These are *rough notes* based on my own notes I prepared for the Week 7 lecture. They go with the annotated lecture slides you can see on the "More" page of the course website. Although they are rough, I'm providing them here in case they help you remember what we did in lecture.

1 Legend

Text that should already be on slides.
Descriptions of other things that should already be on slides.

Text I should write during lecture.

Descriptions of other things I should draw or write or do.

2 Countable sets

Definition. A function $f : A \to B$ is *surjective* (or *onto*) if ...

$\forall y \in B.\, \exists x \in A.\, f(x) = y$

(arrow to "if") In definitions, if often means iff.

Definition. A function $f : A \to B$ is *surjective* (or *onto*) if $\forall y \in B.\, \exists x \in A.\, f(x) = y$.

Definition. A set C is *countable* if it is empty or there exists a surjective function $f : \mathbb{N} \to C$.

C	countable?
$\{1, 2, 3\}$	
$\mathbb{N}$	
$\mathbb{Z}$	

(Beside $\{1, 2, 3\}$) Yes: $f(1) = 1, f(2) = 2, f(3) = 3, f(\text{other}) = 1$.

(Beside $\mathbb{N}$) Yes: $f(n) = n$.

(Beside $\mathbb{Z}$) Yes: $0, -1, 1, -2, 2, -3, \ldots$.

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

(circle "it is empty or") Needed? Try to make $f : \mathbb{N} \to \emptyset$. $f(0) = ?$ No functions $f : \mathbb{N} \to \emptyset$ exist.

Exercise: write a simpler definition that doesn't need a special case.

Theorem 1.
- If A and B are countable, so is $A \cup B$.
- If A is countable and $B \subseteq A$, then B is countable.
- If A and B are countable, so is $A \times B$.

We won't prove the whole theorem, but let's see why natural numbers times natural numbers is countable.

$\mathbb{N} \times \mathbb{N}$ is countable:

	0	1	2	...
0	0	1	3	
1	2	4		
2	5			
⋮				

$f(k)$ = coordinates where k appears. E.g. $f(4) = (1, 1)$.

Exercise: write a formula for $f(k)$.

Theorem 2. $\mathbb{Q}$ is countable.

Lemma. If A is nonempty and countable and there is a surjective function $f : A \to B$, then B is countable.

Proof of Theorem 2:

Define $f : \mathbb{Z} \times \mathbb{Z}^+ \to \mathbb{Q}^+$ by $f(x, y) = x/y$.

f is surjective. (Definition of $\mathbb{Q}$?) By Theorem 1, $\mathbb{Z} \times \mathbb{Z}^+$ is countable, so by Lemma, $\mathbb{Q}$ is countable.

Theorem 3. $\{0, 1\}^*$ (the set of finite binary sequences) is countable.

$\lambda, 0, 1, 00, 01, 10, 11, 000, \ldots$

$f : \mathbb{N} \times \mathbb{N} \to \{0, 1\}^*$

How can we define a surjective function from the naturals times the naturals to the set of finite binary sequences?

$$f(i, j) = \begin{cases} j \text{ as an } i\text{-digit-long binary number}, & j < 2^i \\ \lambda, & \text{otherwise} \end{cases}$$

f surjective, so by Lemma, $\{0, 1\}^*$ is countable.

3 Diagonalization

3.1 $\mathcal{P}(\mathbb{N})$ is uncountable

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

If A has n elements, how many elements does $\mathcal{P}(A)$ have?
Is $P(\mathbb{N})$ countable?

- 2^n

If we only want the finite subsets, that's countable. But there's no way to get all the infinite subsets.

(Keep definition of power set.)

Theorem 4. $\mathcal{P}(\mathbb{N})$ is uncountable.

(arrow to "uncountable") not countable

Consider any $f : \mathbb{N} \to \mathcal{P}(\mathbb{N})$.

$x \in S$?	0	1	2	...
f(0)				
f(1)				
f(2)				
⋮				

$f(0) = \emptyset$ *(Fill in $f(0)$ row with F ....)* $f(1) = \mathbb{N}$ *(Fill in $f(1)$ row with T ....)* $f(2) =$ even numbers *(Fill in $f(2)$ row with T F T F ....)*

(Circle the first few diagonal entries.)

Make a new set D which is the opposite of the diagonal. It is not in the range of f.

We've shown that no function from naturals to power set of naturals is a surjection. So, the power set of the naturals is uncountable. Let's turn that into a concise proof.

Proof:

Suppose $\mathcal{P}(\mathbb{N})$ is countable. Then $\exists$ surjective $f : \mathbb{N} \to \mathcal{P}(\mathbb{N})$.

Let $D = \{n \in \mathbb{N} | n \notin f(n)\}$.

D is that set we made by flipping everything on the diagonal.

Since f surjective, $\exists\ d \in \mathbb{N}$ s.t. $f(d) = D$.

So, our set appears on row d in that table.

Is $d \in D$?

Case 1: Suppose $d \in D$. Then $d \in f(d)$, so $d \notin D$. Contradiction.

Case 2: Suppose $d \notin D$. Then $d \notin f(d)$, so $d \in D$. Contradiction.

$\mathcal{P}(\mathbb{N})$ is not countable. (proof by contradiction) □

Cantor's diagonal argument

This is called Cantor's diagonal argument. The method used here is sometimes called *diagonalization*.

3.2 The Halting Problem

The Halting Problem
Determine whether a program will keep running forever, or eventually stop.

The Halting Problem is about figuring out whether a program will eventually halt, meaning stop running.

