Reminder

Midterm 1

February 11th, 18:30 - 21:30

Covers Lectures 1-6 (first 3 homeworks)
What have we done so far?

> Introduction to the course

*Computer science is no more about computers than astronomy is about telescopes.*

> Logic

Foundation of mathematics

> Formalization of computation/algorithm

Deterministic Finite Automata

Turing Machines

> The study of computation

Computability
What have we done so far?

> The study of computation

Computability

- Most problems are *undecidable*.
- Some very interesting problems are *undecidable*.

But most interesting problems are *decidable*!
What is next?

> The study of computation

Computability

Computational Complexity (Practical Computability)

- How do we define complexity?
- What is the right level of abstraction to use?
- How do we analyze complexity?
- What are some interesting problems to study?
- What can we do to better understand the complexity of problems?
What is next?

1 million dollar question
(or maybe 6 million dollar question)

\[ P = NP \]
Introduction to Computational Complexity

Computational complexity of an algorithm.

Computational complexity of a problem.
  - complexity of the *best* algorithm computing the problem.

Complexity with respect to what?
  - time (number of steps)
  - space (memory)
  - randomness
  - quantum resources

Our focus for now
Church-Turing Thesis:

With respect to \textit{computability} the particular computational model doesn’t matter.

Unfortunately, this is \textbf{not} true with respect to computational complexity.

The model makes a difference.
How the model can affect running time
A multitape Turing machine

Ordinary TM with multiple tapes, each with its own head.

Number of tapes is fixed (cannot grow with input size).

\[ \delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R\}^k \]

e.g. \( (q_i, a_1, \ldots, a_k) \mapsto (q_j, b_1, \ldots, b_k, R, L, \ldots, L) \)
A multitape Turing machine

Ordinary TM with multiple tapes, each with its own head.

Is it more powerful?

Every multitape TM has an equivalent single tape TM.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

On input string w:
- Scan the input and **reject** if a 0 is found to the right of a 1.
- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape, cross off a single 0 and a single 1.
- If 0s remain but no 1s remain **or** 1s remain but no 0s remain **reject**
- Else **accept**

**Number of steps:** \[ O(n^2) \] \( (n \text{ is the input length.}) \)
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a two-tape TM take?

On input string \( w \):
- Scan the input and reject if a 0 is found to the right of a 1.
- Scan the 0s until the first 1, and copy the 0s to Tape 2.
- Scan the 1s. For each 1 read, cross off a 0 on Tape 2.
- If all 0s are crossed off before all 1s are read, reject.
- If all 0s are crossed off accept.
- Else reject (some 0s remain).

Number of steps: \( O(n) \)
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

On input string w:
- Scan the input and **reject** if a 0 is found to the right of a 1.
- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (# of 1s + # of 0s) is odd, **reject**.
  - Scan the tape. Cross off every other 0 starting with first 0.
    Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain **accept**.
- Else **reject**.
How the model can affect time

\[ L = \{0^k 1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

$L = \{0^k 1^k : k \geq 0\}$

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (# of 1s + # of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (# of 1s + # of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0.
    Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

$L = \{0^k1^k : k \geq 0\}$

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (# of 1s + # of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

$L = \{0^k1^k : k \geq 0\}$

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (# of 1s + # of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k 1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (# of 1s + # of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

$L = \{0^k1^k : k \geq 0\}$

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (# of 1s + # of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

$L = \{0^k1^k : k \geq 0\}$

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (# of 1s + # of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (# of 1s + # of 0s) is odd, \textbf{reject}.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain \textbf{accept}.
- Else \textbf{reject}.
How the model can affect time

$$L = \{0^k1^k : k \geq 0\}$$

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
  - If no 0s and no 1s remain accept.
  - Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (# of 1s + # of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k 1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0.
    Cross off every other 1 starting with first 1.
  - If no 0s and no 1s remain accept.
  - Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (# of 1s + # of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
  - If no 0s and no 1s remain accept.
  - Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0.
    Cross off every other 1 starting with first 1.
  - If no 0s and no 1s remain accept.
  - Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k 1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

