything false becomes true and vice versa, then $\wedge$ becomes $\vee$ and vice versa. Just look at their definitions: note how the definition of $\vee$ is the same as that for $\wedge$ but with all the T's and F's switched.

The way we usually think about $p \vee q$, though, is that $p \vee q$ is true whenever *at least one* of p, q is true.

Which statement from the side board does this correspond to? (The one about prerequisites.) We use the word "or" with two different meanings in English. This one, where it's OK for both things to be true, is called *inclusive* or. The other is called *exclusive* or, and written $p \oplus q$. It is true when *exactly one* of p, q is true, and false otherwise. (Put another way, it is true when p and q are different, and false when they are the same.) Exclusive or is very important in computer science, but rarely comes up in mathematical logic. We'll return to it later in the course perhaps.

Example. Let r = "It is raining today", e = "I had eggs for breakfast", and c = "I had cereal for breakfast." Consider the sentence "Either it is raining, or I ate cereal and not eggs for breakfast." How do we translate this in to propositional logic?

$$r \vee (c \wedge \neg e)$$

(Notes: not clear which kind of or to use; it doesn't matter too much. The parenthesis may technically not be required, but it's best to include them.)

What about "If I ate eggs for breakfast, then it is raining"?

4 Implication & Propositional Equivalences (1.3) (Wednesday 29 January)

Let r = "It is raining today", e = "I had eggs for breakfast", and c = "I had cereal for breakfast." Consider the sentence "Either it is raining, or I ate cereal and not eggs for breakfast." How do we translate this in to propositional logic?

$$r \vee (c \wedge \neg e)$$

(Notes: not clear which kind of or to use; it doesn't matter too much. The parenthesis may technically not be required, but it's best to include them.)

Booleans in Disco In Disco, `T` and `F` are values of type `Bool`. We can write logical AND using `and`, `&&`, `/\`, or $\wedge$; similarly for OR. NOT can be written `not` or $\neg$.

We can get Disco to show us truth tables for operators, e.g. `:table not` or `:table /\`.

Implication

Definition 4.1. Let p, q be propositions. The *conditional* or *implication* "if p, then q" is written $p \to q$.

p is the *premise* or *hypothesis*. q is the *conclusion*. (Note, there are lots of words for everything! Don't memorize, but you will need to be familiar with them.) Let's figure out what the truth table should be. I like to think of it this way: if someone makes you a promise of the form "if ... then ...", in what scenarios have they lied to you?

Consider the propositions

p = "You fulfill all the requirements for an A."
q = "You get an A."

In the syllabus I have in fact promised that the proposition $p \to q$ is true. Let's consider every possible scenario; in which scenario should you be mad at me for lying to you?

- You complete all the requirements and get an A. This is great; I have kept my promise.
- You complete all the requirements but don't get an A. Obviously in this scenario I lied and you should be super mad at me, and probably complain to my department chair or the Provost.
- Let's next consider the scenario where you don't complete the requirements for an A, and don't get an A. You might be mad at yourself but you certainly can't be mad at me. I still made a true statement.

- The last case is the one that sometimes trips people up: what if you don't complete the requirements, but you get an A anyway? You obviously won't be upset in this scenario (though other students might!), but *did I lie*? Actually, I didn't. I only said what would happen if you DO complete the requirements. I am free to do whatever I want (including giving you an A anyway) if you don't. (In fact, I occasionally do this in special circumstances!)

So overall we have the following truth table:

p	q	$p \to q$
T	T	T
T	F	F
F	T	T
F	F	T

In other words, the only scenario in which an implication $p \to q$ is false is when p is true but q is false. It is true in all other cases. "If the premise is false, anything goes!"

Note that we have *lots* of ways to express conditionals in natural language besides just "if p then q". For example, "if p, (then) q", "p implies q", "q if p", "p only if q", and so on. If you're not sure, make a truth table!

Definition 4.2. Given an implication $p \to q$, we have:

- the *converse* is $q \to p$
- the *inverse* is $\neg p \to \neg q$
- the *contrapositive* is $\neg q \to \neg p$ (the contrapositive is the *inverse of the converse* or vice versa)

Theorem 4.3. *An implication and its contrapositive always have the same truth value. On the other hand, an implication does* not *always have the same truth value as its inverse or converse.*

Talk to your neighbor and convince yourself that an implication has the same truth value as its contrapositive. How would we prove this?

4.1 Tautologies and equivalence

Definition 4.4. Let p, q be propositions. The *biconditional* $p \leftrightarrow q$ ("p if and only if q") is true when p and q have the same truth value, and false otherwise.

Remark. We often abbreviate "if and only if" as "iff".

Definition 4.5. A proposition that is always true, no matter the values of the propositional variables it contains, is a *tautology*. One that is always false is a *contradiction*.

Example. Get the class to come up with some examples of tautologies, contradictions, and some that are neither.

How would we prove these? Use logical reasoning, or use a truth table. Or use Disco!

Definition 4.6. Two propositions that always have the same truth values are *logically equivalent*, written $p \equiv q$. Alternatively, $p \equiv q$ when $p \leftrightarrow q$ is a tautology.

Example. Show that $p \to q \equiv \neg p \vee q$.

Make a truth table:

p	q	$p \to q$	$\neg p$	$\neg p \vee q$
T	T	T	F	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

Notice that the columns for $p \to q$ and $\neg p \vee q$ are exactly the same, which means that these two propositions always have the same truth value. Hence they are logically equivalent.

Example. Show that $p \to q \equiv \neg q \to \neg p$.

5 Algebraic laws for propositional logic (Friday 31 January)

5.1 Core laws

I'm going to show you a core set of logical equivalences that you should know. All others can be derived using these. The goal is to get rid of truth tables: we used them to bootstrap ourselves into understanding propositional logic operations, but from now on we can work more algebraically.

There are two important organizing/mnemonic principles:

1. Most equivalences have an "opposite world" version where we switch $\wedge/\vee$ and T/F.

2. We can get some intuition by thinking in terms of multiplication and addition:

   - T is kind of like 1
   - F is kind of like 0
   - $\wedge$ is kind of like $\times$
   - $\vee$ is kind of like $+$, but with a maximum result of 1

   This analogy is not perfect but it works well as a mnemonic to help us remember some of the equivalences.

Name	equivalence	"opposite world" equivalence
Identity	$p \wedge \mathsf{T} \equiv p$	$p \vee \mathsf{F} \equiv p$
Annihilation	$p \wedge \mathsf{F} \equiv \mathsf{F}$	$p \vee \mathsf{T} \equiv \mathsf{T}$
Idempotence	$p \wedge p \equiv p$	$p \vee p \equiv p$
Commutativity	$p \wedge q \equiv q \wedge p$	$p \vee q \equiv q \vee p$
Associativity	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	$(p \vee q) \vee r \equiv p \vee (q \vee r)$
Distributivity	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
De Morgan	$\neg(p \wedge q) \equiv \neg p \vee \neg q$	$\neg(p \vee q) \equiv \neg p \wedge \neg q$
Contradiction/Tautology	$p \wedge \neg p \equiv \mathsf{F}$	$p \vee \neg p \equiv \mathsf{T}$

Remark. Idempotence, "opposite world distributivity", and the De Morgan laws are the surprising ones that don't have a nice counterpart in terms of multiplication and addition.

Then there are three special equivalences which don't have opposite world versions:

Double negation elimination	$\neg(\neg p) \equiv p$
Implication	$p \to q \equiv \neg p \vee q$
Iff	$p \leftrightarrow q \equiv (p \to q) \wedge (q \to p)$

Note that associativity means we are justified in writing things like

$$p \wedge q \wedge r \wedge s \wedge t \wedge \ldots$$

with no parentheses, since where we put parentheses doesn't matter. (But if we mix $\wedge$ and $\vee$ we definitely need parentheses.)

You should convince yourself of all these, using either a truth table or thinking about a logical argument.

From now on, we don't have to use truth tables anymore, we can prove any equivalence we want using these core equivalences. Here are a couple examples.

Example. Prove $(p \to q) \wedge (p \to r) \equiv p \to (q \wedge r)$.

$$\begin{array}{ll} & (p \to q) \wedge (p \to r) \\ \equiv & \quad \{ \quad \text{Implication} \quad \} \\ & (\neg p \vee q) \wedge (\neg p \vee r) \\ \equiv & \quad \{ \quad \text{Factor out } \neg p \text{ (distributivity, backwards)} \quad \} \\ & \neg p \vee (q \wedge r) \\ \equiv & \quad \{ \quad \text{Implication} \quad \} \\ & p \to (q \wedge r) \end{array}$$

S'20: Only made it to here. Can do more examples if time.

Example. Show $\neg(p \vee (\neg p \wedge q)) \equiv \neg p \wedge \neg q$.

$$\begin{array}{ll} & \neg(p \vee (\neg p \wedge q)) \\ \equiv & \quad \{ \quad \text{DeMorgan} \quad \} \\ & \neg p \wedge \neg(\neg p \wedge q) \\ \equiv & \quad \{ \quad \text{DeMorgan} \quad \} \\ & \neg p \wedge (\neg\neg p \vee \neg q) \\ \equiv & \quad \{ \quad \text{Double negation elimination} \quad \} \\ & \neg p \wedge (p \vee \neg q) \\ \equiv & \quad \{ \quad \text{Distributivity} \quad \} \\ & (\neg p \wedge p) \vee (\neg p \wedge \neg q) \\ \equiv & \quad \{ \quad \text{Contradiction} \quad \} \\ & \mathsf{F} \vee (\neg p \wedge \neg q) \\ \equiv & \quad \{ \quad \text{Identity} \quad \} \\ & \neg p \wedge \neg q \end{array}$$

If yet more time left, show $(p \wedge q) \to (p \vee q)$ is a tautology, by deriving equivalence to T.

6 Predicates and Quantifiers (1.4, 1.5) (Monday 3 February)

By this point we've caught up with the ancient Greeks. However, it turns out that propositional logic isn't expressive enough to talk about everything we want to reason about mathematically. In particular we can't make general sorts of statements involving words like "all" or "some". We are able to write tests with `forall` in Disco; let's think more formally about what this means.

6.1 Predicates

Consider the statement

$$x + 2 = 5.$$

We actually used this as an example on the first day of class: it is *not* a proposition because we can't say whether it is true or false; it depends on x.

```
Disco> :test x + 2 == 5
Error: there is nothing named x.
https://disco-lang.readthedocs.io/en/latest/reference/unbound.html
```

However, it's *almost* a proposition: it's "waiting for" a value of x to be filled in, at which point it will be a proposition. We can make this explicit by defining $P(x)$ to be the statement "$x + 2 = 5$"; then $P(x)$ is a *predicate*, that is, a function that takes x as input and outputs a proposition.

```
P : N -> Bool
P(x) = x + 2 == 5
```

Example.

- $P(2)$ is the proposition $2 + 2 = 5$ (which happens to be false).
- $P(3)$ is the proposition $3 + 2 = 5$ (which happens to be true).

We can also make multi-argument predicates, just like we can have functions of multiple variables. For example, let $T(x, y)$ mean "x is less than the square of y".

```
T : N * N -> Bool
T(x,y) = x < y^2
```

6.2 Quantifiers

Definition 6.1. If $P(x)$ is a predicate and D is some "domain" (*i.e.* set of values, *i.e.* type) then

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

(pronounced "for all x in D, $P(x)$") is a proposition which is true iff $P(x)$ is true for every x in the domain D.

Remark. You also often see notation $\forall x \in D.P(x)$.

Remark. D is called the *domain of discourse*. Sometimes mathematicians omit writing it as part of the notation, and simply require that it be mentioned off to the side somehwere what the domain of discourse is, but I think this is silly: specifying the domain of discourse is not optional (it can determine whether the proposition is true or false), so we should force ourselves to put it in the notation, otherwise the notation is ambiguous.

Example.
```
Disco> :test forall n:N. P(n)
  - Certainly false: ∀n. P(n)
    Counterexample:
      n = 0
Disco> :test forall n:N. n >= 0
  - Possibly true: ∀n. n >= 0
    Checked 100 possibilities without finding a counterexample.
Disco> :test forall b:Bool. (b /\ b) == b
  - Certainly true: ∀b. (b /\ b) == b
    No counterexamples exist.
Disco> :test forall x:Z. x^2 >= 0
  - Possibly true: ∀x. x ^ 2 >= 0
    Checked 100 possibilities without finding a counterexample.
```

Remark. We should think of $\forall$ as being like a big "and". That is,

$$\forall x : D.\ P(x) \equiv P(x_1) \wedge P(x_2) \wedge P(x_3) \wedge \dots$$

where x_1, x_2, x_3, ... are all the elements of the domain D. But writing out such giant lists of propositions would be tedious; the domain could even be infinite, as in the last example! But even though this is not literally true it will help us think about what properties $\forall$ should have.

Definition 6.2. If $P(x)$ is a predicate and D any domain, then

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

("there exists an x in D such that $P(x)$") is a proposition which is true iff there is at least one x in D such that $P(x)$ holds.

Example.
```
Disco> :test exists x:N. P(x)
  - Certainly true: ∃x. P(x)
    Found example:
      x = 3
Disco> :test forall x:N. exists y:N. T(x,y)
  - Possibly true: ∀x. ∃y. T(x, y)
    Checked 100 possibilities without finding a counterexample.
Disco> :test forall y:N. exists x:N. T(x,y)
  - Possibly false: ∀y. ∃x. T(x, y)
    No example was found; checked 100 possibilities.
    Counterexample:
      y = 0
```

Remark. Dually to $\forall$, we can think of $\exists$ as a giant "or", that is,

$$\exists x : D.\ P(x) \equiv P(x_1) \vee P(x_2) \vee P(x_3) \vee \dots$$

even though it is not actually defined this way.

7 More quantifiers: De Morgan, examples (Wednesday 5 February)

7.1 Properties of $\forall$, $\exists$

Everything follows from thinking about $\forall$ in terms of $\wedge$ and $\exists$ in terms of $\vee$!

First we have the De Morgan laws for quantifiers (see if they can come up with these laws themselves!):

$$\neg(\forall x : D.\ P(x)) \equiv \exists x : D.\ \neg P(x)$$
$$\neg(\exists x : D.\ P(x)) \equiv \forall x : D.\ \neg P(x)$$

(There are other equivalences such as

$$\forall x : D.\ (P(x) \wedge Q(x)) \equiv (\forall x : D.\ P(x)) \wedge (\forall x : D.\ Q(x))$$
$$\exists x : D.\ (P(x) \vee Q(x)) \equiv (\exists x : D.\ P(x)) \vee (\exists x : D.\ Q(x))$$

but these are rarely all that useful.)

Example. What is the negation of $\forall a{:}\mathbb{N}.\ \exists b{:}\mathbb{N}.\ (a = 0) \vee (a + b = 0)$?

Translate each English sentence into formal logic.

Example. "Every natural number is less than or equal to its own square."

$$\forall n : \mathbb{N}. n \leq n^2$$

Example. "The square of any integer is nonnegative."

$$\forall x : \mathbb{Z}. x^2 \geq 0$$

Example. "1369 is a perfect square."

$$\exists n : \mathbb{N}. n^2 = 1369$$

Note that this is much nicer than trying to write something like

$$\text{IsInteger}(\sqrt{n}).$$

$\sqrt{-}$ is complicated, and we don't know how to define IsInteger.

Example. "n is even."

$$\text{Even}(n) = \exists k{:}\mathbb{Z}.\ 2k = n.$$

Example. "n is odd."

$$\text{Odd}(n) = \exists k{:}\mathbb{Z}.\ 2k + 1 = n.$$

Or we could define it as $\neg\,\text{Even}(n)$. (Actually slightly nontrivial to prove formally that these are the same thing—we won't be able to prove it until we go over the Division Algorithm in April!)

Example. "If n is even, then $n + 2$ is even."

$$\forall n{:}\mathbb{Z}.\ \text{Even}(n) \to \text{Even}(n + 2)$$

Note that bare variables in mathematical statements like this tend to hide an implicit $\forall$!

8 Quantifier restriction (Friday 7 February)

How would we encode the (false) proposition "every even integer is equal to 2"? We could write

$$\forall x : \text{Even}.\ x = 2$$

Or, if we wanted to use a more primitive quantifier, we could equivalently write

$$\forall x : \mathbb{Z}.\ \text{Even}(x) \to (x = 2)$$

Draw some pictures. In general, suppose D is some domain and S is a subset, and let $S(x)$ be a predicate meaning "x is a member of S". Then

$$\forall x : S.\ P(x) \equiv \forall x : D.\ S(x) \to P(x).$$

Likewise, if D is some domain and S is a subset, then

$$\exists x : S.\ P(x) \equiv \exists x : D.\ S(x) \wedge P(x).$$

As a final example, let's encode the statement "Every even integer greater than 2 can be written as the sum of two primes" in formal propositional logic. Assume we have a predicate $\text{Prime}(n)$ meaning "n is prime" (we will see how to define this later).

$$\forall n : \mathbb{N}.\ (\text{Even}(n) \wedge n > 2) \to (\exists p : \mathbb{N}.\ (\text{Prime}(p) \wedge (\exists q : \mathbb{N}.\ \text{Prime}(q) \wedge n = p + q)))$$

9 Introduction to proofs (Monday 10 February)

What is a *proof*? In a court of law, a proof is convincing evidence. In math we desire something more logically ironclad.

Definition 9.1. A *proof* is a logically valid argument that establishes the truth of a statement. What this looks like can vary a lot depending on the context, audience, and subject matter.

We distinguish two types of proofs (though it's really more of a continuum). A **formal** proof:

- Uses only axioms (assumptions) and things previously proved.
- Consists of a series of steps where each step is a valid *logical inference* from previous steps or assumptions.
- Is often expressed in formal notation.

We will *not* write these! The only places you will ever see a real, formal proof is (1) in a geometry class, or (2) a computer-checked formal proof (these are really interesting, take my Functional Programming class if you want to learn more!).

In contrast, an **informal** proof:

- Uses only axioms and things previously proved (same as formal proofs!).
- May omit or combine steps.
- Is often expressed in natural language.
- *In principle* could be expanded into a complete formal proof.
- Has other humans as its audience.

We will start more towards the formal end so that you understand it.

9.1 Logical inference

What is a valid *logical inference*? We will leave this mostly to our intuition. (Your book covers it in section 1.6 but it's overwhelming and unhelpful, IMO.) You have been using valid logical inferences your whole life. We'll give just two examples:

1. *If $A \to B$ is true and A is true, then B is true.* This is called *modus ponens* but you don't need to know that. One important point to make is that we don't have to just rely on our intuition to establish the validity of this reasoning principle. We can encode it as a proposition and check that it is a tautology (actually doing so makes for a nice exercise):

$$(A \to B) \wedge A \to B \equiv \mathsf{T}.$$

2. *If $A \wedge B$ is true, then A is true.* If you have established something of the form $A \wedge B$, you can always just ignore one of the two parts if you wish. Again, we can show this is valid by showing it is a tautology:

$$(A \wedge B) \to A \equiv \mathsf{T}.$$

There are many more; see your book for a list, if you like. But you'll probably be just fine.

Definition 9.2. A *theorem* is a statement that can be established to be true via a valid proof. (There are other terms you can read about in your book, *i.e.* "conjecture", "lemma", and so on, but we will discuss them as needed.)

9.2 How to Prove Things

See document, `proofs.pdf`. Just follow these three easy steps!

1. Translate the statement into propositional logic (using $\wedge$, $\vee$, $\neg$, $\to$, $\forall$, and $\exists$).
2. Write a proof outline corresponding to the propositional logic formula.
3. Use your intuition/insight/ingenuity/specific content knowledge to fill in the missing bits.

This might feel a bit facetious (cf How To Draw An Owl meme), but also serious: step 2 is mechanical, and gets us a lot of the way there; step 3 is the step that requires some creativity/insight.

Let's look at each propositional logic operator in turn, and look at how to write an appropriate proof outline for each. (See "How to Prove Things" document, `proofs.pdf`, for templates!)

9.2.1 $p \wedge q$

To prove a conjunction $p \wedge q$, prove p and prove q.

Give an example. Prove "$3 < 5 \wedge 8 \cdot 2 = 16$".

9.2.2 $p \vee q$

To prove a disjunction $p \vee q$, you can do *any* of the following:

1. Prove p.
2. Prove q.
3. Use a proof by contradiction (we will talk about this next class).

10 More proofs and proof examples (Wednesday 12 February)

10.0.1 $p \to q$

Remember, if p is false, we don't care; in that case $p \to q$ is going to be true no matter what. So we care about the case when p is true. In that case q had better be true too! So to prove $p \to q$ we can (temporarily) *suppose* that p is true, then show that q must be true in that imaginary world. (Note: "assume" means "this is something we think is true"; "suppose" is just hypothetical, i.e. expressing a "what if".)

To prove an implication, $p \to q$, you can:

1. Temporarily *suppose* p is true, then prove q. (You will sometimes hear this called a *direct proof*, but the name does not matter.)
2. Prove the *contrapositive*, that is, prove $\neg q \to \neg p$. Sometimes this is easier.

10.0.2 $p \leftrightarrow q$

To prove an if and only if, $p \leftrightarrow q$, prove $(p \to q) \wedge (q \to p)$.

10.0.3 $\neg p$

To prove a negation $\neg p$:

1. Use De Morgan laws to "push the negation inwards", then use one of the other proof rules. For example, if you wanted to prove something of the form $\neg(p \wedge q)$, first use a De Morgan law to transform this into $\neg p \vee \neg q$; then use one of the methods listed above for proving a disjunction.
2. Prove $p \to \mathsf{F}$. (Note $p \to \mathsf{F} \equiv \neg p \vee \mathsf{F} \equiv \neg p$.)

Example. Prove $P \to (Q \vee R)$.

Proof. We must prove $P \to (Q \vee R)$. So suppose P is true; we will show $(Q \vee R)$.

> To show $(Q \vee R)$, we will in fact show that Q holds.
>
> > *Proof of Q, using P*
>
> Since we showed Q, therefore $Q \vee R$.

Since we showed $Q \vee R$ under the supposition P, therefore $P \to (Q \vee R)$. □

Example. Prove $(A \vee B) \to \neg C$ using a proof by contrapositive.

Proof. We must show $(A \vee B) \to \neg C$, which we will do by proving the contrapositive, namely, $C \to \neg(A \vee B)$.

> To prove $C \to \neg(A \vee B)$, suppose C is true; we will show $\neg(A \vee B)$.
>
> > $\neg(A \vee B)$ is equivalent to $\neg A \wedge \neg B$, which we will prove by showing each individually.
> >
> > > *Proof of $\neg A$, using C*
> >
> > > *Proof of $\neg B$, using C*
> >
> > Thus, we have shown $\neg A \wedge \neg B$, or, equivalently, $\neg(A \vee B)$.
>
> Since we showed $\neg(A \vee B)$ while supposing C, therefore $C \to \neg(A \vee B)$.

Therefore, the contrapositive holds: $(A \vee B) \to \neg C$. □

Example. Prove that $(2n + 1 > 6) \to (n > 2)$.

Proof. To prove $(2n + 1 > 6) \to (n > 2)$, suppose $(2n + 1 > 6)$, then we must show $(n > 2)$.

- Starting from $(2n + 1 > 6)$ we can reason as follows:

$$
\begin{array}{ll}
& 2n + 1 > 6 \\
\to & \quad \{ \quad \text{subtract 1 from both sides} \quad \} \\
& 2n > 5 \\
\to & \quad \{ \quad \text{divide both sides by 2} \quad \} \\
& n > 5/2 \\
\to & \quad \{ \quad 5/2 > 2, \text{ and } > \text{ is transitive} \quad \} \\
& n > 2
\end{array}
$$

Therefore, since we have shown that $n > 2$ must be true under the assumption that $2n + 1 > 6$, therefore $(2n + 1 > 6) \to (n > 2)$. □

10.0.4 $\forall x : D.P(x)$

To prove a "universally quantified" statement $\forall x : D.\ P(x)$:

1. Let c stand for an *arbitrary* element of the domain D, and prove $P(c)$. *Arbitrary* means we don't assume *anything* about c except the fact that it is in the domain D. This means that the same proof will work for every single element in D.

   Warning: note it is common practice to use the same variable name for x and c.

2. Use induction (we will talk about this later in the course).

10.0.5 $\exists x : D.P(x)$

To prove an "existentially quantified" statement $\exists x : D.\ P(x)$:

1. Pick a specific c (a "witness") in the domain D and prove $P(c)$.
2. Use a proof by contradiction.

Example. Prove: if n is an odd integer, then n^2 is also odd.

First step: translate to predicate logic! Notice that it's talking about a variable n. It is really making a statement about *all possible values* of n. This is common in mathematics, to make a "for all" statement just by mentioning some variables without explicitly saying a word like "all" or "every". So the correct translation is

$$\forall n{:}\mathbb{Z}.\ \mathrm{Odd}(n) \to \mathrm{Odd}(n^2).$$

Now, what does $\mathrm{Odd}(n)$ mean?

Definition 10.1. An integer n is *odd* if there is some integer k such that $n = 2k + 1$.

That is, $\mathrm{Odd}(n) = \exists k{:}\mathbb{Z}.\ n = 2k + 1$.

We could also define odd as "not even", but this more positive version will be easier to work with. (Of course, we could prove that the two definitions are equivalent—as a nice challenge you might like to try proving that $\forall n : \mathbb{Z}.\ \mathrm{Odd}(n) \leftrightarrow \neg\,\mathrm{Even}(n)$. One direction is not too hard to prove; the other direction will require something called the Division Algorithm which we will learn about later.)

So let's prove $\forall n : \mathbb{Z}.\ \mathrm{Odd}(n) \to \mathrm{Odd}(n^2)$. We start by introducing an arbitrary integer n; then we have to prove $\mathrm{Odd}(n) \to \mathrm{Odd}(n^2)$. This is an implication, so we prove it by supposing that $\mathrm{Odd}(n)$ is true, and then proving in that case $\mathrm{Odd}(n^2)$ is true as well. We can expand the definition of Odd to figure out what our assumption $\mathrm{Odd}(n)$ means and what we have to show for $\mathrm{Odd}(n^2)$. The whole thing goes like this:

Proof. To show $\forall n{:}\mathbb{Z}.\ \mathrm{Odd}(n) \to \mathrm{Odd}(n^2)$, Let k be an arbitrary integer; then we must show that $\mathrm{Odd}(k) \to \mathrm{Odd}(k^2)$.

> To show $\mathrm{Odd}(k) \to \mathrm{Odd}(k^2)$, suppose $\mathrm{Odd}(k)$ is true, that is, there exists an integer j such that $k = 2j + 1$. Then we must show that $\mathrm{Odd}(k^2)$ is true, that is, there exists an integer p such that $k^2 = 2p + 1$.
>
> > $k^2 = (2j + 1)^2 = 4j^2 + 4j + 1 = 2(2j^2 + 2j) + 1$, so we can pick $p = 2j^2 + 2j$, which is an integer since k is an integer.
>
> Therefore, $\mathrm{Odd}(k) \to \mathrm{Odd}(k^2)$.