Who has accidentally written a function that just runs forever instead of returning?

The Halting Problem

(The following code also goes on the slide, but Latex is not cooperating.)

```
def halts(P: str, x: str):
  """Returns True if P is a Python function definition, and
  that function applied to x would eventually finish running
  ("halt").
  """
  ...

>>> halts("""
... def g(x):
...   return x
... """, "hello")
True
>>> halts("""
... def h(x):
...   while True:
...     print("Still going")
... """, "abc")
False
```

(still same slide)
Theorem 5. `halts()` can't be implemented.

It's unusual to be able to prove that a program is impossible to write.

Proof.

For a contradiction, suppose halts() can be implemented.

```
D_str = """
def D(x: str):
  if halts(x, x):
    while True:
      pass
  else:
    return
"""
```

What does D(D_str) do?

Case 1: Suppose halts(D_str, D_str) returns False. Then, from the code, D(D_str) runs forever. So halts must return True. Contradiction.

Case 2: Suppose halts(D_str, D_str) returns True. Then, from the code, D(D_str) returns. So halts must return False. Contradiction.

halts() cannot be implemented. (Proof by contradiction.) □

We can also view this as a table:

halts	P_0	P_1	P_2	...
P_0	T	F	F	
P_1	T	T	F	
P_2	T	F	T	
⋮				

(Arrow to rows and columns.) All syntactically correct Python functions.

If halts(P_i, P_i) == False then D(P_i) halts, so halts(D, P_i) == True.

If halts(P_i, P_i) == True then D(P_i) runs forever, so halts(D, P_i) == False.

(Circle diagonal elements.) D is the opposite of the diagonal, so it cannot be in the table.

We've shown:
- $\mathcal{P}(\mathbb{N})$ is uncountable.
- The Halting Problem is uncomputable.

Others:

- Russell's paradox

There's a story here. For a while, mathematicians were trying to come up with a complete set of axioms that you could use to prove anything in mathematics.

At that time there was a mathematician named Frege. At one point, Frege put together a great mathematical work, laying out a system for doing mathematics. Unfortunately, just as Frege's work was being published, another mathematician, named Bertrand Russell, pointed out a problem.

Let $D = \{s \mid s \notin s\}$ (the set of all sets that don't contain themselves)

$D \in D$ IFF $D \notin D$: contradiction

This was a disaster for Frege's work. His mathematical system lets you define this set D, which leads to a contradiction. That means you can prove any statement you want!

(Arrow to D.) Any system that lets you define this is inconsistent.

Solution: set things up so you can't define D.

Set-builder notation: $\{x \in A \mid \ldots\}$.

(Arrow to "$\in A$".) This is important!

Requiring set-builder notation to start from a bigger set helps us avoid this problem.

But let's go back to that fundamental goal, of finding a universal collection of axioms for doing all of mathematics.

- Gödel incompleteness: every system of axioms is either inconsistent or incomplete.

(Arrow to "inconsistent".) Possible to prove a contradiction

(Arrow to "incomplete".) Some things can't be proved true or false

$D =$ "There is no proof that D is true."

4 Preview: analyzing runtime

(On a slide.)

```
def contains_negative_pair(numbers: List[int]):
  """Returns True if there's some positive integer x such
    that
  x and -x are in numbers."""
  for a in numbers:
    if a == 0:
```

```
      continue
    for b in numbers:
      if a == -b:
        return True
  return False
```

```
>>> contains_negative_pair([1, 2, -1])
True
>>> contains_negative_pair([0, 2, -3])
False
```

How long does `contains_negative_pair` take to run?

> Let $n =$ `len(numbers)`.
>
> *(Annotate "if a == 0" line.)* Runs n times. A few nanoseconds each.
>
> *(Annotate "if a == -b" line.)* Runs $\leq n^2$ times.
>
> Runtime $= an^2 + bn + c$ nanoseconds??

You might be able to write the amount of time this takes as some function like this in nanoseconds. But it will depend on your computer, and whether other processes are running on the same computer, and so on.

All of that is important, but it's also really hard. So computer scientists often do a simpler kind of analysis. The important thing about the runtime is the n squared term.

> Forget about the details: write $O(n^2)$.
>
> Double $n \to$ runtime x4.

The n squared term tells us that every time you double the length of the input, the runtime gets multiplied by approximately four, at least when n is large. That's a useful starting point.

Let's define what big oh of n squared means.

5 Big Oh notation

5.1 Definition of Big Oh

Big O Notation
Let $\mathcal{F}$ be the set of all functions from $\mathbb{N}$ to $\mathbb{R}^+$.
For any $f \in \mathcal{F}$, let

$$O(f) = \{g \in \mathcal{F} \mid \exists c \in \mathbb{R}^+. \exists b \in \mathbb{N}. \forall n \in \mathbb{N}. (n \geq b \text{ IMPLIES } g(n) \leq cf(n))\}$$

In other words:

> $f \in O(g)$ means for all sufficiently large n, $g(n)$ is at most a constant factor times $f(n)$.
>
> Example:
>
> $6n + 4 \in O(3n)$ because when $n \geq 2$, $6n + 4 \leq 8n \leq 3 \cdot 3n$.
>
> Here, $c = 3$ and $b = 2$. Can also choose $c = 4, b = 1$. Or infinitely many other choices.
>
> Conventionally, $O(n^2)$ means $O(f)$ where $f(n) = n^2$.

5.2 Properties of Big Oh

Properties of big O
Summary on "Further reading" page of course website.
(This slide has more stuff, but we didn't get to this slide in lecture.)