$L = \{0^k1^k : k \geq 0\}$

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (# of 1s + # of 0s) is odd, **reject**.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain **accept**.
- Else **reject**.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, **reject**.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain **accept**.
- Else **reject**.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (# of 1s + # of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0.
    Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

$L = \{0^k 1^k : k \geq 0\}$

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (# of 1s + # of 0s) is odd, **reject**.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain **accept**.
- Else **reject**.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
  - If no 0s and no 1s remain accept.
  - Else reject.
How the model can affect time

$L = \{0^k1^k : k \geq 0\}$

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (# of 1s + # of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0.
    Cross off every other 1 starting with first 1.
  - If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{ 0^k 1^k : k \geq 0 \} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (number of 1s + number of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

$L = \{0^k1^k : k \geq 0\}$

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (# of 1s + # of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
  - If no 0s and no 1s remain accept.
  - Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

$L = \{0^k1^k : k \geq 0\}$

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (number of 1s + number of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

$L = \{0^k1^k : k \geq 0\}$

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (# of 1s + # of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0.
    Cross off every other 1 starting with first 1.
  - If no 0s and no 1s remain accept.
  - Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (# of 1s + # of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

\[ L = \{0^k 1^k : k \geq 0\} \]

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
- If no 0s and no 1s remain accept.
- Else reject.
How the model can affect time

$L = \{0^k 1^k : k \geq 0\}$

How many steps does a single-tape TM take?

- Repeat while both 0s and 1s remain on the tape:
  - Scan the tape. If (\# of 1s + \# of 0s) is odd, reject.
  - Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.
  - If no 0s and no 1s remain accept.
  - Else reject.

**Number of steps:** $O(n \log n)$
$L = \{0^k1^k : k \geq 0\}$

Why is it correct?

- Scan the tape. If (# of 1s + # of 0s) is odd, reject.
- Scan the tape. Cross off every other 0 starting with first 0. Cross off every other 1 starting with first 1.

(# of 1s + # of 0s) is odd if and only if
(# of 1s) and (# of 0s) have different parities.

Sequence of parities of (# of 1s) —> binary representation of (# of 1s) in the input
Sequence of parities of (# of 0s) —> binary representation of (# of 0s) in the input
How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

Can we do better?

\[ O(n \log n) \text{ is the best for 1-tape TMs.} \]

\[ O(n) \text{ is the best for 2-tape TMs.} \]
How the model can affect time

\[ L = \{0^k 1^k : k \geq 0\} \]

A function in Python:

```python
def twoFingers(s):
    lo = 0
    hi = len(s) - 1
    while (lo < hi):
        if (s[lo] != 0 or s[hi] != 1):
            return False
        lo += 1
        hi -= 1
    return True
```

# of steps

<table>
<thead>
<tr>
<th># of steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>3? 4? 5?</td>
</tr>
</tbody>
</table>

Seems like \( O(n) \)
How the model can affect time

\[ L = \{0^k 1^k : k \geq 0\} \]

\[ h_i = 1 \]

Initially \( h_i = n-1 \) (the length of the input - 1)

How many bits to store \( h_i \) ? \( \sim \log_2 n \)

What if \( n \) is a power of 2?

\[ h_i = 10000000\ldots0 \]

\[ h_i = 0111111\ldots1 \]

\( \sim \log_2 n \) steps
# How the model can affect time

\[ L = \{0^k1^k : k \geq 0\} \]

A function in Python:

```python
def twoFingers(s):
    lo = 0
    hi = len(s)-1
    while (lo < hi):
        if (s[lo] != 0 or s[hi] != 1):
            return False
        lo += 1
        hi -= 1
    return True
```