Therefore, since k was arbitrary, $\mathrm{Odd}(n) \to \mathrm{Odd}(n^2)$ for all integers n. □

The above proof has a lot of detail to help us keep track of what is going on. A more experienced mathematician might write something more like this:

Proof. Let n be an odd integer, and suppose $n = 2k+1$. Then $n^2 = (2k+1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$, so n^2 is odd as well. □

But you are welcome and encouraged to include a lot of detail about what you are doing especially as you are starting out.

11 More proof examples; proof by contradiction (Friday 14 February)

Example. Prove that if n is an integer and $3n + 2$ is odd, then n is odd.

Again, the first step is to translate into predicate logic:

$$\forall n : \mathbb{Z}.\ \text{Odd}(3n + 2) \to \text{Odd}(n).$$

Let's try proving it.

Proof. Let n be an arbitrary integer. We must show $\text{Odd}(3n+2) \to \text{Odd}(n)$.

> To show $\text{Odd}(3n + 2) \to \text{Odd}(n)$, suppose $3n + 2$ is odd; we must show n is odd as well.
>
> > Since $3n + 2$ is odd, by definition there is some integer k such that $3n + 2 = 2k + 1$. Solving for n, we find that $n = \frac{2k+1}{3}$... but it is unclear where to go from here. We need to show that n is of the form $2j + 1$ for some j, but it doesn't really look like that. We declare this proof attempt a failure.

□

However, we have another technique at our disposal for proving an implication: prove the contrapositive! (Note this proof uses $\neg\,\text{Odd}(n) \to \text{Even}(n)$, which we technically haven't proved, but we will assume it for now!)

Proof. Let m be an arbitrary integer; we must show $\text{Odd}(3m + 2) \to \text{Odd}(m)$.

> We will do this by showing the contrapositive, $\neg\,\text{Odd}(m) \to \neg\,\text{Odd}(3m+2)$, that is, $\text{Even}(m) \to \text{Even}(3m + 2)$.
>
> > So suppose $\text{Even}(m)$, that is, there exists an integer k such that $m = 2k$. We must show $\text{Even}(3m+2)$, that is, $3m+2 = 2j$ for some integer j.
> >
> > > We calculate as follows: $3m+2 = 3(2k)+2 = 6k+2 = 2(3k+1)$. Thus, $j = 3k + 1$ is the desired integer such that $3m + 2 = 2j$, so $\text{Even}(3m + 2)$.
> >
> > Thus, since we were able to show $\text{Even}(3m+2)$ under the supposition that $\text{Even}(m)$, therefore $\text{Even}(m) \to \text{Even}(3m + 2)$.
>
> Therefore, the contrapositive $\text{Odd}(3m + 2) \to \text{Odd}(m)$ is also true.

Since m was an arbitrary integer, this shows that $\forall n : \mathbb{Z}.\ \text{Odd}(3m + 2) \to \text{Odd}(m)$. □

11.1 Proof by contradiction

To prove p, prove $\neg p \to \mathsf{F}$.

Most of the other proof techniques we have seen so far have applied to a specific logical connective like $\wedge$, $\vee$, $\to$, and so on. This technique, however, is a general technique that may be useful in proving any statement. The idea is that in order to prove p, we may suppose that p is false, that is, suppose $\neg p$, and from it derive a contradiction; this means that p must have been true in the first place.

Note $\text{Rational}(x) = \exists p{:}\mathbb{Z}.\ \exists q{:}\mathbb{Z}.\ (x = p/q)$.

Example. Prove that $\sqrt{2}$ is irrational.

First, translate to formal logic notation: $\neg(\text{Rational}(\sqrt{2}))$.

This is a classic proof by contradiction[1] discovered by the ancient Greeks.

Proof. By contradiction. Suppose that $\sqrt{2}$ *is* rational. Then by definition there exist integers a and b, with $b \neq 0$, such that $a/b = \sqrt{2}$. We may further assume that a and b have no common factors, that is, a/b is reduced to lowest terms (if it were not, we could just cancel common factors until none are left, which would not affect equality with $\sqrt{2}$). We can reason as follows:

$$
\begin{array}{lll}
& a/b = \sqrt{2} & \\
\to & & \{ \quad \text{Multiply both sides by } b \quad \} \\
& b\sqrt{2} = a & \\
\to & & \{ \quad \text{Square both sides} \quad \} \\
& 2b^2 = a^2 & \\
\to & & \{ \quad \text{Definition} \quad \} \\
& a^2 \text{ is even} & \\
\to & & \{ \quad \text{Contrapositive of lemma proved in class} \quad \} \\
& a \text{ is even} &
\end{array}
$$

Since a is even, there must be an integer c such that $a = 2c$. Continuing,

$$
\begin{array}{lll}
& 2b^2 = a^2 & \\
\to & & \{ \quad \text{Substitute } a = 2c \quad \} \\
& 2b^2 = (2c)^2 & \\
\to & & \{ \quad \text{Expand} \quad \} \\
& 2b^2 = 4c^2 &
\end{array}
$$

[1] Advanced footnote: sometimes, especially when thinking in terms of computation, one wants to use a so-called *constructive* logic, which allows only constructive proofs, that is, proofs which actually show how to find things which are claimed to exist. Such logic necessarily does away with proof by contradiction (and a few other equivalent principles, such as the fact that $\neg p \vee p \equiv \mathsf{T}$). However, this proof is still a valid proof in a constructive logic: it is actually proving a statement of the form $\neg\text{Rational}(\sqrt{2})$ by proving $\text{Rational}(\sqrt{2}) \to \mathsf{F}$, which is still valid even in constructive logic. So from a pedantic point of view this is not actually a proof by contradiction after all, it is just the normal way that one proves a negation. However, for the purposes of this class we certainly won't be making this distinction; I include it in the notes just as an interesting aside.

$\rightarrow$ { Divide both sides by 2 }
$b^2 = 2c^2$

But this implies—using identical reasoning as for a—that b is even as well. Hence a and b are both even and have a prime factor of 2 in common. But a and b supposdely share no common factors! This is a contradiction, and we may thus conclude that our assumption—namely, that $\sqrt{2}$ is rational—must be false! $\square$

Example. Another example of proof by contradiction (for you to work out): prove that in any right triangle, $a + b > c$, where the length of the hypotenuse is labelled c and the lengths of the two other sides are labelled a and b. (*Hint*: use the Pythagorean Theorem!)

12 No class: Mid-Winter Break (Monday 17 February)

13 No class: Snow day (Wednesday 19 February)

14 Introduction to set theory (Friday 21 February)

14.1 Introduction to set theory

Definition 14.1. A *set* is an unordered, finite or infinite collection of objects, called *elements* or *members* of the set. Sets cannot contain a given element more than once, and the order of elements in a set does not matter. Put another way, the only thing that matters about a given element is whether it is in the set or not.

Remark. We typically use capital letters to stand for sets.

The notation $x \in S$ is a proposition that means "x is an element of set S"; $x \notin S$ is an abbreviation for $\neg(x \in S)$.

14.2 Writing sets

There are two basic ways to write down a set.

1. We can write a (finite) set by listing its elements, separated by commans, within curly braces.
   Example. $S = \{1, 3, 5, 7, 9\}$.

   We can also sometimes abbreviate using dots:
   Example. $S = \{1, 2, 3, \ldots, 100\}$.

2. We can also use *set builder* notation, whose basic form is {value | conditions}.
   Example. $\{x \mid x \text{ is a positive integer less than } 5\}$
   Example. $\{x \mid x \in \mathbb{Z}^+, x < 5\}$
   Example. $\{x \mid x \text{ is an odd integer}, 0 \leq x < 100\}$
   Example. $\{2x + 5 \mid x \in \mathbb{N}, 0 \leq x \leq 10\} = \{5, 7, \ldots, 25\}$

14.3 Some special sets

There are several sets that occur so frequently we give them special names.

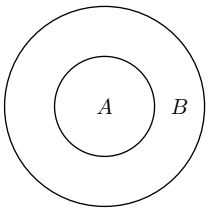
- $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ is the (infinite) set of all *natural numbers*. (Note controversy.)
- $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ is the set of *integers*. (Z stands for *Zahlen*, which means "numbers" in German.)
- $\mathbb{Z}^+ = \{1, 2, 3, \ldots\}$ is the set of *positive integers*.
- $\mathbb{Q} = \{p/q \mid p \in \mathbb{Z}, q \in \mathbb{Z}, q \neq 0\}$ is the set of *rational numbers*.
- $\mathbb{R}$ is the set of *real numbers*.
- $\mathbb{C}$ typically represents the set of *complex numbers*, though we will not use complex numbers in this class.

15 More set theory (Monday 24 February)

15.1 Subsets and equality

Definition 15.1. A is a *subset* of B, written $A \subseteq B$, iff every element of A is also an element of B, that is,

$$\forall a \in A.\ a \in B.$$



Remark. According to the definition, given a set A, is $A \subseteq A$? That is, is a set a subset of itself?

Intuitively, given the way we typically use the word "subset" in English, we might expect the answer to be "no". However, given the definition above, $A \subseteq A$ is true, since every element of A is an element of A.

(Note there is a notion of being a *strict subset*: $A \subset B$, "A is a strict subset of B", means that $A \subseteq B$ but $A \neq B$; however, it is rarely useful.)

Why do we define subset this way? It would actually make things more complicated, with more weird special cases, if we disallowed a set from being a subset of itself.

Now that we have defined subset, we can define equality of sets:

Definition 15.2. $A = B$ iff $(A \subseteq B) \wedge (B \subseteq A)$.

Remark. This is typically how we prove that two sets are equal: show that each is a subset of the other. We might want to do this, for example, in a situation where we have two sets which are described in very different ways, and it is an interesting and nontrivial fact that these two different descriptions actually result in the same set.

Example. Let $A = \{2x + 3 \mid x \in \mathbb{Z}\}$ and $B = \{x \mid x \in \mathbb{Z}, \text{Odd}(x)\}$. Prove that $A = B$.

Proof. By definition, $A = B$ if $A \subseteq B$ and $B \subseteq A$. We will prove both.

- We will first show $A \subseteq B$. By definition, this means $\forall y \in A.y \in B$. So let $a \in A$ be an arbitrary element of A. We must show $a \in B$ as well.
  - By definition of A, if $a \in A$ there must exist some $x \in \mathbb{Z}$ such that $a = 2x + 3$. We can rewrite this as $a = 2(x + 1) + 1$, thus showing that a is odd, and hence $a \in B$.

  Therefore, since $a \in A$ was arbitrary and we showed $a \in B$, therefore by definition $A \subseteq B$.

- Next, we show $B \subseteq A$. ...

□

Definition 15.3. The *empty set*, written $\varnothing$ or $\{\}$, is the set with no elements.

Question: is $\varnothing \subseteq \mathbb{N}$?

By definition, this means $\forall a \in \varnothing.\ a \in \mathbb{N}$, so it really comes down to what we do with a $\forall$ whose domain is empty. In fact, such a $\forall$ must be true, which we can see in a few different ways.

- One way to understand this intuitively is to think about when someone making a quantified statement has lied. If I say "every February the 47th, I fly to the moon", that should be a true statement: I am not lying because there is never any February 47th to prove otherwise.
- We can also recall that we can turn a quantifier with a restricted domain into a quantifier with a larger domain using an implication. In particular,

  $$\forall a \in \varnothing.\ a \in \mathbb{N} \equiv \forall a : D.\ (a \in \varnothing) \to (a \in \mathbb{N})$$

  where D can be any domain we like. $a \in \varnothing$ is always false since $\varnothing$ has no elements; and an implication whose hypothesis is false is always true.

Show how we can use and create sets in Disco. In Disco, the *type* of a finite set with elements taken from the domain (type) `T` is `Set(T)`.

16 Set operations (Wednesday 26 February)

Today we'll learn about some operations we can do on sets.

16.1 Cardinality

The *cardinality* of a **finite** set is the number of distinct elements it contains, *i.e.* a natural number. The cardinality of a set A is written $|A|$.

Example. $|\{1, 3, 5\}| = 3$. $|\{2, 4, \ldots, 100\}| = 50$. $|\varnothing| = 0$.

Notice we can't yet say anything about the cardinality of an **infinite** set!

16.2 Cartesian product

Definition 16.1. If A and B are sets, the *Cartesian product* of A and B is the set of all possible ordered pairs of elements from A and B. That is,

$$A \times B = \{(a, b) \mid a \in A, b \in B\}.$$

Example. Let $A = \{1, 2, 3\}$ and $B = \{\triangle, \square\}$. Then

$$A \times B = \{(1, \triangle), (1, \square), (2, \triangle), (2, \square), (3, \triangle), (3, \square)\}.$$

We can think of this in terms of filling out a 2D grid where the rows are labeled with elements from A and the columns with elements from B:

	$\triangle$	$\square$
1	$(1, \triangle)$	$(1, \square)$
2	$(2, \triangle)$	$(2, \square)$
3	$(3, \triangle)$	$(3, \square)$

Each cell in the grid corresponds to one possible combination of an element from A with an element from B.

From this we can also see that for finite sets, the cardinality of a Cartesian product is the product of the cardinalities:

$$|A \times B| = |A| \times |B|.$$

16.3 Power set

Definition 16.2. The *power set* of a set A, written $\mathcal{P}(A)$, is the set of all possible subsets of A.

Example. Let $A = \{1, 2, 3\}$. Then

$$\mathcal{P}(A) = \{\varnothing, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}.$$

How do we know if we've gotten them all? Notice that to make a particular subset we get to independently choose whether each element will be in the subset or not. So we associate a Boolean variable to each element of the set, and list all possible combinations of T and F, just like we would when making a truth table:

1?	2?	3?	subset
T	T	T	$\{1, 2, 3\}$
T	T	F	$\{1, 2\}$
T	F	T	$\{1, 3\}$
T	F	F	$\{1\}$
F	T	T	$\{2, 3\}$
F	T	F	$\{2\}$
F	F	T	$\{3\}$
F	F	F	$\varnothing$

Or we could just dispense with the booleans and make a table where we either list or don't list each element in all possible combinations:

	1?	2?	3?	
{	1	2	3	}
{	1	2		}
{	1		3	}
{	1			}
{		2	3	}
{		2		}
{			3	}
{				}

This line of reasoning shows us that in general, if A is a *finite* set and $|A| = n$, then $|\mathcal{P}(A)| = 2^n$. Note this even works when A is empty: we have $\mathcal{P}(\varnothing) = \{\varnothing\}$ (since the empty set is the only subset of itself), and $2^0 = 1$.

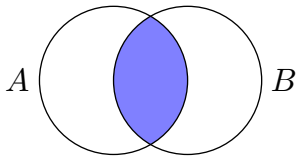
Notice how this is all pointing to the fact that we should allow both $A \subseteq A$ and $\varnothing \subseteq A$. The entire set and the empty set both show up naturally when enumerating every possible combination of true/false values, and we get an elegant formula, 2^n, for the number of subsets including the entire set and the empty set. If we were to disallow these from being subsets, we would end up with a less elegant formula like $2^n - 2$ for the number of subsets, and would have to use the word "except" when describing how to generate all possible subsets. We would essentially be fighting against the universe. The entire set and the empty set *want* to be subsets.

16.4 Intersection

Definition 16.3. The *intersection* of sets A and B is the set of elements common to both:

$$A \cap B = \{x \mid x \in A \wedge x \in B\}.$$

We can visualize this using a Venn diagram:

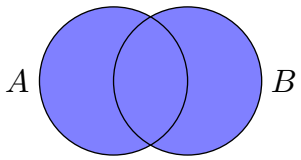


16.5 Union

Definition 16.4. The *union* of sets A and B is the set of elements which are in either:

$$A \cup B = \{x \mid x \in A \vee x \in B\}.$$

We can visualize it as follows:



What can we say about the cardinality of $A \cup B$? First, we can give it some lower and upper bounds:

$$\max(|A|, |B|) \leq |A \cup B| \leq |A| + |B|.$$

But we can also say something exact:

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

If we add the cardinality of A and B, any elements common to both are counted twice, so we have to subtract them in order to find the cardinality of their union (which has no duplicates). This is (a simple special case of) the *Principle of Inclusion-Exclusion*, or **PIE**.

Remark. The notation for intersection and union is not an accident. Notice that $\cap$ is defined in terms of $\wedge$ and $\cup$ is defined in terms of $\vee$; intersection is kind of like "AND for sets" and union is "OR for sets". (Also $\cup$ looks like a "U" for "union".)

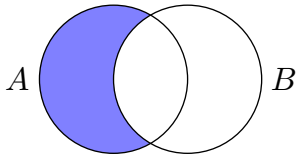
17 More set operations and proofs (Friday 28 February)

17.1 Difference and complement

Definition 17.1. The *difference* of two sets A and B consists of the elements in A but not B:

$$A - B = \{x \mid x \in A \wedge x \notin B\}.$$

We can visualize it like this:

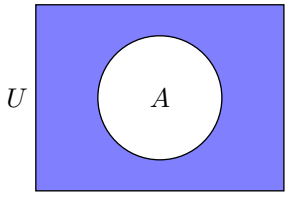


Finally, we want to define the *complement* of a set as the set of things which are *not* in the set. However, we have to be careful: this is a bit nonsensical on its own. For example, if $A = \{1, 2, 3\}$, what elements are in its complement? Surely the number 4 is, but what about my sister? Or Beethoven? Or the small brown rock I picked up and threw into a pond last Tuesday? We have to restrict ourselves to some *universal* set, a "universe of discourse" that contains all the potential elements we are considering; in a given context, all the sets we work with will be subsets of the universal set. (If we are working with *typed* sets, where the elements are all taken from a particular type, then the type itself serves as the universal set.)

Definition 17.2. Given a universal set U, the *complement* of a set A is all the elements of U which are not in A:

$$\overline{A} = U - A = \{x \mid x \in U \wedge x \notin A\}.$$

We can visualize it as follows:



Remark. Set union, intersection, and complement follow laws which are exactly analogous to the logical equivalences for $\vee$, $\wedge$, and $\neg$. For example, $\cup$ and $\cap$ are associative, commutative, and idempotent, and have $\varnothing$ and the universal set U as their identity and/or annihilator; $\cap$ distributes over $\cup$ and vice versa, that is,

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C);$$

there are also analogues of De Morgan's laws, for example,

$$\overline{A \cup B} = \overline{A} \cap \overline{B}.$$

These follow from the fact that each set operation is defined in terms of a corresponding logical operation. Below is one example.

Theorem 17.3. *For all sets A, B, and C,*

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C).$$

Proof. Let A, B, and C be arbitrary sets. We will show $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$, by showing $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$ and vice versa.

> First, we will show $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$. Let $p \in A \cap (B \cup C)$; we must show $p \in (A \cap B) \cup (A \cap C)$.
>
> > Since $p \in A \cap (B \cup C)$, it is an element of both, that is, $p \in A$ and $p \in B \cup C$. Likewise, since $p \in B \cup C$, then either $p \in B$ or $p \in C$.
> > If $p \in B$, then $p \in A \cap B$ because we know $p \in A$ also.
> > Likewise, if $p \in C$, then $p \in A \cap C$.
> > In either case, $p \in (A \cap B) \cup (A \cap C)$ since it must be in either $(A \cap B)$ or $(A \cap C)$.

> Next, we must show $(A \cap B) \cup (A \cap C) \subseteq A \cap (B \cup C)$.
> (similar proof, omitted)

□

Another example one can try to prove is as follows:

Theorem 17.4. *For all sets S and T, $\overline{S \cup T} = \overline{S} \cap \overline{T}$.*

18 Relations (Monday 3 March)

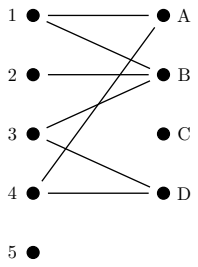
Definition 18.1. Let A and B be sets. A *(binary) relation* from A to B is a subset of $A \times B$.

We write $a \; R \; b$ to denote $(a, b) \in R$ ("a is related to b by R").

Example. Let S be set of students at Hendrix, and C the set of courses offered next year. Let R be the relation

$$\{(s, c) \mid s \text{ is preregistered for } c\}.$$

We might visualize this relation something like this, with students on the left and classes on the right:



We can see that several students (students 1, 3, and 4) are registered for two classes. Class B has three students registered for it, and classes A and D have two students each. Student 5 is either a senior or forgot to register for classes. Class C is a sad class that no one registered for.

Most relations we care about are actually from a set to itself.

Definition 18.2. A *relation on a set* A is a relation from A to A (that is, a subset of $A \times A$).

Example. Here are some example relations on $\mathbb{N}$:

- $R_1 = \{(a, b) \mid a \leq b\}$
- $R_2 = \{(a, b) \mid a > b\}$
- $R_3 = \{(a, b) \mid a = b\}$
- $R_4 = \{(a, b) \mid a = b + 1\}$
- $R_5 = \{(a, b) \mid a + b \leq 10\}$
- $R_6 = \{(a, b) \mid a \text{ divides } b\}$
- $R_7 = \{(a, b) \mid a, b \text{ end with the same digit in base } 10\}$

And a few more example relations over other sets:

- $R_8 = \{(P, Q) \mid P, Q \text{ propositions }, P \to Q\}$
- $R_9 = \{(P, Q) \mid P, Q \text{ propositions }, P \leftrightarrow Q\}$
- $R_{10} = \{(S, T) \mid S \subseteq T\}$

Many relations we care about have one or more nice properties.

Definition 18.3. A relation R on a set A is *reflexive* if

$$\forall a \in A.\, a\ R\ a,$$

that is, every element is related to itself.

Of our example relations, the $\leq$ (R_1), equality (R_3), divisibility (R_6), last digit (R_7), implication (R_8), biconditional (R_9) and subset (R_{10}) relations are reflexive, whereas the others are not. Note that there exist natural numbers which are related to themselves under R_5; for example, $2\ R_5\ 2$ since $2 + 2 \leq 10$. However, it is not the case that *every* natural number is related to itself under R_5 (for example, $7 + 7 \not\leq 10$), so R_5 is not reflexive.

Definition 18.4. A relation R on a set A is *symmetric* if

$$\forall a, b \in A.\, a\ R\ b \to b\ R\ a,$$

that is, whenever a and b are related, then b and a are also related.

The only symmetric relations among our examples are R_3 (equality), R_5, R_7, and R_9.

Definition 18.5. A relation R on a set A is *antisymmetric* if

$$\forall a, b \in A.\, (a\ R\ b) \wedge (b\ R\ a) \to a = b,$$

that is, the only way we can possibly have both $a\ R\ b$ and $b\ R\ a$ is when a and b are actually the same. In other words, whenever a and b are different, at most one of $a\ R\ b$ or $b\ R\ a$ can hold. Intuitively, this says that the relation R is as far from being symmetric as possible: there's not even a single pair of values that exhibits symmetry.

R_1 ($\leq$) is clearly antisymmetric: if $a \leq b$ and $b \leq a$ then we definitely have $a = b$. R_{10} (subset) is similarly antisymmetric. R_6 (divisibility) is also antisymmetric on the natural numbers: the only way to have both $a \mid b$ and $b \mid a$ is if $a = b$. R_3 (equality) is antisymmetric but for a boring reason. It may require a bit more headscratching to see, but R_2 and R_4 are also antisymmetric. For example, take R_2 ($>$). For this to be antisymmetric, we must have

$$\forall a, b \in \mathbb{N}.\, (a > b) \wedge (b > a) \to a = b.$$

Is this true? In fact, it is, since it is impossible for $(a > b) \wedge (b > a)$ to ever be true, and an implication is always trivially true when its premise is false.

Note finally that R_8 is *not* antisymmetric: if $P \to Q$ and $Q \to P$ then they are logically equivalent, but it does not mean they are exactly the same.

Definition 18.6. A relation R on a set A is *transitive* if

$$\forall a, b, c \in A.\, (a\ R\ b) \wedge (b\ R\ c) \to (a\ R\ c).$$

The only relations among our examples which are *not* transitive are R_4 and R_5. All the rest are transitive. We use the transitivity of these relations all the time! For example, transitivity of $=$ is what allows us to prove $X = Y$ by showing a chain of equalities $X = X_1 = X_2 = \cdots = Y$.

19 Equivalence Relations (Wednesday 5 March)

A very important class of relations is *equivalence relations*, which in some suitable sense "act like equality". We have used several of them this semester already!

Definition 19.1. An *equivalence relation* on a set A is a reflexive, symmetric, transitive relation.

I don't know of a good way to explain why these particular properties are the "right" ones to capture "equality-like" things. But hopefully as we see some examples you will be convinced.

Example. Of course, equality itself is an equivalence relation. We can check that it is indeed reflexive, symmetric, and transitive.

Example. Consider the relation L on the set of all English words, where two words are related if and only if they have the same number of letters. This is an equivalence relation:

- Reflexive? Check, each word has the same number of letters as itself.
- Symmetric? Check, if a has the same number of letters as b, then b has the same number of letters as a.
- Transitive? Check, if a has the same number of letters as b, and b has the same number of letters as c, then a has the same number of letters as c.

Example. What about the example from last time where two numbers were related when they had the same last digit in base 10? This is an equivalence relation too. Reflexivity, symmetry, and transitivity follow straightforwardly from the definition. Later in the semester, we will explore a more general version of this relation.

Example. If and only if, *aka* biconditional ($\leftrightarrow$), is an equivalence relation on propositions.

Example. What about the divisibility relation? It is reflexive and transitive; however, it is not symmetric, so it is not an equivalence relation.

Example. Consider the relation on integers given by

$$\{(a, b) \mid 1 \geq |a - b|\}.$$

In other words, a and b are related when they are "very close", that is, the distance between them is at most 1. This is reflexive and symmetric, but not transitive, so it is not an equivalence relation.

So, what kinds of things can we say about equivalence relations in general? First, let's define the important concept of an *equivalence class*.

Definition 19.2. Let $\sim$ be an equivalence relation on A, and let $a \in A$. The *equivalence class* of a, denoted $[a]$, is the set of all elements of A which are related to a, that is,

$$[a] = \{b \in A \mid a \sim b\}.$$

XXX picture of an equivalence class sitting inside A.

Example. Under the equivalence relation L (words are related if they have the same length), the equivalence class of "dog" is the set of all three-letter English words:

$$[\text{dog}] = \{\text{dog}, \text{cow}, \text{hat}, \text{eat}, \dots\}.$$

This is a finite (but large) set.

Example. Under the equivalence relation $\equiv_4$, the equivalence class of 1 is

$$[1] = \{\dots, -7, -3, 1, 5, 9, \dots\}$$

This is an infinite set.

