Review:

- We have seen that
  - The set of all (Java) programs is countable
  - The set of all functions $f : \mathbb{N}_+ \rightarrow \{0,1,...,9\}$ is not countable

- Let’s review that second proof
  - Consider any listing of such functions $f_1, f_2, f_3, f_4, ...$

Supposed listing of functions: $\mathbb{N}_+ \rightarrow \{0,1,...,9\}$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>$f_2$</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>...</td>
</tr>
<tr>
<td>$f_3$</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>5</td>
<td>7</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>$f_4$</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>...</td>
</tr>
<tr>
<td>$f_5$</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>$f_6$</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>$f_7$</td>
<td>7</td>
<td>1</td>
<td>8</td>
<td>2</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>2</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>$f_8$</td>
<td>6</td>
<td>1</td>
<td>8</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>4</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Flipped Diagonal Function $D: \mathbb{N}_+ \rightarrow \{0,1,...,9\}$

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>5</td>
<td>$\bar{1}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>$f_2$</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>...</td>
</tr>
<tr>
<td>$f_3$</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>5</td>
<td>7</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>$f_4$</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>...</td>
</tr>
<tr>
<td>$f_5$</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>$f_6$</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>$f_7$</td>
<td>7</td>
<td>1</td>
<td>8</td>
<td>2</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>2</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>$f_8$</td>
<td>6</td>
<td>1</td>
<td>8</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>4</td>
<td>1</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Flipping Rule:

- If $f_n(n)$ is 5, make $D(n)=1$
- If $f_n(n)$ is not 5, make $D(n)=5$
**Flipped Diagonal Function $D: \mathbb{N}_+ \rightarrow \{0,1,...,9\}$**

|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ...
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$D$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For all $n$, we have $D \neq f_n$ since $D(n) \neq f_n(n)$

$\Rightarrow$ list was incomplete

$\Rightarrow \{ f \mid f : \mathbb{N}_+ \rightarrow \{0,1,...,9\} \}$ is not countable

**Non-computable functions**

- We have seen that
  - The set of all (Java) programs is countable
  - The set of all functions $f : \mathbb{N}_+ \rightarrow \{0,1,...,9\}$ is not countable

- So...
  - There must be some function $f : \mathbb{N}_+ \rightarrow \{0,1,...,9\}$ that is not computable by any program!

**Back to the Halting Problem**

- Suppose that there is a program $H$ that computes the answer to the Halting Problem

- We will build a table with a row for each program (just like we did for uncountability of reals)

- If the supposed program $H$ exists then the $D$ program we constructed as before will exist and so have a row in the table

- We will see that $D$ must have entries like the “flipped diagonal”
  - $D$ can’t possibly be in the table.
  - Only assumption was that $H$ exists. That must be false.

**Some possible inputs $x$**

<table>
<thead>
<tr>
<th>$&lt;$P$&gt;$ entry is shorthand for CODE($P$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&lt;$P$&gt;$</td>
</tr>
<tr>
<td>$P_1$</td>
</tr>
<tr>
<td>$P_2$</td>
</tr>
<tr>
<td>$P_3$</td>
</tr>
<tr>
<td>$P_4$</td>
</tr>
<tr>
<td>$P_5$</td>
</tr>
<tr>
<td>$P_6$</td>
</tr>
<tr>
<td>$P_7$</td>
</tr>
<tr>
<td>$P_8$</td>
</tr>
<tr>
<td>$P_9$</td>
</tr>
</tbody>
</table>
Some possible inputs $x$

$(P, x)$ entry is 1 if program $P$ halts on input $x$ and 0 if it runs forever.

- $D$ behaves like flipped diagonal.

**Function $D(x)$:**

- if $H(x, x) == \text{true}$ then
  - while (true); /* loop forever */
- else
  - return; /* do nothing and halt */
- endif

Recall: code for $D$ assuming subroutine $H$ that solves the halting problem.

