Announcements:

- Midterm 1 next week:
  - Logistics announcement by Thursday
  - Prep Material: Canvas → Files → Midterm Prep
  - Likely: 2 pages of typset cheatsheet allowed. No other external refs.
- Homework 3 due Thursday

- Advanced Sections: Christina Ilvento on Differential Privacy!

  Thursdays at 4:30.
Where we are:

Part I: Circuits:
Finite computation, quantitative study

Part II: Automata:
Infinite restricted computation, quantitative study

Part III: Turing Machines:
Infinite computation, qualitative study

Part IV: Efficient Computation:
Infinite computation, quantitative study

Part V: Randomized computation:
Extending studies to non-classical algorithms
Today:

- Definition of Turing Machines
- A function $F$ not computable by DFA or Circuits
- Computing $F$ with Turing Machine
Definition of Turing Machine (TM)

- Recall: DFA = Finite state control + input on tape + move right on each step.
- In a nutshell: TM = DFA + “Write” + “Move left+right on tape”
  - (Either “Write” / “Move left+right” on its own insufficient)
- TM: Main change:
  - More Involved Transition function: $T$ (now $\delta$):
    - $\delta$: (current state, read symbol) $\mapsto$ (new state, write symbol, direction of move/halt)
  - Explicit halting (don’t just end after reading last input bit)
    - Computes functions: output = concatenation of $\{0,1\}$ symbols on tape.
Formal Definition

- (Barak, Definition 7.1):
- TM on $k$ states and alphabet $\Sigma \equiv \{0,1,\triangleright,\phi\}$
  is given by $\delta: [k] \times \Sigma \rightarrow [k] \times \Sigma \times \text{Action}$,
  where $\text{Action} = \{L,R,S,H\}$
  - $L=$Left, $R=$Right, $S=$Stay (don’t move), $H=$Halt (done!!)
- Operation:
  - Start in state 0, Tape $T = \Phi x_0 \ldots x_{n-1} \phi \phi \phi \ldots$, Head ($i$) at $x_0$
  - General step: current state $q$ ; input symbol $\sigma$:
    Let $\delta(q,\sigma) = (r,\tau,X) \Rightarrow$ Write $\tau$ on tape (overwriting $\sigma$) ; Move to state $r$;
    Move Head left ($i \leftarrow i - 1$) if $X = L$; right if $X = R$; don’t move if $X = S$.
  - Repeat General step until $X = H$
**TM Example**

- Example: \( k = 1; \Sigma = \{0,1,\triangleright,\phi\} \); \( \delta(0,\sigma) = \begin{cases} 
(0,0,R) & \text{if } \sigma \in \{0,1\} \\
(0,\phi,H) & \text{if } \sigma \notin \{0,1\} 
\end{cases} \)

- What does TM output on \( \triangleright101\phi \ldots \) (in future, we won’t write \( \triangleright \) or \( \phi \))

Output: 000
TM Example

• Example: $k = 1; \Sigma = \{0, 1, \triangleright, \phi\}; \delta(0, \sigma) = \begin{cases} (0, 0, R) \text{ if } \sigma \in \{0, 1\} \\ (0, \phi, H) \text{ if } \sigma \notin \{0, 1\} \end{cases}$

• What does TM output on $\triangleright 101\phi$ ... (in future, we won’t write $\triangleright$ or $\phi$)

• What function does TM compute?

Def: Output of Turing Machine = concatenation of Os & 1s on tape.

$01A11B10 \Rightarrow 011110$
A “hard” function

- \( f: \{0,1\}^* \rightarrow \{0,1\}, \ f(x) = 1 \iff x = 1^n \ \text{for} \ n = 2^t \ \text{for integer} \ t \)

\[
\begin{align*}
  f(\text{null}) &= 0 \\
  f(1) &= 1 \\
  f(1^2) &= 1 \\
  f(1^{256}) &= 1
\end{align*}
\]

\( f(1^{200}) = 0 \)
Exercise Break 1

• $f: \{0,1\}^* \to \{0,1\}$, $f(x) = 1 \iff x = 1^n$ for $n = 2^t$ for integer $t$

• (30 sec) Prove that no circuit computes $f$

• (4 min 30 sec) Prove no DFA computes $f$
  • Part 1: Focus on big idea; defer calculations/parameter settings.
  • Part 2: Get your hands dirty; do calculations+parameter settings.
if length is $2^t$ of input then __ must be a star in $\varepsilon$.

Then $g < 2^t$

\[ a \quad b \quad c \]
\[ \quad 1 \quad 1 \quad 1 \]

matches \( \ast \)

\[ a+b+c=2^t \Rightarrow f(1) = 1 \]

\[ a+2b+c \]

\( f(1) = 1 \)
DFA has $C$ states

if

$\exists x \& y$ s.t.