Suppose we have two elements $a, b \in A$. What can we say about the potential relationship between their equivalence classes, $[a]$ and $[b]$? It turns out there are only two possibilities: either they are exactly the same, or they do not overlap at all.

Theorem 19.3. *Let $\sim$ be an equivalence relation on a set A. For any $a, b \in A$, the following three propositions are logically equivalent:*

1. $a \sim b$
2. $[a] = [b]$
3. $[a] \cap [b] \neq \varnothing$

Proof. We will prove the chain of implications $1 \to 2 \to 3 \to 1$, which (by transitivity of implication!) is enough to show they are all logically equivalent.

- $1 \to 2$. Suppose $a \sim b$. We wish to show that $[a] = [b]$, which we can do by showing that both $[a] \subseteq [b]$ and $[b] \subseteq [a]$. XXX draw a picture

  Let's show that $[a] \subseteq [b]$. Let $x \in [a]$ be an arbitrary element; we will show that $x \in [b]$ also. Since $x \in [a]$, by definition $a \sim x$. We assumed that $a \sim b$; by symmetry this means $b \sim a$ as well. Then, by transitivity, $b \sim x$; so, by definition, $x \in [b]$.

  We omit the argument that $[b] \subseteq [a]$, since it is exactly parallel.

- $2 \to 3$. If $[a] = [b]$, then $[a] \cap [b] = [a] \cap [a] = [a]$ (by idempotence of $\cap$). This cannot be the empty set: since $a \sim a$ by reflexivity, we always have $a \in [a]$, so an equivalence class $[a]$ is never empty.

- $3 \to 1$. Suppose $[a] \cap [b] \neq \varnothing$, which means it has at least one element, call it $x \in [a] \cap [b]$. XXX draw a picture This means $x \in [a]$ and $x \in [b]$, so by definition of equivalence classes we have $a \sim x$ and $b \sim x$. By symmetry and transitivity, we conclude that $a \sim b$.

□

This means that the equivalence classes form a *partition* of A. XXX draw a picture.

Definition 19.4. A *partition* of a set A is a collection of subsets A_i such that

- $\bigcup A_i = A$
- $A_i \cap A_j = \varnothing$ if $i \neq j$
- $A_i \neq \varnothing$

We can check these for the equivalence classes of an equivalence relation:

- The union of all the equivalence classes is the set A, since each element at least shows up in its own equivalence class.
- Two different equivalence classes never intersect.

20 Functions (Friday 7 March)

What is the definition of a function?

20.1 Functions

Definition 20.1. Let A and B be sets. (Note: some books say *nonempty* sets, but there's no particular reason to make this restriction!) A *function* f from A to B, written $f : A \to B$, is a relation from A to B such that every $a \in A$ is related to exactly one $b \in B$.

A is called the *domain* and B is the *codomain*. We write $f(a) = b$ to denote the unique element $b \in B$ that is assigned to a.

We can think of a function like this: XXX drawing

Here is an example that's not a function, since there are multiple arrows coming out of a single element of A. XXX drawing

And here's another example of a non-function: there's an element of A with no arrow coming out of it.

Definition 20.2. The *range* of a function $f : A \to B$ is the set

$$\{b \in B \mid \exists a \in A.\ f(a) = b\},$$

that is, the subset of B consisting of all those b's which are actually the image of (at least one) $a \in A$.

Example. Let $\text{Shp} = \{\diamond, \triangle, \square\}$. Now define a function $g : \text{Shp} \to \mathbb{N}$ by

$$\begin{aligned} g(\diamond) &= 6 \\ g(\triangle) &= 6 \\ g(\square) &= 32. \end{aligned}$$

This is a valid function since every element of Shp is assigned to exactly one element of $\mathbb{N}$. The domain of g is Shp, the codomain is $\mathbb{N}$, and the range of g is the set $\{6, 32\}$.

Example. Let $h : \mathbb{N} \to \mathbb{N}$ be defined by $h(n) = 2n$. The domain and codomain of h are both $\mathbb{N}$; the range is the set of all even natural numbers.

Example. Let $\ell : \mathbb{N} \to \text{Shp}$ be the function which sends all multiples of 3 to $\diamond$, all numbers which are one more than a multiple of 3 to $\triangle$, and all which are two more than a multiple of 3 to $\square$. That is,

$$\begin{aligned} \ell(3n) &= \diamond \\ \ell(3n+1) &= \triangle \\ \ell(3n+2) &= \square \end{aligned}$$

This is a valid function since every natural number falls into exactly one of these three cases.

21 1-1 and onto functions (Monday 10 March)

Start by showing how to define functions in Disco.

Question: what do we need to be true in order to be able to turn a function around into a valid function that goes the opposite direction?

XXX picture

This is a valid function, but if we reverse the direction of all the arrows the result would not be a valid function. There are two problems:

1. Different elements of the domain map to the same element of the codomain. If we reverse the arrows that element will have multiple arrows coming out of it, which is not allowed.

2. Some elements of the codomain are left out. If we turn the arrows around, these will not be assigned to anything.

We can define some special types of functions that don't have these problems.

Definition 21.1. A function $f : A \to B$ is *one-to-one* (*injective*, an *injection*) (your book says "injunction"!?) iff no two different elements of the domain map to the same element of the codomain. That is,

$$\forall a_1 \in A.\, \forall a_2 \in A.\, f(a_1) = f(a_2) \to a_1 = a_2,$$

or equivalently, taking the contrapositive,

$$\forall a_1, a_2 \in A.\, a_1 \neq a_2 \to f(a_1) \neq f(a_2).$$

If we want to prove that a function is injective, we typically use the first formulation: for arbitrary $a_1, a_2 \in A$, we must prove $f(a_1) = f(a_2) \to a_1 = a_2$, which we can do by supposing $f(a_1) = f(a_2)$ and proving that $a_1 = a_2$.

To prove that a function is *not* injective, we need to prove the negation:

$$\begin{array}{ll} & \neg(\forall a_1, a_2 \in A.\, a_1 \neq a_2 \to f(a_1) \neq f(a_2)) \\ \equiv & \qquad \{ \quad \text{De Morgan} \quad \} \\ & \exists a_1, a_2 \in A.\, \neg(a_1 \neq a_2 \to f(a_1) \neq f(a_2)) \\ \equiv & \qquad \{ \quad \text{Implication} \quad \} \\ & \exists a_1, a_2 \in A.\, \neg(a_1 = a_2 \vee f(a_1) \neq f(a_2)) \\ \equiv & \qquad \{ \quad \text{De Morgan} \quad \} \\ & \exists a_1, a_2 \in A.\, a_1 \neq a_2 \wedge f(a_1) = f(a_2) \end{array}$$

That is, we pick a_1 and a_2 which are not equal, and show that $f(a_1) = f(a_2)$.

Example. Which of g, h, and ℓ are injective? For those that are not, could we make another function with the same domain and codomain which is injective?

- g is not injective, since $\diamond \neq \square$ but $g(\diamond) = g(\square) = 6$. We could easily make a different function with the same domain and codomain which is injective. For example, the function which is the same as g on $\triangle$ and $\square$ but sends $\diamond$ to 349.

- h is injective, which we can prove as follows.

  Proof. Let $m, n \in \mathbb{N}$ be arbitrary natural numbers. Then we wish to prove that $h(m) = h(n) \to m = n$. So suppose $h(m) = h(n)$, that is, $2m = 2n$. Then dividing both sides by 2 yields $m = n$, as desired. □

- ℓ is very much not injective (for example, every single multiple of 3 maps to $\diamond$). Moreover, there is no way to make an injective function with the same domain and codomain.

Definition 21.2. A function $f : A \to B$ is *onto* (*surjective*, a *surjection*) if "every element of B is covered", that is,

$$\forall b \in B.\ \exists a \in A.\ f(a) = b.$$

Put another way, f is onto if the range of f is the same as the codomain.

To show something is *not* onto, we just have to show the negation:

$$\begin{array}{ll} & \neg(\forall b \in B.\ \exists a \in A.\ f(a) = b) \\ \equiv & \quad \{ \quad \text{De Morgan} \quad \} \\ & \exists b \in B.\ \forall a \in A.\ f(a) \neq b. \end{array}$$

That is, "there is some $b \in B$ which is not the image of any $a \in A$ under f.

Example. Which of g, h, and ℓ are onto? For those that are not, could we make another function with the same domain and codomain which is onto?

- g is not onto. There are many, many elements of the domain (that is, natural numbers) which are not the output of g for any input. There is no function with the same domain and codomain we could make that would be onto—the domain is too small to cover everything in the codomain!

- h is not onto either. The range of h is only the even natural numbers; all the odd natural numbers are not covered. However, we could make an onto function from $\mathbb{N}$ to $\mathbb{N}$, for example, the *identity* function $f(n) = n$.

- ℓ is onto: every shape is the output of the function for some (actually, many) inputs.

22 Bijections and Countable Sets (Wednesday 12 March)

Definition 22.1. A function $f : A \to B$ which is both one-to-one and onto (injective and surjective) is called *invertible* (*bijective*, a *bijection*). We write $f^{-1} : B \to A$ for the *inverse* of f, that is, the function defined by $f^{-1}(b) = a$ if and only if $f(a) = b$.

Remark. Careful: there is an unfortunate notation overlap here. Usually x^{-1} means $1/x$. But f^{-1} is *not* the same as $1/f$! It just denotes the inverse function of f.

Normally, to prove a function is a bijection we can show it is both 1-1 and onto. Alternatively, it turns out we can show it has an inverse. Careful though, to prove g is the inverse of f we have to prove *both* $g(f(a)) = a$ for all a in the domain of f, *and* $f(g(b)) = b$ for all b in the codomain. (*Exercise*: prove that if there is a function g such that $g(f(a)) = a$ for all a in the domain of f, then f is injective; likewise, if there is g such that $f(g(b)) = b$ for all b in the codomain, then f is surjective.)

S'20: Didn't make it to these examples.

Example. Let $f : \mathbb{Z} \to \mathbb{Z}$ be defined by $f(x) = x + 1$. Is f invertible?

- Check: is it one-to-one? Yes; for any $x, y \in \mathbb{Z}$, if $x + 1 = y + 1$ then $x = y$.
- Onto? Yes, for any $y \in \mathbb{Z}$, $y - 1 \in \mathbb{Z}$ and $f(y - 1) = y$.

Hence f is invertible, and in fact, $f^{-1}(x) = x - 1$.

Example. Is $f : \mathbb{N} \to \mathbb{N}$ defined by $f(n) = \lfloor n/2 \rfloor$ invertible? No, it is not one-to-one.

Example. Is $f : \mathbb{Z} \to \mathbb{Z}$ defined by $f(x) = 3x + 2$ invertible? No, it is not onto.

Example. What about $f : \mathbb{Q} \to \mathbb{Q}$ defined by $f(x) = 3x + 2$? This looks similar to the previous example, but it is a different function! In fact this f is both one-to-one and onto; $f^{-1}(s) = \frac{s-2}{3}$.

You are in a large room with thousands of people. Each person is wearing either a red shirt or a blue shirt. You want to know whether there are more people wearing red shirts, more with blue shirts, or an equal number. You have only five minutes and a microphone. What should you do?

Draw some diagrams of injective, surjective, and bijective functions.

Example. Previously we considered the function $f : \mathbb{N} \to \mathbb{N}$, $f(n) = 2n$; it is not onto. However, if we define $2\mathbb{N}$ as the set of all even natural numbers, then $f : \mathbb{N} \to 2\mathbb{N}$ defined in the same way is indeed a bijection. This is a bit strange though, because $2\mathbb{N} \subseteq \mathbb{N}$, yet there can be a bijection between them, *i.e.* their elements can be matched up.

Infinity really messes with our intuition. With finite sets, if $A \subseteq B$ and they are not equal, then $|A| < |B|$ and their elements can't be matched up. But infinite sets don't work that way. In fact, we have been using the notation $|A|$ for the cardinality of sets, but we don't even know what that should mean for infinite sets.

However, *matching things up* is even more fundamental than counting. It's probably how people started counting in the first place. Let's see if this can guide us to some ideas about the cardinality of infinite sets.

22.1 Countable sets

Definition 22.2. Let A, B be sets. If there exists a bijection $f : A \to B$ then we say A and B have the same cardinality, and write $|A| = |B|$.

If there is an injection $f : A \to B$ then we say $|A| \leq |B|$.

Note that this definition makes sense for finite sets and the intuition we already have for what the cardinality of a finite set is. If there is a bijection between two finite sets, then they definitely have the same (natural number) size; if there is an injection from A to B, then B definitely has to be at least as big as A.

So now let's think about infinite sets.

Definition 22.3. A set A is *countable* if it has the same cardinality as a subset of $\mathbb{N}$. Note that $\mathbb{N}$ is a subset of itself: in particular, we say A is *countably infinite* if $|A| = |\mathbb{N}|$.

Every finite set is countable; the situation looks like this:

$$\begin{array}{lcccccc} A: & a_0 & a_1 & a_2 & a_3 & \dots & a_n \\ & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow \\ \mathbb{N}: & 0 & 1 & 2 & 3 & \dots & n \end{array}$$

The definition says a set is countable if it has a bijection to *any* subset of $\mathbb{N}$, but we often use a subset like $\{0, 1, 2, \dots\}$.

For infinite countable sets, the situation looks the same, but there is no endpoint:

$$\begin{array}{lccccc} A: & a_0 & a_1 & a_2 & a_3 & \dots \\ & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow \\ \mathbb{N}: & 0 & 1 & 2 & 3 & \dots \end{array}$$

We can think of this scenario in several equivalent ways:

- A is countable if it has a bijection to (a subset of) $\mathbb{N}$. (definition)
- A is countable if its elements can be put in a list. This is the same because we can always number the elements of a list $0, 1, 2, \dots$.
- A is countable if we can write (or imagine) a program that prints the elements of A one after another, in such a way that any given element of A will eventually be printed after some finite amount of time (if we are willing to wait long enough).

Example. The set of shapes $\{\triangle, \square, \diamond\}$ is countable since we can make a bijection to the set $\{0, 1, 2\}$:

$$\begin{array}{ccc} \triangle & \square & \diamond \\ \updownarrow & \updownarrow & \updownarrow \\ 0 & 1 & 2 \end{array}$$

More generally, any finite set is countable.

Example. The set $2\mathbb{N}$ of all even natural numbers is countably infinite, since $f(n) = 2n$ is a bijection between $\mathbb{N}$ and $2\mathbb{N}$:

$$\begin{array}{rccccc} \mathbb{N}: & 0 & 1 & 2 & 3 & \dots \\ & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \\ 2\mathbb{N}: & 0 & 2 & 4 & 6 & \dots \end{array}$$

Example. Is the set of integers $\mathbb{Z}$ countable? At first glance it might seem like it's not, since it is infinite in *two* directions and $\mathbb{N}$ is only infinite in *one* direction. And indeed, if we try to list all the elements of $\mathbb{Z}$ in a naive way, it fails: "First list 0, then all the positive integers, then all the negative integers" is not a valid list, since it would take infinitely long to list all the positive integers and we would never get around to any of the negative integers.

But $\mathbb{Z}$ *is* countable, which we can show by reordering the elements in a clever way, for example:

$$\begin{array}{rcccccccc} \mathbb{N}: & 0 & 1 & 2 & 3 & 4 & 5 & 6 & \dots \\ & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \\ \mathbb{Z}: & 0 & 1 & -1 & 2 & -2 & 3 & -3 & \dots \end{array}$$

This will eventually get to any integer we want after some finite amount of time, whether positive or negative. We could figure out an explicit formula for a bijection following this pattern, something like

$$f(n) = \begin{cases} -(n/2) & n \text{ is even} \\ (n+1)/2 & n \text{ is odd} \end{cases}$$

but we don't necessarily need to; just demonstrating the pattern and arguing why it will reach every integer is enough.

Example. Is $\mathbb{N} \times \mathbb{N}$ countable?

The elements of $\mathbb{N} \times \mathbb{N}$ are ordered pairs of two natural numbers, for example, $(3, 5)$. We can imagine them in a 2D grid:

$$\begin{array}{cccc} \vdots & & & \iddots \\ (2,0) & (2,1) & (2,2) & \\ (1,0) & (1,1) & (1,2) & \\ (0,0) & (0,1) & (0,2) & \dots \end{array}$$

At first it seems like the answer should be no. Each row of the 2D grid is the same size as a complete copy of $\mathbb{N}$; and we have infinitely many rows! How could infinitely many copies of $\mathbb{N}$ be the same size as $\mathbb{N}$?

And indeed, if we try listing the elements by row or column it definitely doesn't work. "First, list all the elements in row 0; then, list all the elements in row 1; then row 2; and so on"—we will spend an infinitely long time just listing elements of the form $(0, n)$ and never even get around to pairs starting with anything other than 0.

However, surprisingly, $\mathbb{N} \times \mathbb{N}$ *is* countable! Again, we can show this by listing the pairs in a clever order that guarantees we will reach any specific pair in a finite amount of time. The key idea is to list them by *diagonals*, that is, we list the pairs (x, y) in order of the sum $x + y$. First we list all the pairs with sum 0 (there's only one, $(0, 0)$), then all the pairs with sum 1 (there are two of them, $(1, 0)$ and $(0, 1)$), then the pairs with sum 2, and so on. We end up with a list that looks like this[2]:

$$\begin{array}{rccccccccc} \mathbb{N}: & 0 & 1 & 2 & 3 & 4 & 5 & 6 & \dots \\ & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \updownarrow & \\ \mathbb{N} \times \mathbb{N}: & (0,0) & (1,0) & (0,1) & (2,0) & (1,1) & (0,2) & (3,0) & \dots \end{array}$$

[2]Challenge: work out a formula for this bijection!

23 Countable and uncountable sets (Friday 14 March)

Once we have this idea, we can see that $\mathbb{N} \times \mathbb{N} \times \mathbb{N}$ must also be countable: we can either explicitly list them by "diagonal slices" through the corner of the infinite 3D cube; or we can just observe that we can do the $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ bijection twice to get a bijection $(\mathbb{N} \times \mathbb{N}) \times \mathbb{N} \to \mathbb{N} \times \mathbb{N} \to \mathbb{N}$.[3] Likewise, $\mathbb{N}^4$, $\mathbb{N}^5$, and any finite Cartesian product of $\mathbb{N}$ with itself is also countable.

23.1 Is everything countable?

At this point you might wonder if every infinite set is countable. (But if so, why is there a name for it?) In fact, there are some infinite sets that are *not* countable!

23.2 The rationals are countable!

Let's think about $\mathbb{Q}$, the set of rational numbers. There are a lot, right? In between any two integers there are infinitely many rational numbers. If you pick two rational numbers that are really close together, there are infinitely many rational numbers between those. No matter how far you zoom in there will always be infinitely many rational numbers in the part of the number line you are looking at! It is like infinitely fuzzy fuzz. So surely there are more rational numbers than there are natural numbers, right?

Wrong! A rational number is just a pair of integers: a numerator and a denominator. The function $f : \mathbb{Q} \to \mathbb{Z} \times \mathbb{Z}$ that sends the rational number p/q to the pair (p, q) is injective: if two rational numbers are not the same, $p/q \neq r/s$, then certainly the pairs (p, q) and (r, s) are not the same. So $|\mathbb{Q}| \leq |\mathbb{Z} \times \mathbb{Z}| = |\mathbb{N}|$! If you weren't weirded out by the fact that $|\mathbb{N} \times \mathbb{N}| = |\mathbb{N}|$ before, you should be now.

23.3 Non-countable sets

Are there any non-countable sets? How would we prove it if there were? Let's use our superpower:

$$\begin{aligned}
&\neg\text{Countable}(A) \\
\equiv\ &\neg\exists f : \mathbb{N} \to A.\ \text{Bijective}(f) \\
\equiv\ &\neg\exists f : \mathbb{N} \to A.\ \text{Injective}(f) \wedge \text{Surjective}(f) \\
\equiv\ &\forall f : \mathbb{N} \to A.\ \neg\text{Injective}(f) \vee \neg\text{Surjective}(f)
\end{aligned}$$

So to prove that a set A is not countable, we must show that for *every* possible function f from $\mathbb{N}$ to A, either f is not injective or it is not surjective. Note

[3] Challenge: work out the details of showing that this is valid. For example, we need to know that if $f : A \to B$ is a bijection then we can use it to make a bijection between $A \times C$ and $B \times C$.

that a function from $\mathbb{N}$ to A is the same as an infinite list of values from the set A. So we have to show that every possible infinite list of A either has repeated elements, or leaves some elements out.

Consider the set $[0, 1)$ of real numbers between 0 (inclusive) and 1 (exclusive). Each such real number can be written as an infinite decimal,

$$0.d_1d_2d_3d_4\ldots$$

(note this even works for things with finite decimal expansions because we can stick an infinite number of zeros on the end, like $1/2 = 0.500000000\ldots$).[4]

Now, suppose we have a list of real numbers. It might start something like this:

$$\begin{aligned} r_1 &= 0.d_{11}d_{12}d_{13}d_{14}\ldots \\ r_2 &= 0.d_{21}d_{22}d_{23}d_{24}\ldots \\ r_3 &= 0.d_{31}d_{32}d_{33}d_{34}\ldots \\ r_4 &= 0.d_{41}d_{42}d_{43}d_{44}\ldots \\ &\vdots \end{aligned}$$

Remember that this list represents a function $\mathbb{N} \to [0, 1)$. We claim that no matter what this function is, it is not surjective, that is, there is at least one real number in the interval $[0, 1)$ that gets left out. We're going to build a real number

$$0.c_1c_2c_3c_4\ldots$$

by specifying it digit-by-digit.

- For the first digit, c_1, pick any digit which is not equal to d_{11}, the first digit of the first number in the list.
- For the second digit, c_2, pick any digit which is not equal to d_{22}, the second digit of the second number in the list.
- ...and so on: for digit c_i pick any digit which is not equal to d_{ii}, the ith digit of the ith number in the list.

We now claim that the resulting number $c = 0.c_1c_2c_3c_4\ldots$ is nowhere on the list! It cannot be equal to r_1, because c and r_1 have different first digits. It cannot be equal to r_2, because they differ in their second digits. It cannot be equal to any r_i, because the ith digit of c was chosen in such a way that it is different than the corresponding digit of r_i.

We assumed nothing about our list of real numbers other than the fact that it was a list; we have therefore shown that *any* list of real numbers in the interval $[0, 1)$ must necessarily be incomplete. Therefore, it is not possible to create a

[4] There is also a finicky issue with some real numbers having multiple representations as infinite decimals, for example, $0.5000000000\cdots = 0.4999999999\ldots$, but we're not going to worry about this for now. It's annoying but not too hard to deal with this.

bijection between $\mathbb{N}$ and $[0, 1)$, so the set of real numbers in the interval $[0, 1)$ is uncountable. (Of course, this means the set of real numbers $\mathbb{R}$ is uncountable too.)

S'20: Drew a hierarchy of set cardinalities on the board and talked for a little while about the implications of having non-countable sets and the fact that $\mathcal{P}(A)$ is always bigger than A, and so on

24 Sequences (2.4) (Monday 17 March)

24.1 Sequences

Definition 24.1. A *sequence* is a function from $\mathbb{N}$ (or a subset of $\mathbb{N}$) to a set A. We use subscript notation like a_n, b_n to denote the image of n, *i.e.* the nth term of the sequence.

In other words, a sequence can also be thought of as a list where each element has a number:

$$a_0, a_1, a_2, a_3, \dots$$

The subset of $\mathbb{N}$ serves as a set of *indices*; we often use $\mathbb{N}$ itself, or sometimes $\mathbb{Z}^+ = \{1, 2, 3, \dots\}$, or a finite set like $\{0, 1, 2, \dots, n-1\}$.

Remark. The book uses the notation $\{a_n\}$ to refer to a sequence but it's terrible notation (because it looks like a set, but isn't) and I won't use it, even though it is (somewhat) standard.

Example. Consider the sequence of rational numbers with $a_n = 1/n$, starting with a_1. The sequence starts

$$1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \dots$$

Example. The sequence defined by $a_n = 5$ begins

$$5, 5, 5, 5, 5, \dots$$

Example. The string "hello" can be thought of as a finite sequence of letters, with

$$a_0 = h, a_1 = e, a_2 = l, a_3 = l, a_4 = o.$$

Example. Consider the sequence

$$2, 5, 8, 11, 14, 17, \dots$$

Evidently each term in the sequence is three more than the previous term. This sequence can also be described by $a_n = 2 + 3n$, starting at a_0.

This type of sequence, in which there is a common difference between consecutive terms, is known as an *arithmetic* sequence.

Example. Consider the sequence

$$2, 6, 18, 54, 162, \dots$$

In this case, $a_n = 2 \cdot 3^n$ for $n \geq 0$. This type of sequence, with a common ratio between consecutive terms, is known as a *geometric* sequence.

When we define a sequence via a formula for a_n in terms of n like this, we say it is in *closed form*. Another way to define a sequence is via a *recurrence relation*, or just a *recurrence* for short.

Definition 24.2. A *recurrence relation* for a sequence is a rule expressing a_n using one or more previous terms of the sequence.

Example. What sequences are described by the recurrence

$$a_n = 3a_{n-1}?$$

The *solutions* to this recurrence are all geometric sequences with a common ratio of 3. For example, one such sequence would be

$$1, 3, 9, 27, 81, \ldots$$

Of course, another is the example $2, 6, 18, \ldots$ that we saw previously.

If we specify a value for a_0, say, $a_0 = 2$, then the sequence is determined completely: from a_0 we can compute a value for a_1; then from a_1 we can compute a_2, and so on.

Example. Let $a_0, a_1, \ldots$ be a sequence satisfying the recurrence

$$a_n = 2a_{n-1} + 1$$

with $a_0 = 0$.

We have $a_1 = 2a_0 + 1 = 1$; then $a_2 = 2 \cdot 1 + 1 = 3$; then $a_3 = 7$; and so on. The sequence starts

$$0, 1, 3, 7, 15, 31, 63, \ldots$$

Looking at this we notice that every term of the sequence seems to be one less than a power of two, so we conjecture the closed form

$$a_n = 2^n - 1.$$

First, let's check that this works for a_0, which it does: $2^0 - 1 = 1 - 1 = 0$. Next, let's see if a_n satisfies the recurrence when defined in this way. We can do this by substituting our conjectured closed form for a_n into the recurrence and seeing if the two sides are equal:

$$2a_{n-1} + 1 = 2(2^{n-1} - 1) + 1 = 2^n - 2 + 1 = 2^n - 1 = a_n.$$

Yay! This is in fact a valid proof that this closed form and recurrence define the same sequence.