<table>
<thead>
<tr>
<th># of steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
</tr>
<tr>
<td>[1]</td>
</tr>
<tr>
<td>[1]</td>
</tr>
<tr>
<td>[3? 4? 5?]</td>
</tr>
<tr>
<td>[\log n ?]</td>
</tr>
<tr>
<td>[O(n \log n) ?]</td>
</tr>
</tbody>
</table>
How the model can affect time

$L = \{0^k 1^k : k \geq 0\}$

if $(s[lo] \neq 0 \text{ or } s[hi] \neq 1)$:

Initially $lo = 0$, $hi = n-1$

Does it take $n$ steps to go from $s[0]$ to $s[n-1]$?
How the model can affect time

\[ L = \{0^k 1^k : k \geq 0\} \]

A function in Python:

```python
def twoFingers(s):
    lo = 0
    hi = len(s)-1
    while (lo < hi):
        if (s[lo] != 0 or s[hi] != 1):
            return False
        lo += 1
        hi -= 1
    return True
```

<table>
<thead>
<tr>
<th># of steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>( O(n^2) )</td>
</tr>
</tbody>
</table>
How the model can affect time

SO

Number of steps (running time) depends on the particular model you choose.

Which one is the best model?

No such thing.

1. Be clear about what the model is!
2. All reasonable deterministic models are polynomially equivalent.
How the model can affect time

Which model does this correspond to?

```python
def twoFingers(s):
    lo = 0
    hi = len(s)-1
    while (lo < hi):
        if (s[lo] != 0 or s[hi] != 1):
            return False
        lo += 1
        hi -= 1
    return True
```

Which model does this correspond to?

\[ O(n) \]
How the model can affect time

The Random-Access Machine (RAM) model

Good combination of reality/simplicity.

+ , - , / , *, <, >, etc. e.g. 245*12894 take 1 unit time
memory access e.g. A[94] takes 1 unit time

Note:

Good model when, say, you work with int data type.

Not a good model if you are working with 1000000-digit numbers.

Which model are we going to use?
Defining time complexity of an algorithm and Intrinsic complexity of a problem
Defining running time

Recall:

A computational problem $P$ is just a function

$$P : \Sigma^* \rightarrow \Sigma^*$$

that maps instances to solutions.

If $P$ is of the form $P : \Sigma^* \rightarrow \{0, 1\}$

it is called a decision problem.

An algorithm solves $P$ if it outputs the correct solution
on every instance.
Defining running time

With a specific computational model in mind:

**Definition:**

The running time of an algorithm $A$ is a function

$$T_A : \mathbb{N}^+ \rightarrow \mathbb{N}^+$$

defined by

$$T_A(n) = \max_{\text{instances } I \text{ of size } n} \{ \# \text{ steps } A \text{ takes on } I \}$$

We drop the subscript $A$, and write $T(n)$ when $A$ is clear.

$n$ always denotes the input length.
Why worst-case?

We are not dogmatic about it.

Can study “average-case” (random inputs)
Can try to look at “typical” instances.
Can do “smoothed analysis”.

... 

**BUT** worst-case analysis has its advantages:

- An ironclad guarantee.
- Matches our worst-case notion of an alg. solving a prob.
- Hard to define “typical” instances.
- Random instances are often not representative.
- Often much easier to analyze.
Defining intrinsic complexity

With a specific computational model in mind:

The **intrinsic complexity** (with respect to running time) of a problem

$$P : \Sigma^* \rightarrow \Sigma^*$$

is defined by

$$\min_{\text{algorithms } A \text{ that solve } P} T_A(\cdot)$$

How do you compare functions?

$$n^2 \leq 100n$$
The CS way to compare functions:

\[ O(\cdot) \quad \Omega(\cdot) \quad \Theta(\cdot) \]

\[ \leq \quad \geq \quad = \]
Our notation for $\leq$ when comparing functions.

The idea is that these functions represent computational complexity (e.g. time complexity)

We want to use the right level of abstraction!