$9_x = 9_y$

$9_x = 9_y$

$\Rightarrow 1^y \cdot 1^z = 1^{2t}$

$1^x \cdot 1^y = 1^{2t + x - y}$
Hint: \( t = 2^y - y \)

Can prove that \( 2^y + x - y \) is not a power of 2.
F is computable by a Turing Machine

Main Idea: Loop many times:
   Scan string left to right
   Replace every alternate 1 by 0;
   reject if number of 1s is odd and greater than 2.
Say \( \exists \) DFA with \( C \) states computing \( f \).

- \( q_i \) = state after input \( 1^i \), \( i = 0, 1, 2, \ldots, C + 1, \ldots \)

\( \exists 1 \leq x < y \leq k + 1 \) s.t.

\[
q_x = q_y
\]

\[ z = 2^y - y \]

\[
\begin{cases}
1 \cdot 1^z & \Rightarrow \text{must reject} \\
1^y \cdot 1^z = 1^2y & \Rightarrow \text{must accept.}
\end{cases}
\]

\[
2 - y + x + 2^t
\]

for any \( t \).
More details: Alphabet & States

Alphabet $\Sigma = \{0, 1, \triangleright, \phi, \#\}$

0. Start/Not seen any ones
1. Move Right first one
2. Move Right even # of ones
3. Move Right odd # of ones
4. Move Left
5. Clean Right and Reject
6. Clean Left and Reject
Input: 1 1 0 1 1 1 1 1 1

→ State: 0,1,2 → clean right & reject

→ Clean left & reject
<table>
<thead>
<tr>
<th>State/Input</th>
<th>▶</th>
<th>0</th>
<th>1</th>
<th>(\phi)</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>invalid</td>
<td>(5,#, R)</td>
<td>(1,1, R)</td>
<td>(6,0,L)</td>
<td>(0,#, R)</td>
</tr>
<tr>
<td>1</td>
<td>invalid</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>invalid</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>invalid</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>(0, Đ, R)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>(- Đ, H)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Alphabet \(\Sigma = \{0,1, \triangleright, \phi, \#\}\)

0. Start/Not seen any ones
1. Move Right first one
2. Move Right even # of ones
3. Move Right odd # of ones
4. Move Left
5. Clean Right and Reject
6. Clean Left and Reject
\( f: \{0,1\}^* \rightarrow \{0,1\}, \ f(x) = 1 \iff x = 1^n \text{ for } n = 2^t \text{ for integer } t \)

<table>
<thead>
<tr>
<th>State/Input</th>
<th>▶</th>
<th>0</th>
<th>1</th>
<th>(\phi)</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Alphabet \( \Sigma = \{0,1, ▶, \phi, #\} \)

0. Start/Not seen any ones
1. Move Right first one
2. Move Right even # of ones
3. Move Right odd # of ones
4. Move Left
5. Clean Right and Reject
6. Clean Left and Reject

Exercise Break 2:

Fill in rows for states 2 & 4

Keep answer ready (5 triples) to type into chat. Use D for ▶
### Alphabet
\[ \Sigma = \{0,1,\triangleright,\phi,#\} \]

### States and Transitions

<table>
<thead>
<tr>
<th>State/Input</th>
<th>(\triangleright)</th>
<th>0</th>
<th>1</th>
<th>(\phi)</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>invalid</td>
<td>(5,#,R)</td>
<td>(1,1,R)</td>
<td>(-,0,H)</td>
<td>(0,#,R)</td>
</tr>
<tr>
<td>1</td>
<td>invalid</td>
<td>(5,#,R)</td>
<td>(2,#,R)</td>
<td>(-,#,H)</td>
<td>(1,#,R)</td>
</tr>
<tr>
<td>2</td>
<td>invalid</td>
<td>(5,#,R)</td>
<td>(3,#,R)</td>
<td>(4,\phi,L)</td>
<td>(2,#,R)</td>
</tr>
<tr>
<td>3</td>
<td>Invalid</td>
<td>(5,#,R)</td>
<td>(2,1,R)</td>
<td>(6,0,L)</td>
<td>(3,#,R)</td>
</tr>
<tr>
<td>4</td>
<td>(0,\triangleright,R)</td>
<td>invalid</td>
<td>(4,1,L)</td>
<td>invalid</td>
<td>(4,#,L)</td>
</tr>
<tr>
<td>5</td>
<td>invalid</td>
<td>(5,#,R)</td>
<td>(5,#,R)</td>
<td>(6,0,L)</td>
<td>(5,#,R)</td>
</tr>
<tr>
<td>6</td>
<td>(-, \triangleright,H)</td>
<td>invalid</td>
<td>(6,#,L)</td>
<td>invalid</td>
<td>(6,#,L)</td>
</tr>
</tbody>
</table>

0. Start/Not seen any ones
1. Move Right first one
2. Move Right even # of ones
3. Move Right odd # of ones
4. Move Left
5. Clean Right and Reject
6. Clean Left and Reject
Summary & Next

- Achieved today:
  - Defined TM
  - Shown it computes one function that DFA and circuits can’t

- Next Lecture:
  - More examples.
  - Towards equivalence with (all) programs