Example. $a_n = n \cdot a_{n-1}$, $a_0 = 1$. These are the factorial numbers, $a_n = n!$.

Example. Consider the recurrence

$$a_n = 2a_{n-1} - a_{n-2}.$$

In general there are many sequences that satisfy this recurrence. For example, let's pick $a_0 = 3$ and $a_1 = 13$ and see what we get:

$$3, 13, 23, 33, 43, 53, \ldots$$

We could conjecture that $a_n = 10n + 3$ is a closed form for this sequence, and prove it by substituting it for a_n in the recurrence to see if it checks out.

24.2 Some example sequences

Can you describe the following sequences either with a closed form or a recurrence?

- $1, 3, 5, 7, 9, 11, 13, 15, \ldots$

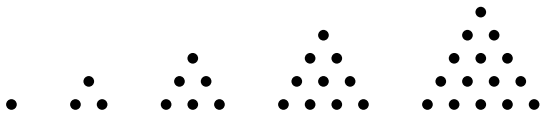
  These are of course the odd numbers. The simplest way to describe them is to notice that each term is two more than the previous term, leading to the recurrence

  $$a_0 = 1$$
  $$a_n = a_{n-1} + 2$$

  Alternatively, we can describe them by the closed form $a_n = 2n + 1$ (if we start at a_0) or $a_n = 2n - 1$ (if we start at a_1).

- $0, 1, 3, 6, 10, 15, 21, \ldots$

  These are the *triangular numbers*, because a_n is the number of dots in an equilateral triangle with n dots on a side:

  Notice how the gap between successive terms increases by 1 every time. This satisfies the recurrence

  $$\triangle_0 = 0$$
  $$\triangle_n = \triangle_{n-1} + n$$

  There is also a closed form for $\triangle_n$ which we'll derive shortly.

- $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$

  These are the famous *Fibonacci numbers*, described by the recurrence

  $$F_0 = 0$$
  $$F_1 = 1$$
  $$F_n = F_{n-1} + F_{n-2}.$$

  There is a closed form for F_n although we probably won't derive it in this class. In fact,

  $$F_n = \frac{1}{\sqrt{5}} (\varphi^n - \hat{\varphi}^n)$$

  where φ and $\hat{\varphi}$ are respectively the positive and negative solutions of $x^2 - x - 1 = 0$, that is, $(1 \pm \sqrt{5})/2$. If you're interested in learning where this comes from, just ask!

S'23: Left out summation this year.

25 Summation and Σ-notation (Wednesday 19 March)

We introduce the notation

$$\sum_{n=i}^{j} a_n$$

to denote the sum of the sequence

$$a_i + a_{i+1} + a_{i+2} + \cdots + a_j.$$

That is, we add up each a_n with n taking on the values i, $i+1$, and so on up to j. n is the *index variable*, i is the *lower limit* and j is the *upper limit*. Σ is a capital Greek letter Sigma, and stands for Σum.

Incidentally, I actually prefer the notation

$$\sum_{i \le n \le j} a_n$$

because we get to reuse familiar notation and we can apply a lot of our intuition and knowledge about dealing with inequalities to successfully manipulating sums.

Example. Write $1 + 3 + 5 + 7 + \cdots + 99$ using Σ-notation.

If we call the first term a_1, this is a sum of terms with closed form $a_n = 2n-1$, and the last term is a_{50}, so we have

$$\sum_{n=1}^{50} (2n - 1).$$

Now, how could we actually evaluate this sum? Let's develop some general techniques and then come back to attack it later.

25.0.1 Sum of a constant

First, notice that

$$\sum_{n=a}^{b} 1 = 1 + 1 + 1 + \cdots + 1 = b - a + 1.$$

Why $b - a + 1$ and not $b - a$, you ask? This is the "fence post problem": if we number the posts of a fence, how many fence posts are there between post a and post b? If there is one post every meter, then the *distance* from post a to post b is certainly $b - a$ meters. But if we try to match up a post to each meter-long section of fence, we find that we have one post left over. For example, if we match up each meter-long section of fence with the post on its left, then the post at the very end is not matched with any section. Hence, the *number of posts* from a to b is one more than the number of sections.

25.0.2 Factoring out constants

$$\sum_n k a_n = k \sum_n a_n$$

This works as long as k is a constant *with respect to* n, that is, k does not contain or depend on n at all. k does not have to be a number; it could be anything at all that does not depend on n. This rule is telling us that if we are adding up a bunch of terms, and every term is k times something, we can factor out all the k's and simply multiply once by k at the end. If we write out what the Σ-notation means, this should become clear:

$$ka_0 + ka_1 + ka_2 + \cdots = k(a_0 + a_1 + a_2 + \dots)$$

Notice, the fact that k does not depend on n is what guarantees that all the k's will all be *the same*, unlike the a_n's which may all be different.

25.0.3 Summing a sum

$$\sum_n (a_n + b_n) = \left(\sum_n a_n\right) + \left(\sum_n b_n\right)$$

This follows because addition is commutative and associative. Written out, the above law is really just saying:

$$(a_0 + b_0) + (a_1 + b_1) + (a_2 + b_2) + \cdots = (a_0 + a_1 + a_2 + \dots) + (b_0 + b_1 + b_2 + \dots)$$

i.e. we are just putting the a_n's and b_n's in a different order.

25.0.4 Triangular numbers

What is the nth triangular number,

$$\triangle_n = \sum_{j=1}^{n} j?$$

We can use a cute trick to compute it. First, write out the sum twice, but reversed the second time. Then add the two equations.

$$\begin{array}{rcccccccccc}
\triangle_n & = & 1 & + & 2 & + & \dots & + & (n-1) & + & n \\
+\triangle_n & = & n & + & (n-1) & + & \dots & + & 2 & + & 1 \\
\hline
2\triangle_n & = & (n+1) & + & (n+1) & + & \dots & + & (n+1) & + & (n+1)
\end{array}$$

We get n copies of $n+1$, so $2\triangle_n = n(n+1)$, and hence

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

25.0.5 Sum of an arithmetic sequence

Example. Compute

$$\sum_{n=1}^{50}(2n-1).$$

We can now compute as follows:

$$\begin{aligned}
&\sum_{n=1}^{50}(2n-1)\\
=&\quad \{\quad \text{sum of a sum}\quad \}\\
&\left(\sum_{n=1}^{50}2n\right)-\left(\sum_{n=1}^{50}1\right)\\
=&\quad \{\quad \text{factor out constant 2, sum of a constant}\quad \}\\
&2\left(\sum_{n=1}^{50}n\right)-(50-1+1)\\
=&\quad \{\quad \text{triangular number formula}\quad \}\\
&2\frac{50(50+1)}{2}-50\\
=&\quad \{\quad \text{arithmetic}\quad \}\\
&50(50+1)-50=50^2+50-50=50^2=2500.
\end{aligned}$$

Example. We could now, in fact, derive a formula for the sum of any arithmetic sequence:

$$\sum_{n=a}^{b}(cn+d)=\dots$$

but I'll let you have the pleasure of working it out!

25.0.6 Sum of a geometric sequence

What about a geometric sequence such as $a_n=r^n$, that is,

$$1+r+r^2+r^3+\cdots+r^n?$$

We can use a similar kind of trick as the one we used to find a closed form for triangular numbers. This time, we multiply by r and then subtract:

$$\begin{array}{rcccccccccccc}
S & = & 1 & + & r & + & r^2 & + & \dots & + & r^n & & \\
rS & = & & & r & + & r^2 & + & \dots & + & r^n & + & r^{n+1}\\
\hline
S-rS & = & 1 & & & & & & & & & - & r^{n+1}
\end{array}$$

Solving for S yields

$$S=\frac{1-r^{n+1}}{1-r}=\frac{r^{n+1}-1}{r-1}.$$

Example. $1+2+2^2+2^3+\cdots+2^n=2^{n+1}-1$.

26 Solving Recurrences (Friday 21 March)

In many situations, it is easy to write down a recurrence relation, but what we really want is a closed form. How can we figure out a closed form for a given recurrence? There are many techniques, but the simplest is to "unfold" the recurrence and look for a pattern.

Example. Let's start by looking at a recurrence for the odd numbers:

$$\begin{aligned} g_0 &= 1 \\ g_n &= g_{n-1} + 2 \qquad\qquad (n \geq 1) \end{aligned}$$

We can start with g_n and keep unfolding it via the recurrence. For example, after unfolding to $g_{n-1} + 2$, the recurrence tells us that $g_{n-1} = g_{n-2} + 2$, so we can unfold again, and so on:

$$\begin{aligned} g_n &= g_{n-1} + 2 \\ &= (g_{n-2} + 2) + 2 \\ &= ((g_{n-3} + 2) + 2) + 2 \\ &= \dots \end{aligned}$$

We see a pattern: g_{n-k} is going to be followed by adding k copies of 2. So in general, we have

$$g_n = g_{n-k} + 2k.$$

Since we know the value of g_0, substituting $k = n$ is helpful:

$$g_n = g_{n-n} + 2n = g_0 + 2n = 1 + 2n.$$

So we have derived the closed form $g_n = 2n+1$ which is what we knew it should be.

Example. As another example, consider the recurrence

$$\begin{aligned} p_0 &= 0 \\ p_n &= 2p_{n-1} + 1 \end{aligned}$$

We can unfold this recurrence as follows:

$$\begin{aligned} p_n &= 2p_{n-1} + 1 \\ &= 2(2p_{n-2} + 1) + 1 \\ &= 2^2 p_{n-2} + 2 + 1 \\ &= 2^2(2p_{n-3} + 1) + 2 + 1 \\ &= 2^3 p_{n-3} + 2^2 + 2 + 1 \\ &= \dots \\ &= 2^k p_{n-k} + 2^{k-1} + 2^{k-2} + \cdots + 2 + 1 \\ &= 2^k p_{n-k} + 2^k - 1 \end{aligned}$$

The last step follows because $2^{k-1} + 2^{k-2} + \cdots + 1$ is the sum of a geometric sequence with ratio $r = 2$. When $k = n$, this becomes

$$p_n = 2^n p_{n-n} + 2^n - 1 = 2^n - 1$$

since $p_0 = 0$.

Example. As another example, consider

$$\begin{aligned} q_0 &= 1 \\ q_n &= 3q_{n-1} + 2 \end{aligned}$$

Unfolding:

$$\begin{aligned} q_n &= 3q_{n-1} + 2 \\ &= 3(3q_{n-2} + 2) + 2 \\ &= 3^2 q_{n-2} + 3 \cdot 2 + 2 \\ &= 3^2(3q_{n-3} + 2) + 3 \cdot 2 + 2 \\ &= 3^3 q_{n-3} + 3^2 \cdot 2 + 3 \cdot 2 + 2 \\ &= \ldots && = 3^k q_{n-k} + 3^{k-1} \cdot 2 + 3^{k-2} \cdot 2 + \cdots + 2 \\ &= 3^k q_{n-k} + 2(3^{k-1} + 3^{k-2} + \cdots + 1) \\ &= 3^k q_{n-k} + 2\frac{3^k - 1}{2} \\ &= 3^k q_{n-k} + 3^k - 1 \end{aligned}$$

When $k = n$, we obtain

$$q_n = 3^n q_0 + 3^k - 1 = 2 \cdot 3^n - 1.$$

We can check that this is correct by (1) computing a few terms of the sequence and checking that our closed form generates the same terms, and/or (2) substituting our closed form into the recurrence to make sure it checks out.

27 No class: Spring break (Monday 24 March)

28 No class: Spring break (Wednesday 26 March)

29 No class: Spring break (Friday 28 March)

30 Induction (Monday 31 March)

When discussing propositional logic, we talked about proof strategies for different types of statements. When proving a forall statement of the form $\forall x \in D.\ P(x)$, the proof strategy we discussed was to let d be an arbitrary element of D and prove $P(d)$. However, for certain domains D there is another way to prove forall statements: induction. In particular, today we will discuss how to prove something for all natural numbers.

Suppose we have a proposition $P(n)$ and we want to prove that $P(n)$ holds for all $n \in \mathbb{N}$. That is, we want to prove all the propositions in the list:

$$P(0) \quad P(1) \quad P(2) \quad P(3) \quad P(4) \quad P(5) \quad \dots$$

We cannot literally prove each one separately, because there are infinitely many. One way we can think about proving them is to arrange them in a chain, like this:

$$P(0) \to P(1) \to P(2) \to P(3) \to P(4) \to P(5) \to \dots$$

The idea is to start by proving $P(0)$ (the "base case") and then "prove the arrows" by proving that each proposition implies the next in the chain—that is, for every $k \geq 0$, $P(k)$ always implies $P(k+1)$ (this is the "inductive step"). We can prove the inductive step by supposing k to be an arbitrary natural number, supposing $P(k)$ is true (this is called the "induction hypothesis", abbreviated IH), and proving that $P(k+1)$ must therefore be true as well.

Another way to think of this is as an infinite chain of dominos:

- Proving the base case corresponds to knocking over the first domino;
- The induction step corresponds to showing that each domino, if it falls down, will knock over the next domino in the line.

In formal predicate logic notation, the principle of induction says

$$P(0) \land (\forall k \in \mathbb{N}.\ P(k) \to P(k+1)) \to (\forall n \in \mathbb{N}.\ P(n)).$$

Note that this involves a $\forall$ inside an $\land$ inside a $\to$! Students often find this confusing. Rely on your skill in manipulating formal propositional logic!

30.1 Example: sum of first n natural numbers

The other day we explored a visual proof that

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

for all n. Now let's give a more formal algebraic proof.

Proof. Let $P(n)$ be the proposition that $1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$. We want to prove $\forall n{:}\mathbb{N}.\ P(n)$. By the principle of induction, we will be able to conclude this if we are able to prove

$$P(0) \land (\forall k{:}\mathbb{N}.\ P(k) \to P(k+1)).$$

Since this is an $\wedge$ we must prove each.

- First we will prove the *base case* $P(0)$, that is, $1 + 2 + \cdots + 0 = \frac{0(0+1)}{2}$. In fact both sides are equal to 0 since the left is an empty sum.
- Now we will prove the *induction case* $\forall k{:}\mathbb{N}.\ P(k) \to P(k+1)$.
  - So let k be an arbitrary natural number; we will prove $P(k) \to P(k+1)$.
  - Suppose $P(k)$, that is, $1+2+3+\cdots+k = \frac{k(k+1)}{2}$. This is our *induction hypothesis*. We must prove $P(k+1)$, that is, $1 + 2 + \cdots + (k+1) = \frac{(k+1)(k+2)}{2}$.

    $$\begin{array}{ll} & 1 + 2 + 3 + \cdots + k + (k+1) \\ = & \quad \{ \quad \text{IH} \quad \} \\ & \frac{k(k+1)}{2} + (k+1) \\ = & \quad \{ \quad \text{algebra} \quad \} \\ & \frac{k^2+k+2k+2}{2} \\ = & \quad \{ \quad \text{algebra} \quad \} \\ & \frac{(k+1)(k+2)}{2} \end{array}$$

□

30.2 Example: growth of 2^n and $n!$

As a second example, consider the predicate $P(n)$ defined as $2^n < n!$. If we try to prove this for all $n \in \mathbb{N}$, we quickly discover that it isn't: $P(0)$ is not true ($2^0 = 1$ which is not less than $0! = 1$); neither is $P(1)$ (2^1 is not less than $1!$), nor $P(2)$, nor $P(3)$. However, all is not lost: $P(4)$ is true ($2^4 = 16 < 4! = 24$), and we suspect that $P(n)$ is true from then on, that is, for all $n \geq 4$. We can still use induction to prove this; we just need to use a different base case.

Theorem 30.1. *$2^n < n!$ for all $n \geq 4$.*

Proof. By induction on n.

- In the base case, $2^4 < 4!$ is true.
- Now let k be an arbitrary natural number such that $k \geq 4$. Suppose $P(k)$ is true, that is,

  $$2^k < k!. \tag{IH}$$

  We wish to show that $2^{k+1} < (k+1)!$ as well.

  $$\begin{array}{ll} & 2^{k+1} \\ = & \quad \{ \quad \text{algebra} \quad \} \\ & 2 \cdot 2^k \\ < & \quad \{ \quad \text{IH} \quad \} \end{array}$$

$$
\begin{array}{lll}
& 2 \cdot k! & \\
< & & \{ \quad 2 < k+1 \text{ since } k \geq 4 \quad \} \\
& (k+1) \cdot k! & \\
= & & \{ \quad \text{definition of factorial} \quad \} \\
& (k+1)! &
\end{array}
$$

So, by induction, $2^n < n!$ for all $n \geq 4$. $\square$

31 Strong induction (Wednesday 2 April)

Sometimes it's difficult—or even impossible—to directly prove $P(k+1)$ from the assumption of $P(k)$. For example, what if we can't prove $P(4)$ from $P(3)$, but we can prove it from $P(2)$? Is that OK? What if we can only prove $P(3)$ by assuming *both* $P(0)$ and $P(1)$? Is that OK?

In general, yes, these are OK: as long as we can prove each proposition from assuming only *previous* propositions, we will eventually be able to prove any of them. This leads to the principle of *strong* induction. Instead of assuming only $P(k)$ as our induction hypothesis and then proving $P(k+1)$, like this:

$$P(k) \to P(k+1)$$

we will instaed assume *all* the previous propositions and use them to prove $P(k+1)$, like this:

$$\big(P(0) \land P(1) \land P(2) \land \cdots \land P(k)\big) \to P(k+1).$$

Written out, a first take on the principle of strong induction thus looks like this:

$$P(0) \land (\forall k \in \mathbb{N}.\ \big(P(0) \land P(1) \land \cdots \land P(k)\big) \to P(k+1)) \to (\forall n \in \mathbb{N}.\ P(n)).$$

Alternatively, we could write it without the ... like this:

$$P(0) \land (\forall k \in \mathbb{N}.\ [\forall j \leq k.\ P(j)] \to P(k+1)) \to (\forall n \in \mathbb{N}.P(n)).$$

We can also make this a bit more general, in two ways:

1. Just like in the example proving $2^n < n!$, the base case need not actually be 0.
2. Sometimes the inductive step might need a certain number of previous cases to "get off the ground". In this case we need more base cases to start.

For a fully general version of strong induction written out using predicate logic, see the video lecture or your textbook. But honestly, it's much more important to understand the ideas behind the above two points. Some examples should help.

Note that despite their names, "strong" induction is not really any "stronger" than normal ("weak") induction. In fact, they are logically equivalent, in the sense that either one can be used to prove the other. This is unsurprising in the case of strong induction implying weak induction: if you get to use strong induction, when you have your IH that says $P(0) \land \cdots \land P(k)$, you may always choose to ignore everything except $P(k)$. What is more surprising is the fact that weak induction implies strong induction. If you want to know more about how this works, see your textbook or ask me.

In the end, then, we can just say "induction" and in some sense it does not really matter whether we are using weak or strong induction. However, it is often helpful to signal to our readers what kind of argument they can expect to see.

31.1 Example: growth of Fibonacci numbers

Recall the definition of the Fibonacci numbers:

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_n &= F_{n-1} + F_{n-2} \qquad \text{when } n \geq 2 \end{aligned}$$

The first few Fibonacci numbers are thus

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots$$

The Fibonacci numbers grow very quickly (for example, the 100th Fibonacci number is already 354224848179261915075). How quickly do they grow? Here's one result that starts to answer this question:

Theorem 31.1. *$F_n \leq 2^{n-1}$ for all $n \in \mathbb{N}$.*

Again, let's try weak induction and see what goes wrong.

Proof attempt.

- In the base case, $F_0 = 0$ which is indeed less than or equal to $2^{0-1} = 1/2$.
- For the induction step, let $k \geq 0$ be an arbitrary natural number, and suppose $F_k \leq 2^{k-1}$; we must show that $F_{k+1} \leq 2^k$. To this end,

$$\begin{aligned} & F_{k+1} \\ = & \qquad \{ \quad \text{definition of Fibonacci numbers}^* \quad \} \\ & F_k + F_{k-1} \\ \leq & \qquad \{ \quad \text{IH} \quad \} \\ & 2^{k-1} + F_{k-1} \end{aligned}$$

  Here is where we get stuck: the IH let us conclude something about F_k but it says nothing about F_{k-1}. And actually, it is worse than that: one of the steps above was actually incorrect! The step marked with an asterisk was not allowed: we wrote F_{k+1} as the sum of two previous Fibonacci numbers, but if $k = 0$ there is no such thing! There is only one Fibonacci number before F_1. Since the induction step needs *two* previous cases, we need to use strong induction, and we also need to start with two base cases so the first induction step has something to get started with.

□

Here's the real proof:

Proof. By strong induction on n.

- The base case for $n = 0$ is as before, but we need a second base case now: $F_1 = 1 \leq 2^{1-1} = 1$.

- Now let $k \geq 1$ (not 0!) be an arbitrary positive integer and suppose as our induction hypothesis that $F_j \leq 2^{j-1}$ for every j from 0 up to k. We must show that $F_{k+1} \leq 2^k$, which we do as follows:

$$
\begin{array}{lll}
& F_{k+1} & \\
= & & \{ \quad \text{definition of Fibonacci numbers (valid since } k \geq 1) \quad \} \\
& F_k + F_{k-1} & \\
\leq & & \{ \quad \text{IH, twice} \quad \} \\
& 2^{k-1} + 2^{k-2} & \\
< & & \{ \quad 2^{k-2} < 2^{k-1} \quad \} \\
& 2^{k-1} + 2^{k-1} & \\
= & & \{ \quad \text{algebra} \quad \} \\
& 2^k &
\end{array}
$$

□

Challenge: can you prove that $(3/2)^n < F_n$ for suitable values of n? Together, these results give us a decent idea of how fast F_n grows.

32 More induction examples (Friday 4 April)

- Quiz today!

32.1 Example: recurrence

Let $a_0 = 0$, $a_n = 2a_{n-1} + 1$. Prove that $a_n = 2^n - 1$ for all $n \geq 0$.

Proof. Let $P(n)$ be the proposition "$a_n = 2^n - 1$". We will show $\forall n : \mathbb{N}.\ P(n)$ by induction.

- $P(0)$ says "$a_0 = 2^0 - 1$", which is true since both sides are equal to 0.
- In the induction step, we must show $\forall k : \mathbb{N}.\ P(k) \to P(k{+}1)$. So let k be an arbitrary natural number, and suppose $P(k)$ is true, that is, $a_k = 2^k - 1$. We must show $P(k+1)$, that is, $a_{k+1} = 2^{k+1} - 1$.

$$
\begin{array}{lll}
& a_{k+1} & \\
= & & \{ \quad \text{definition} \quad \} \\
& 2a_k + 1 & \\
= & & \{ \quad \text{assumption (induction hypothesis)} \quad \} \\
& 2(2^k - 1) + 1 & \\
= & & \{ \quad \text{algebra} \quad \} \\
& 2^{k+1} - 2 + 1 & \\
= & & \{ \quad \text{algebra} \quad \} \\
& 2^{k+1} - 1 &
\end{array}
$$

□

32.2 Example: sum of odds

If we write down some sums of the first few odd numbers, we notice a pattern:

$$
\begin{aligned}
1 &= 1 \\
1 + 3 &= 4 \\
1 + 3 + 5 &= 9 \\
1 + 3 + 5 + 7 &= 16
\end{aligned}
$$

It seems as though we always get a square number as the sum. In particular, we get the square of the number of odds we added up. We conjecture that the sum of the first n odd numbers is equal to n^2, that is:

Theorem 32.1. *For all $n \geq 0$,*

$$1 + 3 + 5 + \cdots + (2n - 1) = n^2.$$

This is a claim of the form $\forall n \in \mathbb{N}.\ P(n)$, where $P(n)$ represents the statement $1 + 3 + 5 + \cdots + (2n - 1) = n^2$, so we can try proving it by induction.

Proof. By induction on n.

- In the base case, when $n = 0$, we have a sum of zero things on the left-hand side, and 0^2 on the right-hand side, which are both equal to 0.
- For the inductive step, let $k \geq 0$ be arbitrary, and suppose $P(k)$ holds, that is,

  $$1 + 3 + 5 + \cdots + (2k - 1) = k^2. \tag{IH}$$

  We must show $P(k + 1)$, that is,

  $$1 + 3 + 5 + \cdots + (2(k + 1) - 1) = (k + 1)^2.$$

  We reason as follows:

  $$\begin{array}{ll} & 1 + 3 + \cdots + (2(k+1) - 1) \\ = & \quad \{ \quad \text{including one more term of the } \ldots \quad \} \\ & 1 + 3 + \cdots + (2k - 1) + (2(k+1) - 1) \\ = & \quad \{ \quad \text{IH} \quad \} \\ & k^2 + (2(k+1) - 1) \\ = & \quad \{ \quad \text{algebra} \quad \} \\ & k^2 + 2k + 1 \\ = & \quad \{ \quad \text{algebra} \quad \} \\ & (k+1)^2 \end{array}$$

  Hence $1 + 3 + 5 + \cdots + (2(k + 1) - 1) = (k + 1)^2$ which is what we wanted to show.

Therefore, by induction, the sum of the first n odd numbers is n^2 for all $n \in \mathbb{N}$. □

Alternatively, we can use summation algebra:

$$\begin{aligned} \sum_{1 \leq k \leq n} (2k - 1) &= 2 \left(\sum_{1 \leq k \leq n} k \right) - \left(\sum_{1 \leq k \leq n} 1 \right) \\ &= 2 \frac{n(n+1)}{2} - n \\ &= n^2 + n - n \\ &= n^2 \end{aligned}$$

33 Divisibility (Monday 7 April)

We're going to embark on a study of basic number theory, which can be thought of as the study of the sets $\mathbb{N}$ and $\mathbb{Z}$. These sets may seem straightforward—and they are, if all you care about is addition. But as soon as we throw multiplication into the mix, things get very interesting indeed. Number theory—and in particular the interaction between addition and multiplication on the integers—is the basis for, *e.g.* all of modern cryptography, as well as one of the most famous open problems in mathematics, the Riemann hypothesis. Questions about prime numbers, factoring, and solving equations over the integers have fascinated humans for thousands of years.

33.1 Divisibility

With that in mind, recall the notion of one number being a "multiple" of another, or (equivalently) one number "evenly dividing" another number. We'll begin by making this notion precise.

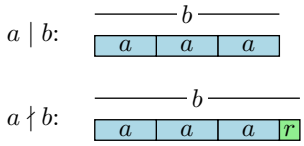
Definition 33.1. If $a, b \in \mathbb{Z}$, we say a *divides* b iff there exists $k \in \mathbb{Z}$ such that $ka = b$.

When a divides b, we also say a is a *divisor* or *factor* of b, and that b is a *multiple* of a. We use the notation

$$a \mid b$$

to denote the fact that a divides b, and write $a \nmid b$ as an abbreviation for $\neg(a \mid b)$.