- If $D$ existed it would have a row different from every row of the table.
- $D$ can’t be a program so $H$ cannot exist!
More than just halting is hard

- We showed
  - if the hypothetical program $H$ deciding the Halting Problem existed, then we could use it to build a program $D$ that cannot possibly exist
  - Since $D$ doesn’t exist, program $H$ cannot exist

- We will use similar approach to show that other important problems are hard
  - if there is a hypothetical program $A$ solving one of these problems, then we could use it to build a program $H$ solving the Halting Problem
  - Since $H$ doesn’t exist, $A$ cannot exist

But first another hard halting-related problem

**Halting Problem:**

- **Given:** - CODE($P$) for any program $P$
  - input $x$
- **Output:**  
  - true if $P$ halts on input $x$
  - false if $P$ does not halt on input $x$

**HaltsNoInput Problem:**

- **Given:** - CODE($Q$) for any program $Q$
- **Output:**  
  - true if $Q$ halts without reading any input
  - false if $Q$ reads input or runs forever without reading any input.
Key idea: Hardcoding an Input

INPUT is “potato”

```java
public String P(String y) {
    return new String(
        Arrays.sort(y.toCharArray()
    );
}
```

Q: Version of P with “hardcoded” input:

```java
public String Q() {
    return new String(
        Arrays.sort("potato".toCharArray()
    );
}
```

Q() behaves the same as P("potato"), except that it doesn’t read any input.

Can write a program Hardcoder that, given CODE(P) and an input string x, produces CODE(Q)

Showing there is no program solving HaltsNoInput

Suppose that hypothetical program A solves HaltsNoInput problem.

```latex
\begin{align*}
\text{A outputs true} & \quad \text{iff Q() reads no input} \\
\text{and (always) halts}
\end{align*}
```

Some notation: Decision problems as sets

Every decision problem can be written as asking about membership in a set.

If a program “decides” a set, then it must output true on all inputs in the set and false on all inputs not in the set.

Halt =\{(CODE(P),x) : P is a program that halts on input x\}

HaltsNoInput =\{CODE(Q) : Q is a program that halts without reading any input\}
Convenient pictures

- Rather than continue to come up with more names like H and A for our hypothetical programs...
- Given a decision problem SET we use the following picture to denote any hypothetical program that solves decision problem SET with ANS denoting its output.

![Diagram of SET with ANS]

Showing HELLO is Undecidable

Consider the set:

**HELLO** = \{CODE(R) : R is a program that reads no input, prints “Hello”, and always halts\}

**CODE(Q)**

**Question:** Does Q() halt?

Step 1: Remove all System.out.print/println statements from CODE(Q).
Step 2: Append System.out.println(“Hello”) at the end of the program code.

Call the new program R

**CODE(R)**

**Question:** Does R() print “Hello” and halt?

Answering with ANS would solve HaltsNoInput!

A Decision Problem We Can Solve

**REGEQUIV** = \{(R₁, R₂) : R₁ and R₂ are equivalent regexps\}

In this case the hypothetical program does exist:

- Convert both to NFAs then DFAs, minimize and compare

![Diagrams of REGEQUIV with examples]

Showing EQUIV is Undecidable

Consider the set:

**EQUIV** = \{(CODE(P), CODE(R)) : P, R are programs, P(x) = R(x) for all inputs x\}

**CODE(Q)**

**Question:** Does Q() halt?

Step 1: Construct P:
  ```java
  public static boolean P() {return true;}
  ```
Step 2: Construct R:
  ```java
  Step a: Replace return type of Q with boolean
  Step b: Replace all return values with true
  Step c: Add “return true;” to the end of the program
  Call this program R
  ```

**CODE(P),CODE(R)**

**Question:** Are P and R Equivalent?

Answering with ANS would solve HaltsNoInput!
Pitfalls

- Not every problem on programs is undecidable! Which of these is decidable?

- Input \texttt{CODE(P) and x}
  Output: \texttt{true} if \texttt{P} prints “ERROR” on input \texttt{x}
  after less than 100 steps
  \texttt{false} otherwise

- Input \texttt{CODE(P) and x}
  Output: \texttt{true} if \texttt{P} prints “ERROR” on input \texttt{x}
  after more than 100 steps
  \texttt{false} otherwise