We can illustrate the situation as follows:



In the situation illustrated at the top, b can be covered exactly with three copies of a, so $a \mid b$. On the other hand, in the bottom illustration, attempting to cover b with copies of a leaves a bit left over; hence $a \nmid b$. Thinking about pictures like this can help us build the right intuition. However, keep in mind that the formal definition is a bit more general than these pictures, since the formal definition allows a and b to be negative, or zero, which the pictures can't really show.

Example.

- $3 \mid 6$ (pick $k = 2$).
- $5 \nmid 16$, since there is no $k \in \mathbb{Z}$ we can pick such that $5k = 16$. (Of course $k = 16/5$ works but that's not an integer.)
- $5 \mid -15$ (pick $k = -3$).

- $4 \mid 4$ (pick $k = 1$).
- $8 \nmid 4$. Note this illustrates the very important point that $\mid$ is not symmetric—in general $a \mid b$ and $b \mid a$ are *not* the same. (In fact, the only case in which $a \mid b$ and $b \mid a$ is when $a = \pm b$.)
- $0 \nmid 3$ since there is no $k \in \mathbb{Z}$ for which $0k = 3$.
- $3 \mid 0$ since we can pick $k = 0$.
- $0 \mid 0$ since we can pick, *e.g.* $k = 671$ to make $0k = 0$ (of course, any $k \in \mathbb{Z}$ will do).

Remark. Notice that there is no need to limit the definition of $a \mid b$ to $a \neq 0$ (as Rosen does). The definition does not actually mention division at all so everything is perfectly well-defined even if $a = 0$.

Theorem 33.2. *Let $a, b, c \in \mathbb{Z}$. Then*

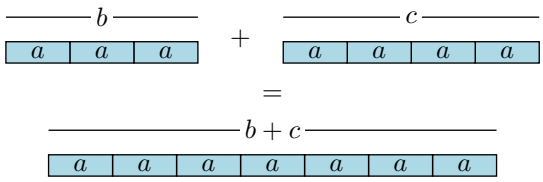
(i) $a \mid a$ *(the divisibility relation is reflexive).*

(ii) if $a \mid b$ *and* $b \mid c$*, then* $a \mid c$ *(divisibility is transitive).*

(iii) if $a \mid b$ *and* $a \mid c$*, then* $a \mid (b + c)$*.*

(iv) if $a \mid b$*, then* $a \mid bc$*.*

Let's prove (i). First, we'll draw a picture:



This isn't quite a proof (because it doesn't really show what happens for negative numbers or zero), but it gives us a good intuition as to why this should be true. If b and c can both be decomposed exactly into copies of a, then their sum will also decompose into copies of a—specifically, the number of copies of a in $b + c$ will be the sum of the number of copies of a in b and the number of copies in c.

Proof. Let $a, b, c \in \mathbb{Z}$ and suppose $a \mid b$ and $a \mid c$. Then by definition there exist $j, k \in \mathbb{Z}$ such that $ja = b$ and $ka = c$. Then

$$b + c = ja + ka = (j + k)a.$$

Thus we have showed how we can write the sum $b + c$ as an integer times a, so by definition $a \mid (b + c)$. □

Remark. By definition we have

$$(a \mid b) \leftrightarrow (\exists k \in \mathbb{Z}.\ k \cdot a = b).$$

But consider that we can characterize $\leq$ similarly:

$$(a \leq b) \leftrightarrow (\exists k \in \mathbb{N}.\ k + a = b).$$

(There is a slight discrepancy because of the need to use $\mathbb{N}$ in the characterization of $\leq$ rather than $\mathbb{Z}$, but if we forget about negative numbers the discrepancy goes away.) That is, $\leq$ is to addition as $\mid$ is to multiplication. The two relations actually share many important properties in common (*e.g.* both are reflexive and transitive, and antisymmetric when restricted to $\mathbb{N}$).

33.2 The Division Algorithm

Recall our picture from above, where a didn't fit exactly into b but we had some small amount left over which we labelled r. We can make this more precise.

Theorem 33.3 (Division Algorithm)**.** *Let $a, d \in \mathbb{Z}$ and $d \in \mathbb{Z}^+$. Then there exist* unique *integers q and r such that $0 \leq r < d$ and $a = dq + r$.*

In this definition q is called the *quotient* and r the *remainder*. We write

$$q = a \,\mathbf{div}\, d$$
$$r = a \,\mathbf{mod}\, d$$

The **div** operator is written `/` in Java and C/C++, and `//` in Python; the **mod** operator is written `%` in many programming languages.

Example.

- What is the quotient and remainder when 101 is divided by 11?

  According to the division algorithm, there exist q and r such that $101 = 11q + r$ and $0 \leq r < 11$. We find that $101 = 11 \cdot 9 + 2$, so $q = 9$ and $r = 2$. We could also express this as

  $$101 \,\mathbf{div}\, 11 = 9$$
  $$101 \,\mathbf{mod}\, 11 = 2$$

- 55 divided by 11? $q = 5$, $r = 0$.

- What is $7 \,\mathbf{mod}\, 11$? This situation is sometimes confusing for students just learning about the `%` operator in a programming language, but applying the definition, we seek (q and) r such that $0 \leq r < 11$ and $11 = 7q + r$. The only values that fit the bill are $q = 0$ and $r = 7$. In other words, if we try to divide 7 by 11, we find that 0 copies of 11 fit into 7, and we are left with the same 7 we started with.

- What is the result of dividing -24 by 11? If we think purely in terms of "how many copies of 11 will fit" we might think the quotient is -2, but that does not make the remainder come out right! In fact, $-24 = 11 \cdot (-3) + 9$, so the quotient is -3 with a remainder of 9.

Why is the Division Algorithm true? We won't give a formal proof, but note that $a = d \cdot 0 + a$; in other words, we can start with $q = 0$ and $r = a$, but probably a is too big: we need $0 \leq r < d$. Whenever $r \geq d$ we can add one to q which decreases r by d. This must eventually land us in the range $0 \leq r < d$.

S '25: Didn't cover this in class; put it on challenge homework instead.

Using the Division Algorithm, we can finally prove that not Even implies Odd!

Theorem 33.4. $\forall x : \mathbb{Z}.\ \neg\,\text{Even}(x) \leftrightarrow \text{Odd}(x)$.

We'll do the forward direction of the proof. The other direction is easier (using proof by contradiction).

Proof. Let x be an arbitrary integer, and suppose x is not even, that is, there does not exist any integer k such that $x = 2k$. We must show x is odd, that is, there exists an integer j such that $x = 2j + 1$.

By the Division Algorithm, there exist unique integers q and r such that $x = 2q + r$, where $0 \leq r < 2$. So there are only two possibilities for r, namely 0 and 1.

- r cannot be zero, since then $x = 2q$ would be even and we assumed x is not even.
- So r must be 1, in which case $x = 2q + 1$ and by definition x is odd.

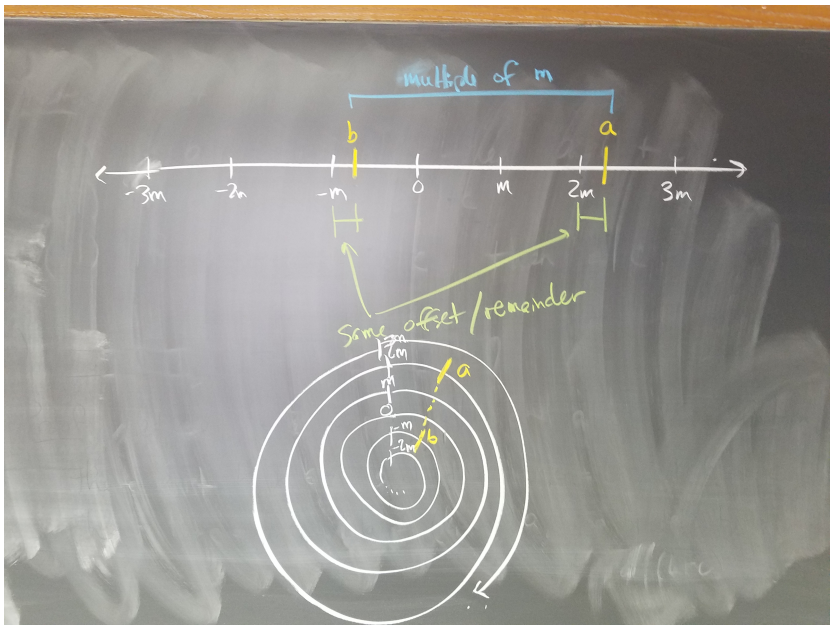
□

34 Modular Equivalence (Wednesday 9 April)

34.1 Modular equivalence

In many situations, we care *only* about the remainder when some number(s) are divided by a particular divisor; for example, when modelling things that repeat cyclically, *i.e.* "loop around". In such cases we might not care exactly how many times something has looped, but only where it currently is in the cycle. This notion is also important when studying divisibility itself: the remainder in some sense measures "how far away" a particular number is from being divisible by the divisor, and keeping track of this for various numbers is often important.

Consider the number line below where we've marked all the multiples of m. If a and b are "at the same place in the cycle" that means they have the same offset from some multiple of m; it also means, intuitively, that their difference is a multiple of m. Since we want to treat every multiple of m as being the same and only care about the remainder or offset between multiples of m, we can think of "curling up" the number line onto itself into an infinite spiral, so that all the multiples of m line up. Then we want to consider all the numbers lined up along any radial line as being "the same".



These ideas motivate the following definition:

Definition 34.1. If $a, b \in \mathbb{Z}$ and $m \in \mathbb{Z}^+$, then a *is congruent to* b *modulo* m, written

$$a \equiv_m b,$$

if and only if

$$m \mid (a - b).$$

Remark. People usually use the notation

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

instead of $a \equiv_m b$. But this notation is so bad that we will not use it. This $\pmod{m}$ is not the same thing as the **mod** operator we introduced earlier. It is not doing anything to b. This is just the way that we indicate that a and b are equivalent "in the $\pmod{m}$ world", that is, if we only care about the remainder after dividing by m.

$\pmod{m}$ is standard notation; however, I prefer $\equiv_m$ because it makes it more clear that we are talking about a specific kind of relationship.

Let's continue studying equivalence modulo m, that is, $\equiv_m$. First, let's prove a couple theorems giving us alternative ways to characterize it.

Theorem 34.2. *Let $a, b \in \mathbb{Z}$ and $m \in \mathbb{Z}^+$. Then $a \equiv_m b$ if and only if there exists $k \in \mathbb{Z}$ such that $a = b + km$.*

Proof.

$$\begin{array}{lll} & a \equiv_m b & \\ \leftrightarrow & & \{ \quad \text{Definition of } \equiv_m \quad \} \\ & m \mid (a - b) & \\ \leftrightarrow & & \{ \quad \text{Definition of } \mid \quad \} \\ & \exists k \in \mathbb{Z}.\ km = a - b & \\ \leftrightarrow & & \{ \quad \text{algebra} \quad \} \\ & \exists k \in \mathbb{Z}.\ a = b + km & \end{array}$$

□

Theorem 34.3. *Let $a, b \in \mathbb{Z}$ and $m \in \mathbb{Z}^+$. Then $a \equiv_m b$ if and only if*

$$a \bmod m = b \bmod m.$$

S'20: Didn't do this proof in class.

Proof. This is an "if and only if", so we have to prove both directions.

($\rightarrow$) Suppose $a \equiv_m b$. Then, by Theorem 34.3, $a = b + km$ for some integer k. By the Division Algorithm, we can write $a = q_1 m + r_1$ and $b = q_2 m + r_2$ with $0 \leq r_1, r_2 < m$; we wish to show that $r_1 = r_2$. Substituting for a and b in the equation $a = b + km$ yields

$$q_1 m + r_1 = q_2 m + r_2 + km.$$

Rearranging,

$$r_1 - r_2 = (q_2 - q_1 - k)m,$$

which means that $(r_1 - r_2)$ is divisible by m. However, we know that $0 \leq r_1, r_2 < m$, and so $-m < r_1 - r_2 < m$. In order for $r_1 - r_2$ to be a multiple of m, the only possibility is that $r_1 - r_2 = 0$, meaning that $r_1 = r_2$.

($\leftarrow$) Suppose $a \mathbf{mod} m = b \mathbf{mod} m$. If we write $a = q_1 m + r_1$ and $b = q_2 m + r_2$ by the Division Algorithm, the assumption that $a \bmod m = b \bmod m$ means $r_1 = r_2$. If we subtract a and b, the r's cancel and we get

$$a - b = (q_1 m + r) - (q_2 m + r) = (q_1 - q_2)m.$$

Thus $m \mid (a - b)$ and therefore, by definition, $a \equiv_m b$. □

Now let's verify that $\equiv_m$ behaves the way we would expect an equality-like thing to behave.

Theorem 34.4. *If $a \in \mathbb{Z}$ and $m \in \mathbb{Z}^+$, then*

$$a \equiv_m a$$

($\equiv_m$ is reflexive*).*

Proof. $a = a + 0m$, so by Theorem 34.3, $a \equiv_m a$. □

Theorem 34.5. *Let $a, b, c \in \mathbb{Z}$ and $m \in \mathbb{Z}^+$. If $a \equiv_m b$ and $b \equiv_m c$, then $a \equiv_m c$ ($\equiv_m$ is* transitive*).*

Proof. Again by Theorem 34.3, since $a \equiv_m b$ and $b \equiv_m c$ we can find $j, k \in \mathbb{Z}$ such that $a = b + jm$ and $b = c + km$. Substituting,

$$a = b + jm = (c + km) + jm = c + (k + j)m,$$

and hence by Theorem 34.3 $a \equiv_m c$. □

Theorem 34.6. *For any $a, b, c, d \in \mathbb{Z}$ and $m \in \mathbb{Z}^+$, if $a \equiv_m b$ and $c \equiv_m d$, then*

(i) $a + c \equiv_m b + d$, and

(ii) $ac \equiv_m bd$.

(that is, $\equiv_m$ is a congruence *with respect to addition and multiplication).*

Proof. (i) By Theorem 34.3 we can write $a = b + jm$ and $c = d + km$. Then

$$a + c = (b + jm) + (c + km) = (b + c) + (j + k)m,$$

and thus by Theorem 34.3 again, $a + c \equiv_m b + d$.

(ii) This time we reason as follows:

$$ac = (b + jm)(c + km) = bc + m(\text{stuff}),$$

that is, when we expand the product, we get one bc term and several other terms that all have an m we can factor out. We don't actually need to do all the algebra—we can see that ac will be bd plus m times some integer, so $ac \equiv_m bd$.

□

Remark. Note that when we have an *equality* $a = b$, we can do whatever we want to it: as long as we do the same thing to both sides they will remain equal. With $\equiv_m$, however, this only works for certain operations! The things we do to an equivalence have to "play nice" with the operation of taking remainders modulo m.

From the proof above, we can see the special way addition and multiplication interact with remainders. For example, if we tried to do a similar proof for division, we would get stuck: there is no reason to believe that $(b+jm)/(c+km)$ would be of the form $\frac{b}{c} + lm$ for some integer l.

35 Solving modular equivalences and modular arithmetic (Friday 11 April)

Example. Solve for x:

$$x + 7 \equiv_3 12.$$

Since $\equiv_3$ is reflexive, we know $-7 \equiv_3 -7$. Then by Theorem 34.1, we can add these two equivalences to conclude

$$x + 7 + (-7) \equiv_3 12 + (-7)$$

and hence

$$x \equiv_3 5.$$

"$x \equiv_3 5$" is a valid solution; we could also write $x \equiv_3 2$ which is a bit simpler, and equivalent since $2 \equiv_3 5$. Note $x \equiv_3 2$ is really a way of expressing infinitely many valid solutions for x: it means that

$$x \in \{\dots, -4, -1, 2, 5, 8, 11, \dots\}.$$

We do not usually need to be this pedantic when solving modular equivalences. It would be acceptable to just write the following: "Starting from $x + 7 \equiv_3 12$, we can subtract 7 from both sides, yielding $x \equiv_3 5 \equiv_3 2$."

Example. Solve for x:

$$101x + 52 \equiv_{10} 68$$

We can first subtract 52 from both sides, yielding

$$101x \equiv_{10} 16 \equiv_{10} 6.$$

It might look like we are stuck at this point: we are not allowed to divide both sides by 101. However, notice that $101 \equiv_{10} 1$, and so $101x \equiv_{10} x$ (since $\equiv_{10}$ is a congruence with respect to multiplication). Hence, we have

$$x \equiv_{10} 6.$$

Example. Solve for x:

$$3x + 19 \equiv_7 2(x - 73)$$

We can solve this as follows:

$$\begin{array}{lll}
\quad 3x + 19 \equiv_7 2(x - 73) & & \\
\rightarrow & \{ & \text{distribute} \quad \} \\
\quad 3x + 19 \equiv_7 2x - 146 & & \\
\rightarrow & \{ & \text{reduce 19 and } -146 \text{ modulo 7} \quad \} \\
\quad 3x + 5 \equiv_7 2x - 6 & & \\
\rightarrow & \{ & \text{subtract 5 from both sides} \quad \} \\
\quad 3x \equiv_7 2x - 11 & & \\
\rightarrow & \{ & \text{subtract } 2x \text{ from both sides} \quad \}
\end{array}$$

$$\begin{array}{ll} & x \equiv_7 -11 \\ \to & \qquad\qquad \{ \quad -11 \equiv_7 3 \quad \} \\ & x \equiv_7 3 \end{array}$$

Example. Solve for x:

$$x + 22 \equiv_{19} 20x + 3$$

Subtracting $x + 3$ from both sides yields

$$19 \equiv_{19} 19x.$$

But since $19 \equiv_{19} 0$, this reduces to

$$0 \equiv_{19} 0,$$

which is always true for any value of x. Hence every integer is a solution for x.

Example. Solve for x:

$$x + 22 \equiv_{19} 20x - 7$$

This is similar to the previous example, but we end up with

$$10 \equiv_{19} 0,$$

which is impossible. Hence in this example there are no solutions for x.

S'21: Used to cover the following about modular arithmetic here (i.e. the set $\mathbb{Z}_n$), but didn't have time for it. Decided to defer it to a later class, if needed. Honestly it might not be needed at all.

35.1 Modular Arithmetic

Definition 35.1. For $m \in \mathbb{Z}^+$, define $\mathbb{Z}_m = \{0, 1, \ldots, m-1\}$.

Note that $|\mathbb{Z}_m| = m$. $\mathbb{Z}_m$ is the set of possible remainders when we divide by m. In fact, $\mathbb{Z}_m$ is the "curled up number line" we drew a picture of the other day. One of the reasons this works nicely is that we can continue to do addition and multiplication on the curled-up number line:

Definition 35.2. For $m \in \mathbb{Z}^+$, define addition $+_m : \mathbb{Z}_m \times \mathbb{Z}_m \to \mathbb{Z}_m$ and multiplication $\cdot_m : \mathbb{Z}_m \times \mathbb{Z}_m \to \mathbb{Z}_m$ as follows:

$$a +_m b = (a + b) \bmod m$$
$$a \cdot_m b = (a \cdot b) \bmod m$$

Example.

- $7 +_{11} 9 = (7 + 9) \bmod 11 = 16 \bmod 11 = 5$.
- $7 \cdot_{11} 9 = (7 \cdot 9) \bmod 11 = 63 \bmod 11 = 8$.

These operations have a number of nice properties that we would expect from things called "addition" and "multiplication":

- Commutative: $+_m$ and $\cdot_m$ are both commutative, *e.g.* $a +_m b = b +_m a$.
- Associative: both operations are associative, *e.g.* $a \cdot_m (b \cdot_m c) = (a \cdot_m b) \cdot_m c$.
- Identities: 0 is the identity element for $+_m$ (that is, $0 +_m a = a +_m 0 = a$) and 1 is the identity element for $\cdot_m$ (that is, $1 \cdot_m a = a \cdot_m 1 = a$).
- Distributivity: $a \cdot_m (b +_m c) = (a \cdot_m b) +_m (a \cdot_m c)$.
- Additive inverses: every $x \in \mathbb{Z}_m$ has an additive inverse, that is, some $y \in \mathbb{Z}_m$ such that $x +_m y = 0$.

A set with two operations having these properties is known as a *commutative ring*; you would learn more about such things in an abstract algebra course.

36 Primes (Monday 14 April)

You all know about prime numbers and prime factorization, but we're going to study them in a bit more rigor than perhaps you've seen before.

Definition 36.1. An integer $p > 1$ is *prime* iff the only factors of p are 1 and p, that is, $a \mid p \to (a = 1 \vee a = p)$. Integers > 1 that are not prime are *composite*.

Remark. 1 is neither prime nor composite! It is special because it's the multiplicative identity.

Example. 2, 3, 5, and 7 are prime. 6 is composite since $2 \mid 6$.

Theorem 36.2 (Fundamental Theorem of Arithmetic (FTA))**.** *Every positive integer n can be written as the product of zero or more primes. The product is unique if the primes are listed in order from smallest to largest.*

There are two things to prove: first, that it is always *possible* to write any positive integer as a product of primes, and second, that the product is always unique. Uniqueness can be proved using techniques from chapter 4 (for a proof, see page 271 of Rosen, Lemma 3). However, the other half requires induction.

Theorem 36.3. *Every integer $n \geq 2$ can be written as a product of primes.*

Note we focus on $n \geq 2$ here; for $n = 1$ we simply note that it is a product of zero primes.

Proof. Let $P(n)$ be the proposition "n can be written as a product of primes". We wish to prove $\forall n \geq 2.\ P(n)$. Let's try a normal proof by (weak) induction and see where it goes wrong.

- For the base case, when $n = 2$, $P(2)$ is clearly true since 2 itself is prime, so it can be written as a "product" of just one prime.
- Let $k \geq 2$ and suppose as our induction hypothesis that k can be written as a product of primes. Then we must show that $k+1$ can also be written as a product of primes.

  If $k + 1$ is prime, then we are done: it can be written as a "product" of one prime (itself). Otherwise, $k+1$ is composite, which means there must be positive integers $2 \leq a \leq b \leq k$ such that $k + 1 = ab$.

  But now we are stuck! Our induction hypothesis only tells us something about k; it says nothing about a and b. The problem is that knowing something about k does not really help us say anything about $k + 1$.

Instead, we can use a proof by strong induction. The base case is the same, but the induction step goes like this:

- Let $k \geq 2$ and suppose as our induction hypothesis that *every number from 2 up to k* can be written as a product of primes. Then we must show that $k + 1$ can also be written as a product of primes.

As before, the interesting case is when $k+1 = ab$ is composite, with $2 \le a \le b \le k$. Now, by the induction hypothesis, we know that a and b can both be written as products of primes; thus, so can $k+1$, since it is the product of a and b.

□

Remark. Note it says *zero* or more primes—what's the product of zero primes? It's 1, of course, the multiplicative identity. This is why the theorem applies to all positive integers, even 1, which can be written uniquely as the product of zero primes. Primes themselves can be written as the product of a single prime.

Theorem 36.4. *If n is composite, it has a prime divisor $\le \sqrt{n}$.*

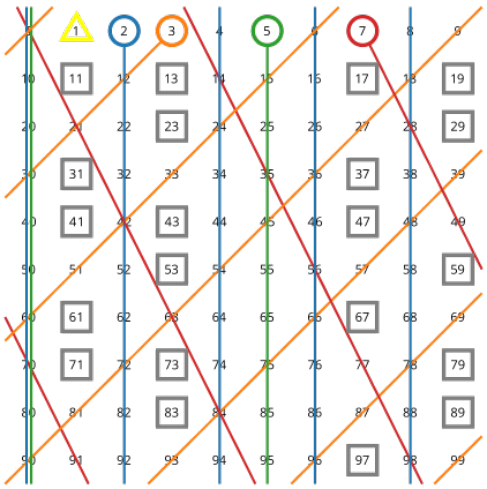
Remark. It's worth spelling out how to write this in formal predicate logic:

$$\forall n \in \mathbb{Z}^+. \text{Composite}(n) \to \exists p \in \mathbb{Z}^+. \text{Prime}(p) \wedge (p \mid n) \wedge p \le \sqrt{n}.$$

The phrase "it has a prime divisor $\le \sqrt{n}$" actually expands into an $\exists$ with a conjunction of three propositions! Each of "prime", "divisor", and "$\le \sqrt{n}$" corresponds to a separate property of the number that is claimed to exist.

Proof. Suppose n is composite. That means there exists a divisor a which is neither 1 nor n. Then in turn that means there must be some b such that $n = ab$, and we note that b also is neither 1 nor n. If both $a > \sqrt{n}$ and $b > \sqrt{n}$, then their product $ab > n$, which is a contradiction—hence either $a \le \sqrt{n}$ or $b \le \sqrt{n}$. Without loss of generality, say $a \le \sqrt{n}$. We are *almost* done but not quite—a is a divisor of n, and $\le \sqrt{n}$, but we don't necessarily know that it is prime. However, by the Fundamental Theorem of Arithmetic, there is some prime p which divides a, and by transitivity of divisibility $p \mid n$, and also $p \le a \le \sqrt{n}$. □

Example. The *sieve of Eratosthenes* is an ancient Greek method for finding all the primes up to some limit. We start by circling 2 as the first prime, then crossing out all multiples of 2. Since the 3 is not crossed out, it must be prime too. We circle it and cross out all multiples of 3. The 4 is already crossed out now, but 5 is not, which means it must be prime—if it were composite it would have some smaller prime divisor, but we have already crossed out all multiples of smaller primes. So we circle 5 and cross out all its multiples; finally we circle 7 and cross out all its multiples. But now we can stop! We have found all the primes up to $\sqrt{100} = 10$. The rest of the numbers up to 10 are already crossed out; by the previous theorem, composite numbers up to 100 must have a prime divisor less than or equal to 10, but we have already crossed out all multiples of primes less than 10. So all the remaining numbers must be prime and we can simply circle them.



Here is another classic piece of mathematics from ancient Greece:

Theorem 36.5. *There are infinitely many primes.*

Proof. By contradiction. Suppose there were only finitely many primes, call them p_1, p_2, ..., p_n. Let Q be one more than the product of all the primes, $Q = p_1 p_2 \dots p_n + 1$. Notice that Q is not divisible by any of the primes, since $Q \equiv 1 \pmod{p}$ for each prime p. But by the FTA, Q must be divisible by a prime! This is a contradiction, and we conclude that in fact there must be infinitely many primes. (*Question to ponder: is the Q in this proof always a prime?*) □

So there is no largest prime; but the currently (as of April 14, 2025) largest *known* prime—that is, the largest number we can specifically point to and say for certain it is prime—is

$$2^{136,279,841} - 1.$$

(See `https://www.mersenne.org/primes/?press=M136279841`.) This number has 41,024,320 decimal digits. If it were printed in a book, with, say, 100 digits per line and 50 lines of digits per page, it would require around 8204 pages—probably a 20-volume set of books taking up a whole shelf!

36.1 Divisibility tests

Example. Is 7431 prime? No, it's divisible by 3 since the sum of its digits is. Many of you perhaps know this divisibility rule, but do you know why it works?

Theorem 36.6. *A number is divisible by* 3 *iff the sum of its digits in base* 10 *is also divisible by* 3.

Note this process can be iterated: for example, 7431 is divisible by 3 if and only if $7+4+3+1 = 15$ is; and 15 in turn is divisible by 3 if and only if $1+5 = 6$ is.

Proof. Let n be any positive integer. First, write n in terms of its base-10 expansion:

$$n = 10^k d_k + 10^{k-1} d_{k-1} + \cdots + 10d_1 + d_0$$

Note first that $10 \equiv_3 1$, since $10 = 3 \cdot 3 + 1$. Since we have proved that equivalence modulo m is compatible with addition and multiplication, we can replace every 10 in the above expansion by 1, and get something that is equivalent modulo 3:

$$n = 10^k d_k + 10^{k-1} d_{k-1} + \cdots + 10d_1 + d_0 \equiv_3 d_k + d_{k-1} + \cdots + d_1 + d_0$$

Thus, we have shown that n is equivalent modulo 3 to the sum of its base-10 digits. □

This also shows why for 2 and 5 we can just look at the last digit: since 2 and 5 are both divisors of 10, we have $10 \equiv_2 0$ and $10 \equiv_5 0$. So replacing 10 with 0 in the expansion for n tells us that $n \equiv_2 d_0$ and $n \equiv_5 d_0$.

What about 7? Let n be a positive integer and write $n = 10m + d_0$. For example, if $n = 7451$ we have $m = 745$ and $d_0 = 1$. We reason as follows:

$$\begin{array}{lll} & 10m + d_0 \equiv_7 0 & \\ \leftrightarrow & & \{ \quad \text{Multiply both sides by } -2 \text{ (see note!)} \quad \} \\ & -20m - 2d_0 \equiv_7 0 & \\ \leftrightarrow & & \{ \quad -20 \equiv_7 1 \quad \} \\ & m - 2d_0 \equiv_7 0 & \end{array}$$

Thus, n is divisible by 7 if and only if $m - 2d_0$ is. That is, we take the last digit of n, double it, and subtract from the number formed by the rest of the digits. For example, doing this process on 7451 yields $745 - 2 \cdot 1 = 743$; doing it again yields $74 - 2 \cdot 3 = 68$, and 68 is not divisible by 7 (it is $7 \cdot 10 - 2$). On the other hand, 7441 yields 742, which yields 70, which is divisible by 7, and indeed, $163 \cdot 7 = 7441$.

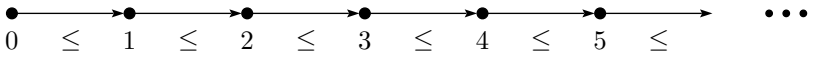
Note, there is actually a key piece missing from our proof above: although we proved that we can always multiply both sides of a modular equivalence by the same number (-2 in this case), we don't yet know that we can *divide* both sides by the same number, which we need for that step to be an *if and only if*. In fact, sometimes we can't! But it turns out this step is allowed since 2 and 7 share no common factors. We will prove this later in the week.

37 GCD and the Euclidean Algorithm (Wednesday 16 April)

What is the GCD of 60 and 18? Of 7169 and 7811?

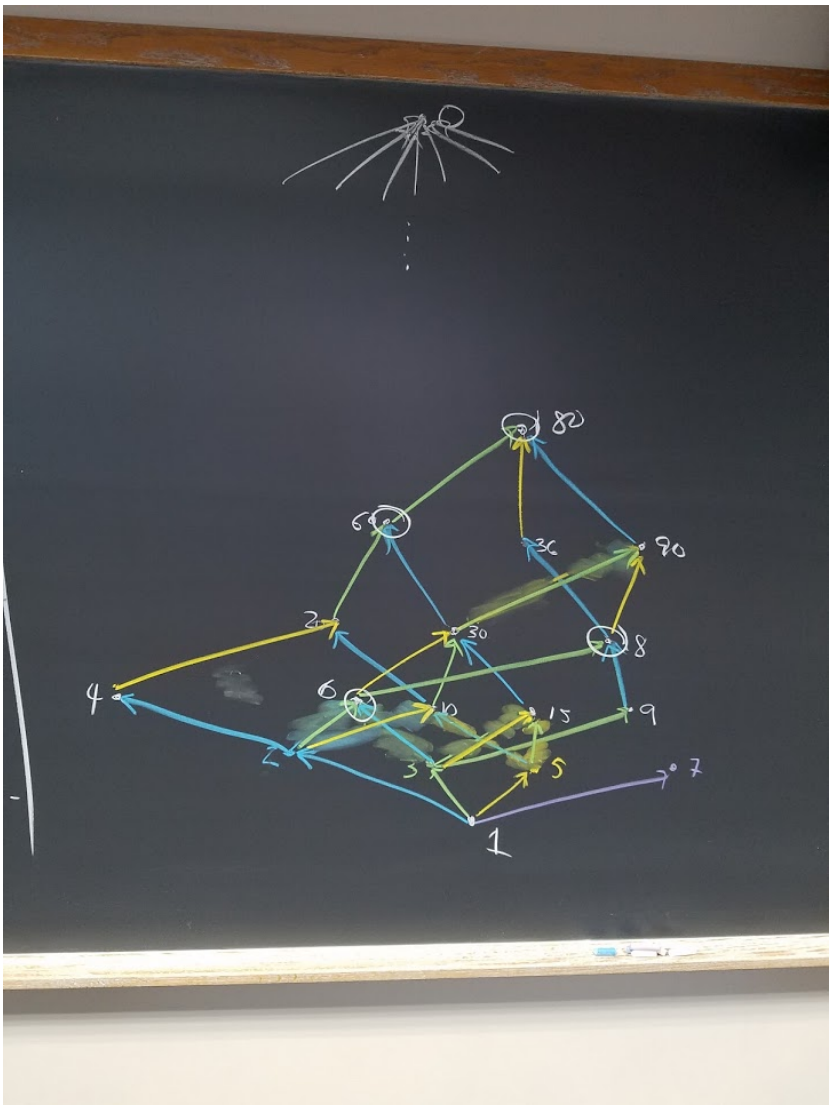
37.1 The divisibility lattice

We are used to thinking of the natural numbers like this:



They are arranged along a number line, in order according to the $\leq$ relation. Here we've drawn an arrow from each number to the next. Notice that because $\leq$ is transitive, there is a path following the arrows from one number to another iff the first is less than or equal to the second. We've seen before that $\leq$ is intimately related to addition, so we can think of this chain of arrows as a picture of the additive structure of $\mathbb{N}$.

The divisibility relation $a \mid b$, on the other hand, gives us a way to think about the *multiplicative* structure of the natural numbers. Let's draw a similar kind of diagram of the natural numbers where a and b are connected by an arrow whenever $a \mid b$. (However, ff $a \mid b \mid c$ then we won't bother drawing an arrow $a \to c$: since divisibility is transitive, we can just follow the path $a \to b \to c$ to see that $a \mid c$. However, abstractly the arrow is still "there", we just won't draw it.)



Obviously this is just a tiny part of the entire (infinite) diagram. Here are a few observations:

- This diagram is obviously much more complicated than the diagram for $\leq$! This is, in some sense, the reason why studying the multiplicative structure of $\mathbb{N}$ is much more interesting than studying the additive structure.
- All the primes are at the bottom, right above 1. Composite numbers come in "layers" above that depending on how many prime divisors they have.
- The *greatest common divisor* of two numbers a and b is the highest number in the diagram from which we can travel up to both a and b. For example,

we can see that the GCD of 60 and 18 is 6, since from 6 there is a path along arrows upwards to both 60 and 18, and there is no higher node with this property.

- Likewise, the *least common multiple* of a and b is the lowest number we can reach from both a and b. In this example, the LCM of 60 and 18 is 180.
- Where should 0 go in this diagram? In fact, it has to go at the "top", since every number divides 0, hence there is an arrow from every natural number up to 0.
- Generally speaking, numbers get bigger as we go up the diagram; but it is not true that higher numbers are always greater than lower ones. For example, 35 would go in the second layer from the bottom, but $35 < 30$ which is in the third layer. And this is dramatically false in the case of 0, which is less than all other natural numbers, but is at the *top* of our divisibility diagram.

37.2 GCD and LCM

Definition 37.1. Let $a, b \in \mathbb{N}$. Then the *greatest common divisor* of a and b, denoted $\gcd(a, b)$, is the unique natural number such that for all $d \in \mathbb{N}$,

$$(d \mid \gcd(a, b)) \leftrightarrow (d \mid a \wedge d \mid b).$$

In the left-to-right direction, this says that anything which divides $\gcd(a, b)$ must be a common divisor of a and b. Because divisibility is transitive, this will clearly be the case when $\gcd(a, b)$ is a divisor of both a and b. And in fact, picking $d = \gcd(a, b)$ shows that this must be the case. In the right-to-left direction, this says that any common divisor of a and b must divide their GCD. This is what makes it the "greatest" common divisor: there can't be any common divisor d which is "above" the GCD.

Remark. Note that this definition doesn't say anything about $\leq$! The word *greatest* in the phrase *greatest common divisor* should really be taken to mean *highest in the divisibility diagram*, not "greater than" in the usual $\geq$ sense.

Example.

$$\begin{aligned} \gcd(6, 10) &= 2 \\ \gcd(2, 3) &= 1 \\ \gcd(2, 4) &= 2 \\ \gcd(3, 3) &= 3 \\ \gcd(0, 7) &= 7 \\ \gcd(0, 0) &= 0 \end{aligned}$$

If we took the word "greatest" in GCD to refer to the $\leq$ relation, then we would have to say $\gcd(0, 0)$ is undefined; every natural number is a common divisor of

0 and 0, and there is no greatest natural number. However, if "greatest" means "highest in the divisibility diagram" then clearly the answer should be 0. And indeed, we can easily check that 0 satisfies the definition: for every $d \in \mathbb{N}$,

$$(d \mid 0) \leftrightarrow (d \mid 0 \wedge d \mid 0)$$

is trivially true.

Definition 37.2. The *least common multiple* of $a, b \in \mathbb{N}$ is the unique natural number such that for all $m \in \mathbb{N}$,

$$(\mathrm{lcm}(a, b) \mid m) \leftrightarrow (a \mid m \wedge b \mid m).$$

We won't talk much more about lcm—it has very similar properties to gcd but with everything flipped.

37.3 Computing GCD and LCM via factoring

One way to compute the GCD of a and b (and the way you're probably familiar with from grade school) is to factor both, and pick the biggest power of each prime that divides both. For example,

$$\gcd(18, 60) = \gcd(2^1 \cdot 3^2 \cdot 5^0, 2^2 \cdot 3^1 \cdot 5^1) = 2^1 \cdot 3^1 \cdot 5^0 = 6.$$

In general, consider writing a natural number n as an infinite product of all the primes raised to powers (if n does not have a certain prime p as a divisor, we just set p's exponent to 0):

$$a = 2^{a_2} 3^{a_3} 5^{a_5} 7^{a_7} \dots$$

Then in general,

$$\gcd(a, b) = \gcd(2^{a_2} 3^{a_3} 5^{a_5} \dots, 2^{b_2} 3^{b_3} 5^{b_5} \dots) = 2^{\min(a_2, b_2)} 3^{\min(a_3, b_3)} 5^{\min(a_5, b_5)} \dots$$

That is, we take the minimum power of each prime. This works even when some of the powers are 0. The proof is not too hard and left as an exercise.

Likewise, the LCM can be found by taking the max of each prime power:

$$\mathrm{lcm}(a, b) = \mathrm{lcm}(2^{a_2} 3^{a_3} 5^{a_5} \dots, 2^{b_2} 3^{b_3} 5^{b_5} \dots) = 2^{\max(a_2, b_2)} 3^{\max(a_3, b_3)} 5^{\max(a_5, b_5)} \dots$$

Incidentally, this shows us that

Theorem 37.3. *For every $a, b \in \mathbb{N}$,*

$$\gcd(a, b) \cdot \mathrm{lcm}(a, b) = ab.$$

Proof. Follows from the fact that $\min(x, y) + \max(x, y) = x + y$. □

37.4 Computing GCD via the Euclidean algorithm

This method of computing the GCD is not very practical for larger numbers, because factoring takes a long time. It turns out there is a much better way that is actually used in practice. To see why it works, we'll first need a few lemmas and theorems about gcd.

Lemma 37.4. $\gcd(a, b) = \gcd(b, a)$.

Proof. Look at the definition of gcd and note that $\wedge$ is commutative. □

Theorem 37.5. *Let $a, b \in \mathbb{N}$. Then for all $k \in \mathbb{Z}$,*

$$\gcd(a, b) = \gcd(a + kb, b).$$

Proof. Omitted in class. See Bonus Proofs section below. □

Theorem 37.6. *Let $a, b \in \mathbb{N}$. Then* $\gcd(a, b) = \gcd(b, a \bmod b)$.

Proof. By the division algorithm write $a = bq + r$. Then

$$\begin{aligned}
&\gcd(a, b) \\
=\ &\{ \text{ Division algorithm, substitute for } a \} \\
&\gcd(bq + r, b) \\
=\ &\{ \text{ Theorem 37.5 } \} \\
&\gcd(bq + r - bq, b) \\
=\ &\{ \text{ algebra } \} \\
&\gcd(r, b) \\
=\ &\{ \text{ gcd is commutative } \} \\
&\gcd(b, r) \\
=\ &\{ \text{ Definition of } \mathbf{mod} \} \\
&\gcd(b, a \bmod b)
\end{aligned}$$

□

Notice that by the division algorithm, $r = a \bmod b$ is guaranteed to be strictly smaller than b. So if we replace $\gcd(a, b)$ by $\gcd(b, a \bmod b)$ then the second input to gcd is getting smaller. This leads to:

Definition 37.7 (Euclidean Algorithm)**.** Let $a, b \in \mathbb{N}$. To find $\gcd(a, b)$, repeatedly apply these rules:

$$\begin{aligned}
\gcd(a, 0) &= a \\
\gcd(a, b) &= \gcd(b, a \bmod b)
\end{aligned}$$

That is, just keep repeating the second case until reaching the first case. This is guaranteed to stop in a finite amount of time since the second input to gcd gets smaller with each step; it must hit zero eventually.

Example.

$$\begin{aligned} &\gcd(60, 18) \\ &= \gcd(18, 6) \\ &= \gcd(6, 0) \\ &= 6 \end{aligned}$$

Example.

$$\begin{aligned} &\gcd(7169, 7811) \\ &= \gcd(7811, 7169) \\ &= \gcd(7169, 642) \\ &= \gcd(642, 107) \\ &= \gcd(107, 0) \\ &= 107 \end{aligned}$$

We can confirm that 107 is indeed a common divisor of $7169 = 107 \times 67$ and $7811 = 107 \times 73$. Since 67 and 73 are both prime there is obviously no bigger common divisor.

37.5 Bonus proofs

Lemma 37.8. *Let $a, b \in \mathbb{N}$. If $a \mid b$ and $b \mid a$, then $a = b$.*

Remark. This property is called *antisymmetry*. Notice $\leq$ has this property too: if $a \leq b$ and $b \leq a$, then $a = b$.

Proof. If $a \mid b$ then there is some $k \in \mathbb{Z}$ such that $ka = b$. Likewise, if $b \mid a$ then $jb = a$. Substituting, we find that $a = jb = jka$, and hence $jk = 1$. Since $j, k \in \mathbb{Z}$ the only possibilities are $j = k = 1$ or $j = k = -1$; since a and b are both natural numbers they can't be negative, so in fact $j = k = 1$ and $a = b$. □

Theorem 37.9. *Let $a, b \in \mathbb{N}$. Then for all $k \in \mathbb{Z}$,*

$$\gcd(a, b) = \gcd(a + kb, b).$$

Proof. Using the lemma above, we will show $\gcd(a, b) = \gcd(a + kb, b)$ by showing each divides the other.

- $\gcd(a, b) \mid a$ and $\gcd(a, b) \mid b$ by definition. Using a theorem we proved previously (Theorem 33.2), we can therefore conclude that $\gcd(a, b) \mid (a + kb)$. Hence, since $\gcd(a, b) \mid (a + kb)$ and $\gcd(a, b) \mid b$, by definition of $\gcd(a + kb, b)$ we conclude that $\gcd(a, b) \mid \gcd(a + kb, b)$.
- By definition, $\gcd(a + kb, b) \mid (a + kb)$ and $\gcd(a + kb, b) \mid b$. Hence by the same theorem as in the other case, it also divides a combination of them, $\gcd(a + kb, b) \mid (a + kb + (-k)b) = a$. Since it divides both a and b, by definition of $\gcd(a, b)$ we have $\gcd(a + kb, b) \mid \gcd(a, b)$.

Since each divides the other, we conclude that $\gcd(a, b) = \gcd(a + kb, b)$. □

38 Bézout's Theorem (Friday 18 April)

A mathematical frog starts at zero on the number line. He can jump 30 units in either direction, or hop 18 units in either direction. Which positions on the number line can he reach? Can he reach 12? 6? 3? 1? 8? 42252? 42254?

Discuss: the frog can reach 12 by jumping right then hopping left. He can reach 6 by hopping right twice then jumping left. He can't reach 3 or 1, since jumps and hops are both even and 3 is odd. 8 is even, but he can't reach that either since in fact both 12 and 30 are multiples of 6, but 8 is not. We can check that $4+2+2+5+2=15$ is divisible by 3, hence 42252 is divisible by 6, so we conjecture he can reach that, but not 42254.

In fact, it turns out that the frog can reach position n if and only if $6 \mid n$. What's special about 6? It's the GCD of 18 and 30.

Theorem 38.1 (Bézout's Theorem)**.** *Let $a, b \in \mathbb{N}$. Then there exist $s, t \in \mathbb{Z}$ such that*

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

This is like saying we can reach $\gcd(a, b)$ by some combination of jumps and hops.

Proof (sketch). Let $S = \{ja + kb \mid j, k \in \mathbb{Z}\}$, the set of all *linear combinations* of a and b. It's a countably infinite set, and definitely contains lots of both positive and negative numbers as well as zero. But it must have some *smallest positive* element—call it d.

- Show $d \mid a$: write $a = dq + r$; show $r \in S$; conclude $r = 0$ since $r < d$ but d is the smallest positive element of S.
- Likewise show $d \mid b$.
- Show if $c \mid a$ and $c \mid b$ then $c \mid d$.
- Conclude $d = \gcd(a, b)$ by definition of gcd.

□

S'20: Above is what we actually did in class. Below is a fuller version of the same proof.

Proof. Let $S = \{ja + kb \mid j, k \in \mathbb{Z}\}$ be the set of all *linear combinations* of a and b. Note that S must have some *smallest positive* element, call it d.[5]

We first show that $d \mid a$. By the Division Algorithm we can write $a = dq + r$ for some q, r such that $0 \leq r < d$. Since $d \in S$ it must be equal to $sa + tb$ for some $s, t \in \mathbb{Z}$. Then

$$r = a - dq = a - (sa + tb)q = a - sqa - tqb = (1 - sq)a + (-tq)b,$$

[5]This is not automatically true for any set—for example, the set of real numbers $\mathbb{R}$ has no smallest positive element. But it is true for sets of integers in particular; this is known as the *well-ordering principle*, which it turns out is equivalent to induction.

which is also a linear combination of a and b. Hence $r \in S$ as well. But $r < d$ and d is the smallest positive element of S; hence r cannot be positive and we must have $r = 0$. Thus $a = dq + r = dq + 0$, so $d \mid a$. A similar argument shows $d \mid b$ as well.

Finally, for any c, if $c \mid a$ and $c \mid b$, then c also divides d since d is a linear combination of a and b.

We have shown that d divides both a and b, and that conversely any common divisor of a and b must also divide d; hence, by definition, $d = \gcd(a, b)$. □

38.1 The Extended Euclidean Algorithm

Given some a and b, how do we actually *find* s and t? The above proof is not much help: it merely shows s and t must exist, but does not actually show us how to find them. Fortunately, there is an efficient procedure for doing so, known as the *Extended Euclidean Algorithm*.

Let's take $a = 60$, $b = 18$ as an example. We begin by making a table with three columns labelled s, t, and $60s + 18t$. In the first row we set $s = 1$ and $t = 0$; in the second row we do the reverse. The value in the third column should always be the corresponding value of the expression $60s + 18t$.

s	t	$60s + 18t$
1	0	60
0	1	18

Of course, $60 \cdot 1 + 18 \cdot 0 = 60$, and $60 \cdot 0 + 18 \cdot 1 = 18$, so we fill in these values in the third column.

The idea is that we are going to run the Euclidean Algorithm on the values in the third column, but along the way, we are going to keep track of the values of s and t that generate each. If we were running the Euclidean Algorithm to compute $\gcd(60, 18)$, our first step would be to replace $(60, 18)$ by $(18, 6)$ (since $60 \textbf{ mod } 18 = 6$). So we place a 6 in the third column under the 18:

s	t	$60s + 18t$
1	0	60
0	1	18
		6

But what should we put in the first two columns? The key is to think about how we get 6 from 60 and 18. We get the remainder 6 by subtracting some number of copies of 18 from 60: in fact, we subtract $60 \textbf{ div } 18 = 3$ copies of 18. The big insight is that we can do exactly the same operation in each of the first two columns, namely, subtract three times the bottom value from the top value. In general, if $60s_1 + 18t_1 = x$ and $60s_2 + 18t_2 = y$, then

$$x - qy = (60s_1 + 18t_1) - q(60s_2 + 18t_2) = 60(s_1 - qs_2) + 18(t_1 - qt_2).$$

So we put $1 - 3 \cdot 0 = 1$ in the first column and $0 - 3 \cdot 1 = -3$ in the second column:

s	t	$60s + 18t$
1	0	60
0	1	18
1	-3	6

And indeed, we can verify that $60 \cdot 1 + 18 \cdot (-3) = 6$.

If we proceed one more step, we find that $18 \textbf{ mod } 6 = 0$, which means that 6 is actually the gcd of 60 and 18, and we have found the coefficients s and t we were looking for.

s	t	$60s + 18t$
1	0	60
0	1	18
1	-3	6
		0

Let's do one more example. This time, let's make a fourth column where we record the quotient q. That is, at each step we look at the two last values in the third column and divide the upper value by the lower; we record the result q in the fourth column next to the lower value; then we subtract q times the bottom row from the one above it to generate the next row. This time let's take $a = 39$ and $b = 16$.

s	t	$39s + 16t$	q
1	0	39	
0	1	16	2
1	-2	7	2
-2	5	2	3
7	-17	1	2
		0	

So we have found that $\gcd(39, 16) = 1$, and that $39 \cdot 7 + 16 \cdot (-17) = 1$ (check this!). In other words, if there is a frog that can jump 39 units or hop 16 units, it could reach 1 starting from the origin by jumping 7 times to the right and 17 times to the left.

39 Modular inverses and Fermat's Little Theorem (Monday 21 April)

0	16	12	8	4
5	1	17	13	9
10	6	2	18	14
15	11	7	3	19

What patterns do you notice? How was the above grid constructed? Can you do something similar with other grid sizes?

5	3	1	6	4	2	7

7	1	8	2	9	3	10	4	11	5	12	6	13

Can you figure out how the above two strips were constructed? Can you make some other examples?

39.1 Modular Inverses

Bézout's Theorem is rarely useful in and of itself; but it's an important building block in several other things we care about more directly. The biggest is the following fact:

Theorem 39.1 (Modular inverses)**.** *Let $a \in \mathbb{Z}$ and $m \in \mathbb{Z}^+$. If $\gcd(a, m) = 1$, then there exists some $b \in \mathbb{Z}$ such that $ab \equiv_m 1$. We say that b is the* modular inverse *of a modulo m.*

Proof. By Bézout's Theorem there exist integers s and t such that $sa + tm = 1$. Since $tm \equiv_m 0$, in fact

$$1 \equiv_m sa + tm \equiv_m sa.$$

So s itself is the modular inverse of a. □

(This theorem can actually be stated as an "if and only if". Can you prove the other direction? *i.e.* if $ab \equiv_m 1$, then $\gcd(a, m) = 1$.)

Of course, this proof shows that not only do modular inverses exist, but we can compute them easily using the extended Euclidean algorithm!

Example. Solve for x:

$$3x \equiv_7 5.$$

Since $\gcd(3, 7) = 1$, we can multiply both sides by the modular inverse of 3, which will "cancel" the 3 from the left. Using the extended Euclidean algorithm

(or just trying a few values), we find that the inverse of 3 is $(-2) \equiv_7 5$. (We can verify that $3 \cdot 5 = 15 \equiv_7 1$.) Multiplying both sides of the equation by 5, we get

$$x \equiv_7 25 \equiv_7 4.$$

So the solution is $x \equiv_7 4$, that is, any value of x which is four more than a multiple of 7 will be a solution. We can check this: $3 \cdot 4 = 12 \equiv_7 5$, as desired.

Example. Solve for x:

$$3x \equiv_9 5 - x.$$

We can't just start by finding a modular inverse of 3: for one thing, we would still have an x on the other side of the equation; for another thing, 3 and 9 share a common factor so 3 actually doesn't have a modular inverse modulo 9. We should add x to both sides of the equation first, which yields

$$4x \equiv_9 5.$$

We find that the modular inverse of 4 modulo 9 is $-2 \equiv_9 7$. Multiplying both sides of the equation by -2 (we could also multiply both sides by 7) we get

$$x \equiv_9 -10 \equiv_9 -1 \equiv_9 8.$$

Checking, if we take $x = -1$ we get $3(-1) = -3 \equiv_9 5 - (-1) = 6$.

39.2 Chinese Remainder Theorem

S'25: Left out CRT this time around in the interest of time. It's technically needed for the proof of RSA but can pretty safely be skipped.

Theorem 39.2 (Chinese Remainder Theorem)**.** *Let* $\gcd(m, n) = 1$ *and* $a, b \in \mathbb{Z}$. *Then the system of equivalences*

$$x \equiv_m a$$
$$x \equiv_n b$$

has a unique solution modulo mn.

Proof. See `https://mathlesstraveled.com/2019/04/10/chinese-remainder-theorem-proof/`. □

One way to understand this theorem is that it gives us a way to replace any two equations $x \equiv_m a$ and $x \equiv_n b$ with a single equation $x \equiv_{mn} c$ (as long as m and n share no common factors). If we have more than two equations, we can simply iterate this process.

Here is a simple way to solve a system of modular equivalences by hand.

Example. Solve the system of equivalences

$$x \equiv_5 1 \tag{1}$$
$$x \equiv_6 2 \tag{2}$$
$$x \equiv_7 3 \tag{3}$$

First of all, we can see that 5, 6, and 7 do not share any common factors, so we are guaranteed a unique solution modulo $5 \cdot 6 \cdot 7 = 210$. Let's see how to find it. The overall plan is as follows:

- Given $x \equiv_m a$, write $x = mk + a$ for some k.
- Substitute this expression for x in the next equivalence.
- Solve for k.
- Substitute the resulting expression for k back into the expression for x.
- Repeat until done.

Let's see this in action. First, from $x \equiv_5 1$ we can write

$$x = 5t + 1 \tag{4}$$

for some $t \in \mathbb{Z}$. Substituting this expression for x into equation (2) yields $5t + 1 \equiv_6 2$, which we can solve for t to get

$$t \equiv_6 5$$

(note that $5 \equiv_6 -1$, so it is its own inverse). We can thus write $t = 6u + 5$ for some $u \in \mathbb{Z}$. Substituting this expression for t back into equation (4), we get

$$x = 5(6u + 5) + 1 = 30u + 26. \tag{5}$$

Now we repeat the process with the last equation. Substituting for x, we get $30u + 26 \equiv_7 3$, which is equivalent to $2u \equiv_7 5$. Solving for u, we find that $u \equiv_7 6$, so we may write $u = 7v + 6$ for some $v \in \mathbb{Z}$. Substituting back into equation (5) yields our final answer,

$$x = 30(7v + 6) + 26 = 210v + 180 + 26 = 210v + 206,$$

meaning the solutions are all of the form $x \equiv_{210} 206$. Indeed, we can verify that $206 \bmod 5 = 1$, $206 \bmod 6 = 2$, and $206 \bmod 7 = 3$.

39.3 Fermat's Little Theorem

If you play around a bit with arithmetic modulo p, one natural question you might end up asking is: if I start with some number a and keep multiplying a by itself (reducing modulo p every time), will I ever reach 1? If so, how many times do I have to multiply by a before reaching 1? If you did this when p is a prime, you would quickly notice that you always seem to reach 1 eventually, and that the number of iterations needed is always a divisor of $p-1$. This leads us to:

Theorem 39.3. *If p is prime and a is any integer such that $p \nmid a$, then*

$$a^{p-1} \equiv_p 1.$$

This seems kind of weird and random, but in fact this theorem expresses something really fundamental about the way multiplication works modulo p. As we will see, this theorem forms an important basis for a lot of modern cryptography. It's also the basis for how tests for primality work. Consider the contrapositive: if $a^{p-1} \not\equiv_p 1$, then p is not prime. So given a number p we can pick some a that shares no common factors with p, and try computing $a^{p-1} \bmod p$. If we get 1, we can't conclude anything; but if we get any result other than 1, we know for sure that p is not prime (but it doesn't tell us what the factors of p are!). This is not a great test by itself, it turns out, but it is the basis for more sophisticated tests.

Proof. Consider the set

$$S = \{a \bmod p, 2a \bmod p, 3a \bmod p, \ldots, (p-1)a \bmod p\}.$$

For example, if $p = 7$ and $a = 3$, we get the set

$$\{3, 6, 2, 5, 1, 4\}.$$

Notice how this contains every integer between 1 and 6. This corresponds to one of the pictures from the start of class—when we count by a's and wrap modulo p, we hit every cell exactly once.

We can make this precise: we claim that if $p \nmid a$ then S contains every number in $\{1, \ldots, (p-1)\}$ exactly once. For suppose there are two multiples of a in S which are equivalent modulo p, that is,

$$ja \equiv_p ka.$$

Then by definition, $p \mid (ja - ka)$ and so $p \mid (j-k)a$. Since p does not divide a, it must divide $(j-k)$. But j and k are both greater than 0 and less than p, so their difference cannot be as small as $-p$ or as big as p. The only way $j-k$ could be divisible by p is if $j-k=0$, that is, $j=k$. This shows that the two "different" multiples of a with the same remainder modulo p actually have to be the same—or, put another way, any two truly different multiples of a must also have different remainders modulo p.

Now consider taking the product of all the elements of S:

$$a \cdot 2a \cdot 3a \cdot \cdots \cdot (p-1)a = a^{p-1}(p-1)!$$

If we consider this modulo p, we know that every possible remainder from 1 up to $(p-1)$ occurs exactly once, so

$$a \cdot 2a \cdot 3a \cdot \cdots \cdot (p-1)a \equiv_p 1 \cdot 2 \cdot 3 \cdot \cdots \cdot (p-1).$$

Putting these together,

$$a^{p-1}(p-1)! \equiv_p (p-1)!.$$

Since $(p-1)!$ shares no common factors with p (p is prime, and $(p-1)!$ is the product of a bunch of numbers less than p, so p cannot possibly be a divisor of $(p-1)!$), it has a multiplicative inverse modulo p, and we are justified in canceling it from both sides of the equation. This leaves us with

$$a^{p-1} \equiv_p 1,$$

which is what we wanted to show. □

40 Public-key cryptography and RSA (Wednesday 23 April)

```
J GPS POF XFMDPNF PVS OFX VOJDPSO PWFSMPSET
```

This is a *Caesar cipher*—shift every letter by the same amount through the alphabet. It's not a very secure cipher, but there are (slightly) more secure variants, such as a *Vigenère cipher*, where instead of a single shift amount we choose a sequence of shift amounts and cycle through them. Typically one would choose a *key word* and use the letters of the key word to determine the shift amounts.

These are all known as *symmetric ciphers*: in order to communicate, sender and receiver must have a *shared, secret key*. The secret key must be known by the people communicating (and no one else), and the same key is used for both encrypting and decrypting. This works fine in many situations, but it has some drawbacks:

- The two parties must somehow agree on a secret key before they can communicate. In order to keep the key a secret, typically this must be done *in person*.
- Because of this need to agree on a secret key, it is impossible for people who have never met to communicate securely.

40.1 Public-key cryptography

There is a different idea known as a *public-key* cryptosystem which does not have these drawbacks. We'll start by outlining the general properties of such a system.

- Everyone who wants to communicate picks their own personal pair of keys: a *private key* and a *public key*.
- The private key should be kept absolutely, 100% secret.
- The public key should be published as widely as possible. You should put your public key on your website, in a public directory of public keys (like a phone book), and so on.
- In order to send a message, the sender looks up the public key of the recipient and uses it to encode the message.
- The recipient decodes the message using their *private* key.

In order for this to make sense the system should have a few properties:

- A public and private key work together so that messages encrypted using a public key can *only* be decrypted using the corresponding private key.

- It should be impossible to figure out someone's corresponding private key if you only know their public key.

It is not at all obvious that such a scheme is even possible! But it *is* possible using some clever mathematics.

40.2 Some facts

Here are a few facts that you'll have to take on faith for now.

- **Multiplying two large integers can be done very quickly.** By *large* we have in mind something like 500 digits. This may sound like a lot, but a modern computer can multiply two 500-digit numbers in a miniscule fraction of a second. If you were abducted by a sadistic math nerd, locked in a room, and told you will not be let out until you correctly multiply two 500-digit numbers, it would be tedious but you could do it. You could probably do it in an afternoon.

- On the other hand, **the opposite operation—factoring a large integer into its prime factors—is believed to be very difficult**. Given a 1000-digit number n, there are approximately 8×10^{496} primes less than $\sqrt{n}$. Even if you could test a trillion trillion trillion trillion trillion primes per second to see whether they are a divisor of n, it would still take many lifetimes of the universe to try them all. We do know of better methods than that—but not anywhere near good enough to make it in any way feasible.

- You might think that this means testing whether a number is prime is difficult too—surely, the only way to test whether a number is prime is to try factoring it? Surprisingly, this is false—**it is possible to quickly test whether a number is prime *without* factoring it**! (In fact, most such tests rest fundamentally upon Fermat's Little Theorem, which tells us that if p is prime and p does not divide a, then $a^{p-1} \equiv_p 1$—so we can take a number n we want to test and some $a < n$ and compute $a^{n-1} \bmod n$. If we get something other than 1, then n is definitely not prime. This isn't a great test, but more sophisticated tests are based on similar ideas.)

These ingredients come together to give us RSA.

40.3 RSA

RSA was discovered in 1977 by Ron Rivest, Adi Shamir, and Leonard Adleman. (It was actually independently discovered four years earlier, by Clifford Cocks, while he was working for the UK's GCHQ—basically the equivalent of the NSA—but this wasn't known until 1997 when it became declassified.)

Here's how generating a private/public key pair works:

- Pick two large (say, 500-digit) primes p and q. This can be done efficiently: just pick a random 500-digit number and test it to see if it is prime (which can be done efficiently), if not, pick another one and repeat. For 500-digit numbers about one in every 2300 integers is prime, so you might have to repeat this procedure a few thousand times before finding a prime, but that's not a big deal. Even if this process were to take a few seconds or even minutes, you only have to do it once when creating a public and private key.

- Compute $n = pq$. As mentioned previously, this can be done very quickly.

- Choose some e such that $1 < e < (p-1)(q-1)$ and $\gcd(e, (p-1)(q-1)) = 1$. For example, we could randomly pick an e and compute the gcd (which is very efficient using the Euclidean Algorithm) to make sure it is 1. If not, pick a different e and repeat. In practice, it can be beneficial to use specific values of e to make encoding more efficient, so one could instead fix the value of e and keep picking p and q until $\gcd(e, (p-1)(q-1)) = 1$.

- Find d such that $de = 1 \pmod{(p-1)(q-1)}$, that is, the modular inverse of e modulo $(p-1)(q-1)$. Since e shares no common factors with $(p-1)(q-1)$, a modular inverse is guaranteed to exist, and we can find it easily with the extended Euclidean algorithm.

- p, q, and d are the private key and should be kept absolutely secret.

- n and e are the public key and should be published widely.

As far as we know, of course, it is very difficult to find p and q by factoring n. In turn this means it is difficult to find d from e. We can easily compute d from e if we know $(p-1)(q-1)$, but there's no way to find $(p-1)(q-1)$ from n without first factoring n to find out what p and q are. This means that no one can figure out d, p, or q (the private key) if they only know n and e (the public key).

Now, to **encrypt** a message:

- Find or look up the public key values n and e for the desired recipient.

- Turn the message into an integer $M < n$. It does not matter how, as long as the sender and receiver agree on how this will be done. The method can even be public—knowing how the message is encoded as an integer will not help anyone read it once it is encrypted. In practice, most information that people want to encrypt is already stored on a computer as a sequence of bits, which can readily be interpreted as an integer. If the message is too long to fit into an integer which is less than n, it can simply be broken into multiple parts, and each part encoded separately.

- Compute $C = M^e \bmod n$. This can be done efficiently due to two key observations:

- It is not necessary to first compute M^e (which might be truly enormous) and then reduce modulo n at the end. Because $(a \cdot b) \bmod n = ((a \bmod n) \cdot (b \bmod n)) \bmod n$, we can keep reducing modulo n along the way, after every multiplication. That way, we never have to deal with numbers bigger than n^2.
- We don't have to simply multiply M by itself e times. Instead we can use something called *repeated squaring*. For example, if we want to compute M^{16}, we can first compute $M \cdot M = M^2$; then we can square that to get $(M^2) \cdot (M^2) = M^4$; squaring that yields $(M^4) \cdot (M^4) = M^8$; and one final squaring yields M^{16}. If we want an exponent which is not a power of two we can simply throw in some extra multiplications by M (for example, if we want M^9 we can get M^8 as described above and then multiply by M). In this way we can compute M raised to very large powers with only a few multiplications (it takes about $2 \log_2 e$ multiplications in the worst case for exponent e).

- Send C.

It turns out that given only C, e, and n, it is (as far as we know) very difficult to find an M such that $M^e \equiv_n C$. So no one can read the encrypted message C even if they know the public key (e, n) that was used to encrypt it.

However, to **decrypt** the encrypted message C, the recipient simply does the same thing but using their private key d as the exponent instead of e. That is, they compute $C^d \bmod n$.

Theorem 40.1. *Encrypting and then decrypting in this way results in recovering the original message, that is,*

$$(M^e)^d \equiv_n M.$$

Proof. Since $de \equiv 1 \pmod{(p-1)(q-1)}$, there exists some integer k such that

$$de = 1 + k(p-1)(q-1).$$

Also, by Fermat's Little Theorem, if $p \nmid M$, we have

$$M^{p-1} \equiv_p 1,$$

so

$$(M^e)^d = M^{1+k(p-1)(q-1)} = M \cdot (M^{p-1})^{k(q-1)} \equiv_p M \cdot 1^{k(q-1)} = M.$$

Likewise, by a similar argument $(M^e)^d \equiv_q M$. Now, M is obviously a solution to the system of equations $x \equiv_p M$, $x \equiv_q M$; but since p and q are primes and hence share no common factors, by the Chinese Remainder Theorem we know that the solution to these equations is unique modulo $pq = n$. Hence $(M^e)^d \equiv_n M$. □

40.4 Digital signatures

There is still a problem with this scheme. Under a symmetric cipher, where sender and receiver share a secret key, when the receiver gets a message encrypted using the secret key they can be sure it came from the sender (unless someone has stolen the secret key). However, in a public key system, anyone in the world can encrypt a message to the receiver, so they have no way of knowing who the message is actually from. There is no a priori reason to trust who the message *says* it is from, because anyone can write a message that says "Hi, this is Alice" whether or not they are actually Alice.

However, this problem can be solved very elegantly by noticing that the operations of encryption and decryption are symmetric, that is, $(M^e)^d = (M^d)^e$. That is, a private key can actually be used to *encrypt* a message, and then the corresponding public key will work to decrypt it. So, suppose A wants to send a secure message to B, and they also want B to be able to verify that the message actually came from them. Instead of directly encrypting the message for B, first A encrypts the message using A's *private* key. This is known as a *digital signature*. Then, A takes the resulting signed message and encrypts it using B's public key as usual. When B receives the encrypted message, they first decrypt it using their private key as usual. The result will not be readable since it has been encrypted using A's private key. But B can now recover the original message by decrypting using A's *public* key, which will undo the effects of encrypting with A's private key. B can now be sure the message came from A, because the only way to make a message that can be successfully decrypted using A's public key would be to use A's private key; and the only person who can do that (assuming A's private key is secret) is A.

41 Principles of combinatorics: the product rule (Friday 25 April)

Combinatorics is the area of mathematics about *counting*, that is, determining the sizes of sets. This sounds like it would be trivial or uninteresting, but it turns out to be deep and fascinating, with many connections to other areas of mathematics.

You may have memorized a few combinatorics formulas in high school (permutations, combinations, ...) but we are going to learn some basic underlying principles—if you understand the principles involved then you can easily rederive the formulas as needed.

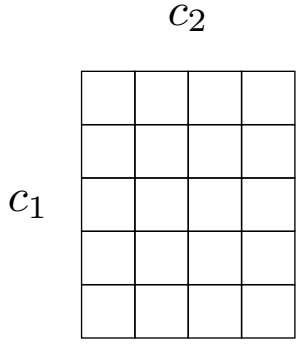
41.1 The product rule

The first principle we will consider is the *product rule*. Note, instead of just talking about the sizes of sets we will often talk in terms of *choosing* elements of a set. The number of elements in a set is the same as the number of ways to choose a different element of the set, so in some sense this makes no difference; but for some reason it is often helpful to think in terms of actively *choosing* elements rather than just passively *counting* them.

> Suppose choosing an item can be decomposed into two *independent* choices, with c_1 ways to make the first choice, and c_2 ways to make the second choice. Then the total number of ways to choose overall is the product $c_1 \times c_2$.

Here, *independent* means that you have the same set of options for both choices no matter what you pick. In other words, choosing a certain option for one of the choices cannot affect which options are available for the other choice.

We can visualize the situation by drawing a rectangular grid, where each row corresponds to one option for the first choice, and each column corresponds to an option for the second choice. Then each grid square corresponds to a unique combination of choices. The fact that the choices are *independent* corresponds to the fact that the picture is perfectly rectangular.



This really boils down to the fact, which we saw before when learning about sets, that

$$|A \times B| = |A| \times |B|.$$

Example. There are three kinds of cereal and seven kinds of fruit. For breakfast, you must choose one cereal and one fruit. How many breakfast choices do you have?

The product rule clearly applies here: the answer is $3 \times 7 = 21$ choices.

Example. This is a *non*-example. Suppose we have the same scenario as the previous problem, *except* that when you choose oatmeal, you are limited to having only strawberries or blueberries.

In this case the product rule does *not* apply (at least not directly) since the choices are no longer independent.

Example. How many 3-letter strings are there using the letters A–Z? Examples of such strings include AAA, HDX, CAR, and so on.

The number of two-letter strings is 26×26 since there are two independent choices: which letter to choose as the first letter, and which to choose as the second. Then the number of 3-letter strings is the number of ways to pick a two-letter string, times the number of ways to pick a third letter, for a total of $(26 \times 26) \times 26 = 26^3 = 17576$.

Similarly, there are 26^4 four-letter strings, 26^5 five-letter strings, and in general there are 26^k k-letter strings. This reasoning shows that we can generalize the product rule to a product of n choices instead of just two. If we have n independent choices to make, we just multiply all of them to compute the total number of choices.

Example. How many subsets are there of an n-element set?

We can use the product rule to answer this question too, though it is less immediately obvious how it applies. When choosing a subset, for each of the n elements we have two choices: whether to include it in the subset or not. All these choices are independent, because each element is allowed to be in the subset or not, regardless of which other elements are or not members of the subset. Hence, by the product rule, the total number of choices is $2 \times 2 \times \cdots \times 2 = 2^n$.

If we identify each subset with a bit string of length n indicating which elements are in the subset and which are out, we can see a connection to the previous example: there are 2^n bit strings of length n, since we can independently choose each bit to be either 0 or 1.

Example. Suppose we have 10 cans of soup (all different) and we want to eat them all, one per day over the next 10 days. In how many ways could we do this? Note this is equivalent to asking how many ways we could order the cans of soup; imagine we first put them in a particular order and then each day just eat the next can in the line.

We might at first think the answer is 10^{10}, since there are 10 days and 10 choices of soup on each day. However, the choices are not independent: once I choose a particular can of soup I can't choose it again on a subsequent day.

(10^{10} is the answer to the related problem of how many soup-eating schedules we can make, if each day we go to the store to buy some soup, and the store carries 10 different kinds. Assuming the store has enough stock, the choices are now independent, since getting a particular soup on one day does not prevent us from picking the same soup again another day.)

However, we can conceive of this problem in terms of independent choices if we change our point of view a little bit. On the first day, I have 10 choices. On the second day, I can no longer choose the soup I ate on the first day, but I can freely choose any soup I want out of the 9 that are left. On the third day, I can choose any of the 8 that remain, and so on. In total, then, I have

$$10 \cdot 9 \cdot 8 \cdot \cdots \cdot 1 = 10! = 3628800$$

ways to do this.

More generally, there are $n(n-1)(n-2)\cdots 1 = n!$ ways to put n things in a particular order, that is, there are $n!$ *permutations* of n things.

Example. What if I have 10 different cans of soup, as in the previous scenario, but now I only want to eat soup for the next four days?

As before, I have 10 choices on the first day, 9 choices on the second day, and so on, for a total of

$$10 \cdot 9 \cdot 8 \cdot 7 = 5040$$

ways. Note this can also be written as

$$10 \cdot 9 \cdot 8 \cdot 7 = \frac{10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1}{6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1} = \frac{10!}{6!}.$$

In general, if I have n things and want to pick out k of them in a particular order, I have

$$n \cdot (n-1) \cdot \cdots \cdot (n-k+1) = \frac{n!}{(n-k)!}$$

ways to do it.

Example. Given finite sets A and B, how many functions $f : A \to B$ are there?

Consider how we might go about choosing a particular function $f : A \to B$. A function from A to B must specify a particular element of B for each element of A. So for the first element $a_1 \in A$, we can choose any element $b \in B$ to which f should send a_1. Likewise, for $a_2 \in A$, we can independently choose any element of B to which f should send a_2, and so on for every element of A. (These choices are all independent because for any old function, we don't care if it sends multiple a_i to the same b, or misses some b's, or whatever.) So for each element of a, we get to make an independent choice of size $|B|$; hence, by the product rule, the total number of ways we could choose a function is

$$\underbrace{|B| \times |B| \times \cdots \times |B|}_{|A|} = |B|^{|A|}.$$

42 Principles of combinatorics: addition and subtraction rules, PIE (Monday 28 April)

> Suppose choosing something means *either* making one choice in c_1 ways *or* making another choice in c_2 ways, and the choices do not overlap at all. Then the total number of choices is $c_1 + c_2$.
>
> In other words, when A and B are disjoint,
>
> $$|A \cup B| = |A| + |B|.$$

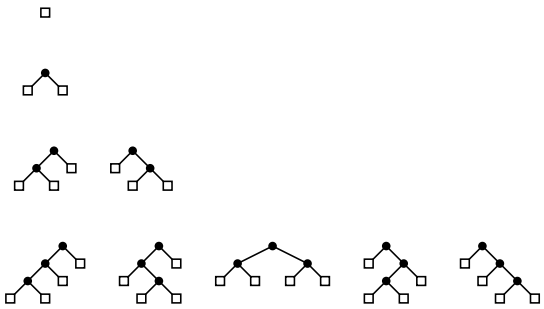
Example. You want to order a single item for lunch. Restaurant A has 13 things on its menu, and Restaurant B has 24 on its menu. How many different choices do you have for lunch, if the two restuarants have completely different menus?

Assuming the restaurant menus do not overlap at all, the answer is simply $13 + 24 = 37$.

(What if the menus do overlap? We will come back to this question.)

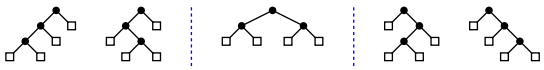
Example. Counting the number of binary trees with a certain number of branch nodes provides a great example putting together many things we have seen so far (the product and sum rules, and recursively defined sets and functions).

Here are all the binary trees with zero, one, two, and three branch nodes:



We can see that there are five different binary trees with three branch nodes. How many will there be with four branch nodes? We are quickly approaching the limit of what we will be able to count while being confident that we have not missed any.

Consider the set of five trees with three branch nodes shown above. Of course, every tree has one branch node at its root; then we have two more branch nodes that we have to put somewhere. We can break the trees into three subsets, depending on how many branch nodes we put to the left and right of the root:



The trees on the left have two branch nodes on the left side of the root and none on the right; the tree in the middle has one branch node on either side; and the

trees on the right have none on the left and two on the right. It's clear that these are the only possibilities, and every tree has to fall into exactly one of these categories. If we put two branch nodes on the left, we have two ways to do it (since there are two different trees with two branch nodes), and only one way to put zero on the right (there is only one tree with zero branch nodes). Similarly, we have one way to make a tree with one branch node to either side, and two ways to put two on the right. In general, we can compute the number of ways we have to make a tree with j nodes on the left and k nodes on the right using the product rule: since the left and right trees can be chosen independently, it is just the number of trees with j branch nodes times the number of trees with k branch nodes.

Put together, this tells us that if $T(n)$ represents the number of binary trees with n branch nodes, then

$$T(3) = T(2)T(0) + T(1)T(1) + T(0)T(2).$$

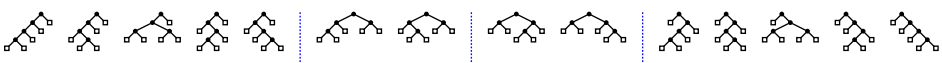
And sure enough, $T(0) = T(1) = 1$ and $T(2) = 2$, so this equals

$$2 \cdot 1 + 1 \cdot 1 + 1 \cdot 2 = 5,$$

as it should.

Using the same pattern we can compute the number of binary trees with four branch nodes:

$$T(4) = T(3)T(0) + T(2)T(1) + T(1)T(2) + T(0)T(3) = 5 + 2 + 2 + 5 = 14.$$



In general, we can define $T(n)$ recursively as

$$T(0) = 1$$
$$T(n) = \sum_{k=0}^{n-1} T(n-k-1)T(k)$$

The resulting sequence of numbers, known as the *Catalan numbers*, is extremely famous:

$$1, 1, 2, 5, 14, 42, 132, 429, 1430, \ldots$$

42.1 Subtraction rule

> If choosing something means making one choice in c_1 ways OR another in c_2 ways, but *the choices overlap*, the total number of choices is $c_1 + c_2 -$ (the number of choices in common).

Another way to write this, which we have seen before, is

$$|A \cup B| = |A| + |B| - |A \cap B|.$$

Example. You again want to order a single item for lunch; restaurant A has 13 things on its menu, and Restaurant B has 24 on its menu. However, there are 9 items which the two restaurants have in common. How many items can you order for your lunch?

Adding $13 + 24$ gives the total number of items on both menus, but it has counted the shared items twice. Subtracting the number of shared items gives the correct total: $13 + 24 - 9 = 28$.

Example. How many strings of 8 bits either start with a 1 or end with 00?

- There are 2^7 strings of 8 bits that start with a 1: the first bit is fixed, and we can independently choose each remaining bit to be 0 or 1.
- Similarly, there are 2^6 strings of 8 bits that end with 00.
- We cannot simply add these numbers, because of overlap: there are some strings of bits that both start with 1 and end with 00. In fact there are 2^5 such strings, since three bits are fixed and we can freely choose the other 5. Thus, the total number of bit strings that start with 1 or end with 00 is

$$2^7 + 2^6 - 2^5 = 160.$$

What if we have *three* (or more) overlapping sets of choices? It is actually possible to come up with a general formula for this situation, known as the Principle of Inclusion-Exclusion.

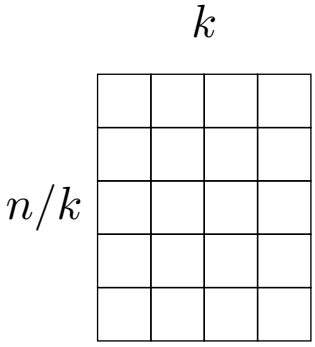
43 The division rule and binomial coefficients (Wednesday 30 April)

Recall the problem with soup, where we have a bunch of different kinds of soup and are trying to count the number of different schedules we could make for ourselves. For example, if we have 10 different kinds of soup and want to eat soup for 4 days, there are $10 \cdot 9 \cdot 8 \cdot 7$ different soup schedules we can make.

Now consider a slightly different question: in how many ways can we choose *which four* soups to eat, if we don't care about the order? In this case $10 \cdot 9 \cdot 8 \cdot 7$ is clearly too big, since there are many different schedules which have the same four soups in different orders.

> Suppose we have n total choices, but we really want to think of some of them as being equivalent. If the choices always come in groups of exactly k choices which are all equivalent, then the number of "really different" choices is n/k.

Put another way: if $|A| = n$ and A is the union of pairwise disjoint subsets, each with size k, then the number of subsets is n/k. We can also think about this in terms of a picture similar to the picture for the product rule.



In this scenario, each square represents a choice, and we know there are n total squares in the entire grid. However, each row represents choices which we want to consider "the same" somehow. If we know there are exactly k squares in each row, we can conclude that there are n/k rows.

Example. How many ways are there to seat 5 people around a table, if we only care about who is sitting to the left and right of who, and not about the specific seats they are in?

One way to answer this question is by first considering that there are 5! ways to assign each person to a specific seat. However, given a specific assignment of each person to a seat, if we have everyone stand up and move one seat to the right, we get a different seat assignment, but as a seating arrangement it should be considered the same, since we only care who is sitting next to who. In fact, for every specific seating arrangement there are four other arrangements which are equivalent, for a total of five equivalent arrangements (corresponding to the

five possible rotations around the table). Thus, by the division rule, there are $5!/5 = 24$ different seating arrangements.

Note we can also see this by first insisting that person A must always sit in a specific chair; since specific chairs don't matter, they might as well. Then we have four choices for who to seat the right of person A, then three choices for who to seat to the right of them, and so on, for a total of 4! choices.

Example. How many ways can we choose 4 out of 10 soups?

We know there are $10 \cdot 9 \cdot 8 \cdot 7$ ways to choose a schedule of four different soups in a specific order. But we want to consider two schedules the same when they use the same four soups, regardless of the order. Given a particular schedule, how many are equivalent to it? This is just the number of ways to put four items in a specific order, which we know is 4!. Hence, by the division rule, the number of ways to choose four out of ten soups is

$$\frac{10 \cdot 9 \cdot 8 \cdot 7}{4 \cdot 3 \cdot 2 \cdot 1} = 210.$$

43.1 Binomial coefficients

More generally, if we want to choose k things out of n—that is, we want to count the number of size-k subsets of a set of size n—we first of all have

$$n \cdot (n-1) \cdot \cdots \cdot (n-k+1) = \frac{n!}{(n-k)!}$$

ways to choose k things in a specific order. But rearranging the k things into a different order is not supposed to make any difference, so for each different subset of k things there are $k!$ orderings which should all be considered equivalent. Thus, by the division rule, there are

$$\frac{n!}{k!(n-k)!} = \binom{n}{k}$$

size-k subsets of a set of size n. The notation $\binom{n}{k}$ is called a *binomial coefficient*, and we pronounce it "n choose k". There is a special notation for them because they show up all over the place in mathematics (not just in combinatorics).

Example. How many 8-bit binary strings are there with exactly three 1's and five 0's?

This is equivalent to picking three out of the eight bit positions to set to 1, so it is $\binom{8}{3} = \frac{8!}{3!5!} = \frac{8 \cdot 7 \cdot 6}{3 \cdot 2 \cdot 1} = 56$.

Example. How many 8-bit binary strings have *at least 3* bits set to 1?

It is easiest to compute this by subtracting the strings we don't want to count from the total:

$$2^8 - \binom{8}{2} - \binom{8}{1} - \binom{8}{0} = 256 - 28 - 8 - 1 = 219.$$

44 Binomial coefficients (Friday 2 May)

Here is a table of the first few binomial coefficients, with n along the left side, and k along the top.

	0	1	2	3	4	5
0	1					
1	1	1				
2	1	2	1			
3	1	3	3	1		
4	1	4	6	4	1	
5	1	5	10	10	5	1

(The blank entries are 0, since they represent clearly impossible things like "the number of ways to pick 4 things out of a set of size 2".) This is called *Pascal's Triangle* and is rather well-known (you may very well have seen it before). Some properties of binomial coefficients:

- $\binom{n}{0} = \binom{n}{n} = 1$. We can prove this either by looking at the formula ($\frac{n!}{0!n!} = 1$) or by thinking about what it means combinatorially: there is exactly one way to choose nothing, or everything, from a set of size n.
- $\binom{n}{1} = n$. Again, we can see this from the formula, or by reasoning that if we want to pick exactly one thing out of a set of size n, we have n ways to do it, one for each element.
- $\binom{n}{k} = \binom{n}{n-k}$. In the formula, $k!$ and $(n-k)!$ just switch places; combinatorially, we have the same number of ways to choose k things as we do to choose which $n-k$ things we are *not* going to choose. In other words, instead of thinking in terms of "choosing k things" we can think of it in terms of "splitting a set of size n into two subsets of sizes k and $n-k$".
- $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$. This is the rule for constructing Pascal's Triangle: each entry is the sum of the entry above it and the one above and to the left. I leave it as a challenge for you to prove this either from the formula or by thinking about it combinatorially.
- $\sum_{k=0}^{n} \binom{n}{k} = 2^n$. These are just two different ways to count all the subsets of a set of size n.

45 Introduction to graphs

Motivation: Königsberg Bridge problem. Is it possible to take a walk and cross all bridges exactly once? 1736—Leonhard Euler solved the problem and invented graph theory along the way. Insight: physical geography of the town doesn't really matter. Only thing that matters is which land areas are connected to others (by bridges). Graph theory is all about modelling connections or relationships between things.

Note, by "graph" we don't mean the kind of thing where you make a picture of some $y = f(x)$. Here's what we do mean:

Definition 45.1. A *graph* $G = (V, E)$ is a set of *vertices* V (aka *nodes*) and a set of *edges* E. Each edge connects two vertices, called its *endpoints*.

Draw example graph of cities connected by plane routes. Note that how we draw it doesn't matter: the only thing that matters is which things are connected by edges.

What are some variants of the idea of a graph? We've given one specific definition above, but actually there are lots of slight variants depending on what we're trying to model.

- Sometimes we want to allow multiple distinct edges between the same pair of vertices (*e.g.* multiple different flights between the same citires); sometimes we don't (*e.g.* whether two people are friends on a social media platform; you can't be friends with someone twice)
- Sometimes we allow "self-loops", *i.e.* edges from a vertex to itself
- Sometimes edges have a particular direction (*directed* graph), and sometimes they don't (*undirected* graph)
- Sometimes edges are labelled with some kind of cost/weight.

Brainstorm some examples!

Definition 45.2. $u, v \in V$ are *adjacent* (aka *neighbors*) if they are the endpoints of an edge.

Definition 45.3. The *degree* of a vertex $v \in V$, written $\deg(v)$, is the number of edges adjacent to v. (Loops count twice.)

Remark. A degree-0 vertex connects to no edges. This is perfectly fine.

46 Vertex degrees and Eulerian paths

Look at some example undirected, unweighted graphs (empty graph; single vertex; disconnected graph; complete graph; Königsberg graph ...). Add up vertex degrees of each. Try to get students to make conjecture.

Lemma 46.1 (Handshake Lemma)**.** *In any undirected graph, the sum of all vertex degrees is twice the number of edges.*

Proof. Each edge contributes 2 to the sum. □

Aside: how many edges are in a *complete graph* with n vertices and all possible edges? Three different ways to answer the same questions:

1. $n(n-1)$ is the sum of all degrees, since each of the n vertices has degree $n-1$. By the Handshake Lemma, the number of edges is thus $n(n-1)/2$.

2. Each edge corresponds to a pair of vertices and vice versa. Hence the number of ways to choose an edge is the number of ways to choose a pair of vertices: $\binom{n}{2}$.

3. The first vertex has $n-1$ edges adjacent to it; the second vertex then has $n-2$ edges that we haven't already counted; the third vertex has $n-3$ edges that we haven't already counted; and so on. Thus the total number of edges is $1+2+3+\cdots+n$.

Incidentally, this constitutes a proof that

$$1+2+3+\cdots+n = \binom{n}{2} = n(n-1)/2.$$

They are all equal since they are all different ways of counting the same thing.

Now back to the Königsberg Bridge problem.

Definition 46.2. An *Eulerian* path is a path which visits every edge exactly once.

Any vertex with an odd degree must be either the start or end of an Eulerian path, since every time we visit a vertex in the middle of a path, it uses up 2 edges, and eventually an odd-degree vertex will be reduced to 1 edge.

Hence any graph with more that 2 odd-degree vertices does not have an Eulerian path. This is what Euler proved. Next time we will consider the converse.

47 Hierholzer's Algorithm

Start with some example graphs, see if it's always possible to create an Eulerian path if there are not too many odd vertices. Last example: use really big complicated graph.

Theorem 47.1 (Carl Hierholzer, 1871)**.** *If a connected graph has at most two odd-degree vertices, then it has an Eulerian path.*

Definition 47.2. An Eulerian *circuit* is an Eulerian path that begins and ends at the same vertex.

This seems to make things harder but it actually makes things easier for us.

Theorem 47.3 (Euler + Hierholzer)**.** *A connected graph has an Eulerian circuit if and only if all vertices have even degrees.*

If we have a graph with two odd-degree vertices, just add an edge between them! Then all vertices will have even degree; use the above theorem to find an Eulerian circuit, and finally delete that one added edge to obtain an Eulerian path in the original graph.

Proof. ($\rightarrow$) Already proved.

($\leftarrow$) Proof by algorithm (Hierholzer's Algorithm). Let G be an undirected, connected graph, and suppose all vertices of G have even degree. By induction on the number of edges of G.

- Base case: G has 0 edges. Then G consists of a single vertex (or no vertices), which does have an Eulerian circuit (just stay where you are and visit all edges).
- Now let $k \in \mathbb{N}$ and suppose that any connected graph with $\leq k$ edges and all even degree vertices has an Eulerian circuit. We must show it also holds for $k+1$ edges.

  So let G be a connected graph with $k+1$ edges and suppose all vertices of G have even degree.

  Pick any vertex of G and start walking (*i.e.* following edges) but never repeat an edge. Claim: you will eventually get stuck back at the same vertex where you started. This is because every vertex has even degree, and we always use up two edges when going through any vertex—so every time we get to a vertex there will always be at least one edge remaining along which we can leave the vertex. The only exception is the starting vertex where we used one edge when leaving it for the first time.

  This will thus generate a circuit C, though it may not include every edge in the graph. If we delete all edges of C from the graph, we will be left with some (possibly disconnected) pieces. Each piece is connected, has $\leq k$ edges, and all vertices still have an even degree (since deleting the edges of a circuit means we delete an even number of edges from each vertex). Thus by the induction hypothesis, each piece has an Eulerian circuit. Now just "splice" the Eulerian circuit for each piece into C at some point where they share a vertex. This results in an Eulerian circuit for the entire graph.

□

Remark. Implementing this efficiently is actually nontrivial but it can be done in $O(n)$ time.

Do example.

48 Bonus material: recursive/inductive definitions

S'23: Didn't cover this material this year. We've been writing recursive functions in Disco all along.

48.1 Recursively defined functions

We can define functions using recursion/induction (we have seen this briefly before, when discussing sequences).

- Base case: define the function on some initial value(s).
- Inductive/recursive step: define the function on other values in terms of its output for previous values.

Example. Let $f : \mathbb{N} \to \mathbb{N}$ be the function defined by

$$f(0) = 3$$
$$f(n+1) = 2f(n) + 3$$

We claim that this is a valid definition that uniquely defines the function on all possible input values $n \in \mathbb{N}$. Why is that? By induction: the first equation shows f is defined for 0; and the second equation shows that whenever it is defined for n, then it is defined for $n+1$ as well. Thus by induction it is defined for all $n \in \mathbb{N}$.

We can list a few values of the function:

$$f(0) = 3$$
$$f(1) = 2f(0) + 3 = 9$$
$$f(2) = 2f(1) + 3 = 21$$

Remark. I am using the words "recursion" and "induction" interchangeably to emphasize the very deep connection between the two concepts. Typically we use the word "recursion" when we are talking about computation and "induction" when we are talking about proof, but in a very deep sense the two are the same thing. In fact, we can think of an "inductive proof" of $\forall n.\ P(n)$ as a recursive function that takes a number n as input and outputs a proof of the proposition $P(n)$; using the induction hypothesis in the proof corresponds to using the function recursively. (In fact, there are computer proof systems that work in exactly this way; if you want to learn more about them, take CSCI 365, Functional Programming!)

Example. Let $F : \mathbb{N} \to \mathbb{N}$ be the function such that $F(n)$ is the nth Fibonacci number. (We can think of this either as a function or a sequence: a sequence is "really" just a function with a domain of $\mathbb{N}$.) Then F is defined by

$$F(0) = 0$$
$$F(1) = 1$$
$$F(n+2) = F(n+1) + F(n)$$

Given a recursively defined function, how do we prove things about it? Using induction, of course! In fact, we've seen examples of this already, for example, when proving some facts about the Fibonacci numbers.

Challenge: prove that $\forall n \in \mathbb{N}.\ f(n) = 3 \cdot (2^{n+1} - 1)$.

48.2 Recursively defined sets

We can also define *sets* using recursion/induction.

- Base case(s): specify one or more elements in the set.
- Inductive step(s): give some rules saying how to construct new elements of the set from existing elements in the set.

Example. Let Σ be a finite set of symbols (the "alphabet"). For example, Σ could be the set of English letters $\{A, B, C, \ldots, Z\}$, or it could be the set of base-ten digits $\{0, \ldots, 9\}$, or the set of bits $\{0, 1\}$.

We then define the set of *strings* over the alphabet Σ, written Σ^*, as follows:

- $\lambda \in \Sigma^*$ (λ represents the *empty* string)
- If $w \in \Sigma^*$ and $x \in \Sigma$, then $wx \in \Sigma^*$.

The rule says that if w is any string in Σ^*, then we can always add one more element from the alphabet Σ to the end of w, resulting in a new (longer) string which is also in Σ^*.

For example, suppose $\Sigma = \{a, b, c\}$. Let Σ_i^* be the set we get after applying the recursive rule i times. Then we have:

$$\begin{aligned}
\Sigma_0^* &= \{\lambda\} \\
\Sigma_1^* &= \{\lambda, \lambda a, \lambda b, \lambda c\} = \{\lambda, a, b, c\} \\
\Sigma_2^* &= \{\lambda, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc\}
\end{aligned}$$

Σ^* is what we get if we can apply the rule as many times as we want, that is,

$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma_n^*.$$

Σ^* contains things like *abaccccabca*, *bbb*, or a million a's followed by a single c.

Example. We can recursively define a set $\mathcal{F}$ of formulas of propositional logic as follows.

- $\mathsf{T} \in \mathcal{F}$, $\mathsf{F} \in \mathcal{F}$, and $x, y, z, \cdots \in \mathcal{F}$ where $x, y, z, \ldots$ are the possible variables we can use.
- If $p, q \in \mathcal{F}$, then
  - $\neg p \in \mathcal{F}$
  - $p \wedge q \in \mathcal{F}$
  - $p \vee q \in \mathcal{F}$
  - $p \to q \in \mathcal{F}$
  - $p \leftrightarrow q \in \mathcal{F}$

Thus, $\mathcal{F}_0 = \{\mathsf{T}, \mathsf{F}, x, y, z, \dots\}$ is just the set of base cases (the so-called "atomic" formulas). $\mathcal{F}_1$ contains $\mathcal{F}_0$ plus everything we can get by one application of the recursive rule:

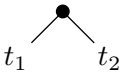
$$\mathcal{F}_1 = \{\mathsf{T}, \mathsf{F}, x, \dots, \neg x, \mathsf{T} \wedge y, z \to \mathsf{F}, x \leftrightarrow y, \dots\}$$

$\mathcal{F}_2$ in turn contains everything in $\mathcal{F}_1$ plus all the things we can get by applying a rule to two things in $\mathcal{F}_1$, for example:

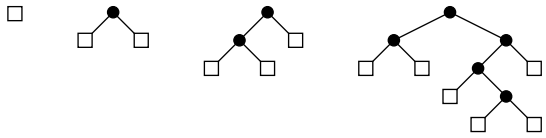
$$\mathcal{F}_2 = \{\dots, (\neg x) \to (\mathsf{T} \wedge y), y \wedge (z \to \mathsf{F}), \dots\}$$

And so on. In this way we get $\mathcal{F}$ as the set of all possible formulas built out of T, F, variables, $\neg$, $\wedge$, $\vee$, $\to$, and $\leftrightarrow$.

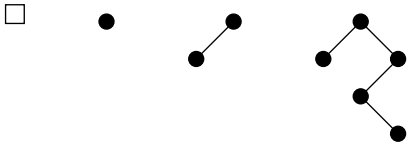
Example. We can define the set of *binary trees* B as follows:

- $\square \in B$ ($\square$ represents the "empty tree")
- If $t_1, t_2 \in B$ then also  $\in B$ (this is a "branch node" with two "children").

For example, various trees in the set B are drawn below:



Note, it is easy to see that every tree will always have empty trees (the little squares) hanging off everywhere on the bottom, so we don't really have to draw them. From now on, we will draw trees without showing the empty tree nodes (they are still there, just not drawn). For example, the trees from above can be drawn like this:



Challenge: how many trees are there with 0 branches? With 1? 2? 3? 4? 5?

49 Bonus material: structural induction

Let S be a recursively defined set, and suppose we wanted to prove something about all elements of S. That is, for some predicate P, we wish to prove

$$\forall s \in S.\ P(s).$$

How would we do it? Induction seems like an obvious choice, but actually it doesn't quite match up: induction, as we've defined it, lets us prove things about all natural numbers, but in this scenario we want to instead prove something about all elements of S.

One way we could do it is by induction on the number of times we have applied the recursive rules for building S. In other words, if S_0 contains the base elements, S_1 is the result of applying the recursive rule(s) to S_0 once, S_2 is the result of applying the rules again, and so on, then we could prove the statement

$$\forall n \in \mathbb{N}.\ \forall s \in S_n.\ P(s)$$

by induction. However, there is a much more straightforward way that more closely matches the way S was defined in the first place, which is the principle of *structural induction*.

To prove $\forall s \in S.\ P(s)$:

- Prove P holds for all the elements of S specified as base case(s).
- For each rule used to build new elements of S, show that if P holds for elements which are already in S, then P also holds for the elements newly constructed by the rule.

Example. Given a binary tree $t \in B$, let $e(t)$ be the number of empty trees ($\square$) contained in t, and $b(t)$ the number of branch nodes ($\bullet$).

Theorem 49.1. *For every $t \in B$, there is one more empty node than branch node, that is,*

$$e(t) = 1 + b(t).$$

Proof. By structural induction.

- First, we must show that the property holds for the empty tree. And indeed,
$$e(\square) = 1 = 1 + b(\square).$$
- Let $t_1, t_2 \in B$ and suppose as our induction hypothesis that the property holds for them, that is $e(t_1) = 1 + b(t_1)$ and $e(t_2) = 1 + b(t_2)$. Then we must show that the property holds also for the new tree with a branch having t_1 and t_2 as its children, call it t.

  The number of empty trees in t is just the sum of the number of empty trees in t_1 and t_2, $e(t) = e(t_1) + e(t_2)$. The number of branch nodes, on the other hand, is one more than the sum of the number in t_1 and t_2, because constructing t adds one new branch node: $b(t) = 1 + b(t_1) + b(t_2)$. Putting these facts together:

$$\begin{array}{ll} & e(t) \\ = & \qquad \{ \quad \text{observation above} \quad \} \\ & e(t_1) + e(t_2) \end{array}$$

$$
\begin{aligned}
&= && \{ \quad \text{IH} \quad \} \\
&\quad 1 + b(t_1) + 1 + b(t_2) \\
&= && \{ \quad \text{algebra} \quad \} \\
&\quad 1 + (1 + b(t_1) + b(t_2)) \\
&= && \{ \quad \text{observation above} \quad \} \\
&\quad 1 + b(t)
\end{aligned}
$$

Thus, by structural induction, this property holds for every tree in B. □

Example. Let's define sets of propositional logic formulas similarly to the example from before, but somewhat simplified for the purposes of this example. First of all, we define $\mathcal{F}$ by:

- $\mathsf{T}, \mathsf{F}, x, y, z, \cdots \in \mathcal{F}$.
- If $p \in \mathcal{F}$, then $\neg p \in \mathcal{F}$.
- If $p, q \in \mathcal{F}$, then $p \wedge q \in \mathcal{F}$, and $p \vee q \in \mathcal{F}$.

So, for example, $\mathcal{F}$ contains things like $\mathsf{T} \wedge \neg x$, or $\neg(x \wedge \neg(y \vee \neg z))$. Now let's define $\mathcal{G}$, which is similar but not quite the same:

- $\mathsf{T}, \mathsf{F}, x, y, z, \ldots, \neg x, \neg y, \neg z, \cdots \in \mathcal{G}$.
- If $p, q \in \mathcal{G}$, then $p \wedge q \in \mathcal{G}$, and $p \vee q \in \mathcal{G}$.

So $\mathcal{G}$ contains negated variables like $\neg x$, but it does not have formulas containing $\neg$ anywhere else. So certain things in $\mathcal{F}$ are also in $\mathcal{G}$, such as $\mathsf{T} \wedge \neg x$. But there are some formulas in $\mathcal{F}$ which are not in $\mathcal{G}$, such as $\neg(x \wedge \neg(y \vee \neg z))$. However, consider this example:

$$
\begin{aligned}
&\quad \neg(x \wedge \neg(y \vee \neg z)) \\
&\equiv && \{ \quad \text{de Morgan} \quad \} \\
&\quad \neg x \vee \neg\neg(y \vee \neg z) \\
&\equiv && \{ \quad \text{double negation} \quad \} \\
&\quad \neg x \vee y \vee \neg z
\end{aligned}
$$

So in this case we are able to transform the original formula, which was not in $\mathcal{G}$, into a logically equivalent one which is in $\mathcal{G}$. Can we always do this? Let's try to prove it.

Theorem 49.2. *For all $p \in \mathcal{F}$, there exists some $q \in \mathcal{G}$ such that $p \equiv q$.*

Proof attempt. By structural induction on $\mathcal{F}$.

- The base elements specified to be in $\mathcal{F}$ are T, F, and variables. Each of these is in $\mathcal{G}$ as well.

- Now suppose that $p \in \mathcal{F}$ and there is some $q \in \mathcal{G}$ such that $p \equiv q$; we must show this is also true for $\neg p$. But here we are kind of stuck. Knowing that there is something equivalent to p in $\mathcal{G}$ doesn't really help us find something in $\mathcal{G}$ equivalent to $\neg p$.

□

Although it seems counterintuitive, in many cases when getting stuck on an inductive proof, the solution is to prove a *stronger* theorem. This is counterintuitive because it seems like proving a stronger theorem would be even harder. But the strength cuts both ways: when proving a stronger theorem we also get to assume a stronger induction hypothesis. The trick is to find the right balance.

In this case, we can actually prove the stronger theorem:

Theorem 49.3. *For all $p \in \mathcal{F}$, there exists some $q \in \mathcal{G}$ such that $p \equiv q$, and some $r \in \mathcal{G}$ such that $\neg p \equiv r$.*

Proof. By structural induction on $\mathcal{F}$.

- The base elements specified to be in $\mathcal{F}$ are T, F, and variables. Each of these is in $\mathcal{G}$ as well, so $\mathcal{G}$ contains things equivalent to them. What about their negations? Well, $\neg\mathsf{T} \equiv \mathsf{F}$, and $\neg\mathsf{F} \equiv \mathsf{T}$, and $\mathcal{G}$ explicitly contains negated variables $\neg x$, $\neg y$, and so on. Hence for all the base elements of $\mathcal{F}$, we see that $\mathcal{G}$ contains things equivalent to both them and their negations.

- Now suppose that $p \in \mathcal{F}$ and there exist $q, r \in \mathcal{G}$ such that $p \equiv q$ and $\neg p \equiv r$; we must show this is also true for $\neg p$. By assumption, r is equivalent to $\neg p$; also, $\neg(\neg p) \equiv p \equiv q$ by assumption. So $\mathcal{G}$ contains things equivalent to both $\neg p$ and its negation.

- Suppose $p_1, p_2 \in \mathcal{F}$ and that $\mathcal{G}$ contains things equivalent to both p_1 and p_2 and their negations—say $p_1 \equiv q_1$, $\neg p_1 \equiv r_1$, $p_2 \equiv q_2$ and $\neg p_2 \equiv r_2$. Let's show $\mathcal{G}$ contains formulas equivalent to both $p_1 \wedge p_2$ and $\neg(p_1 \wedge p_2)$.

  $p_1 \wedge p_2$ is easy: since $q_1, q_2 \in \mathcal{G}$ we know $q_1 \wedge q_2 \in \mathcal{G}$ as well, and so $p_1 \wedge p_2 \equiv q_1 \wedge q_2$ since we assumed $p_1 \equiv q_1$ and $p_2 \equiv q_2$. What about $\neg(p_1 \wedge p_2)$? Well,

$$\neg(p_1 \wedge p_2) \equiv \neg p_1 \vee \neg p_2 \equiv r_1 \vee r_2 \in \mathcal{G}.$$

- The case for $p_1 \vee p_2$ is very similar to the previous case.

□

50 Bonus material: partial orders

50.1 Partial orders

Definition 50.1. A *partial order* is a reflexive, antisymmetric, and transitive relation.

Example. $\leq$ on $\mathbb{Z}$ (or on $\mathbb{N}$, or $\mathbb{R}$, or ...).

Example. The divisibility relation on $\mathbb{N}$ is a partial order.

Example. $\subseteq$ on the power set of a set S.

Example. The relation "must take at the same time as or before" on the set of classes.

We use such relations a lot! We use transitivity to prove relations by a chain of steps, and we often use antisymmetry to prove that things are equal. For example, for sets, we often prove that $A = B$ by proving $A \subseteq B$ and $B \subseteq A$. For natural numbers, we can prove $a = b$ by proving $a \leq b$ and $b \leq a$, or $a \mid b$ and $b \mid a$.

Lemma 50.2. *A partial order $\preccurlyeq$ can never have a cycle $a_1 \preccurlyeq a_2 \preccurlyeq \ldots \preccurlyeq a_n \preccurlyeq a_1$.*

Proof. By transitivity, $a_1 \preccurlyeq a_n$; then, since $a_n \preccurlyeq a_1$, by antisymmetry $a_1 = a_n$. A similar argument shows that $a_1 = a_2 = a_3 = \cdots = a_n$. □

We can visualize posets using a *Hasse diagram*. We draw an edge $a \to b$ when $a \preccurlyeq b$; but if $a \preccurlyeq b$ and $b \preccurlyeq c$ we don't bother drawing an edge $a \to c$ since it is implied by transitivity. Also, since there can never be any cycles, we can draw all the edges pointing in the same direction (typically, up), so we don't actually need to draw arrows on them.

Example. Hasse diagrams for $(\mathbb{N}, \leq)$, for $(\mathbb{N}, |)$, and for $(\mathcal{P}(\{1, 2, 3\}), \subseteq)$.