“Sweet spot”
- coarse enough to suppress details like programming language, compiler, architecture,…
- sharp enough to make comparisons between different algorithmic approaches.
Big Oh

$$8n^2 - 3n + 84$$

Analogous to “too many significant digits”.

$$O(n^2)$$

- We don’t care about constant factors.
  (even a change in alphabet size leads to constant factor difference)

What if the running time is $$10^{20} n^2$$?

- We don’t care about small values of n.
  (the only interesting instances are the big ones)
**Informal:** An upper bound that suppresses constant factors and ignores small $n$.

Suppressing constant factors means suppressing lower order additive terms.

\[ n^2 + 100n + 500 \text{ is } O(n^2) \]

\[ 601n^2 = n^2 + 100n^2 + 500n^2 > n^2 + 100n + 500 \]
Big Oh

$n^2 + 100n + 500$

$n^2$

$n$
**Big Oh**

**Informal:** An upper bound that suppresses constant factors and ignores small \( n \).

For \( f, g : \mathbb{N}^+ \to \mathbb{R}^+ \)

\[
f(n) = O(g(n)) \quad \text{roughly means} \quad f(n) \leq g(n) \quad \text{up to a constant factor and ignoring small } n.
\]

**Formal Definition:**

For \( f, g : \mathbb{N}^+ \to \mathbb{R}^+ \), we say \( f(n) = O(g(n)) \) if there exists constants \( C, n_0 > 0 \) such that

\[
f(n) \leq Cg(n) \quad \text{for all } n \geq n_0.
\]

( \( C \) and \( n_0 \) cannot depend on \( n \).)
Big Oh

Formal Definition:

For $f, g : \mathbb{N}^+ \to \mathbb{R}^+$, we say $f(n) = O(g(n))$ if there exists constants $C, n_0 > 0$ such that $f(n) \leq C g(n)$ for all $n \geq n_0$.

($C$ and $n_0$ cannot depend on $n$.)

\[ \text{Graph showing } f(n), g(n), 2g(n) \text{ for } n \geq n_0. \]
Big Oh

Formal Definition:

For \( f, g : \mathbb{N}^+ \rightarrow \mathbb{R}^+ \), we say \( f(n) = O(g(n)) \) if there exists constants \( C, n_0 > 0 \) such that

\[
f(n) \leq Cg(n) \quad \text{for all } n \geq n_0.
\]

( \( C \) and \( n_0 \) cannot depend on \( n \).)

Example:

\[
f(n) = 3n^2 + 10n + 30 \quad \quad g(n) = n^2
\]

\[f(n) = O(g(n))\]

Take \( C = 4, \quad n_0 = 13\)

\[
3n^2 + 10n + 30 \leq 4n^2 \quad \text{when} \quad n \geq 13
\]
Big Oh

Example:

\[ f(n) = 3n^2 + 10n + 30 \quad \text{and} \quad g(n) = n^2 \]

\[ f(n) = O(g(n)) \]

Take \( C = 4, \ n_0 = 13 \)

\[ 3n^2 + 10n + 30 \leq 4n^2 \quad \text{when} \quad n \geq 13 \]

Proving \( f(n) \) is \( O(g(n)) \) is like a game:

You pick constants \( C, n_0 \)

Adversary picks \( n \geq n_0 \)

You win if \( f(n) \leq Cg(n) \)

You need to make sure you always win.
Big Oh

1000n is $O(n)$

0.00000001n is $O(n)$

$0.1n^2 + 10^{20}n + 10^{10000}$ is $O(n^2)$

$n$ is $O(2^n)$

$0.0000001n^2$ is not $O(n)$

$n \log n$ is not $O(n)$

Note on notation:

People usually write $4n^2 + 2n = O(n^2)$

Better notation would be $4n^2 + 2n \in O(n^2)$
## Run time scaling

<table>
<thead>
<tr>
<th>Running-time:</th>
<th>Ratio:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c \cdot n$</td>
<td>double the input</td>
</tr>
<tr>
<td>$c \cdot n^2$</td>
<td>double the input</td>
</tr>
<tr>
<td>$c \cdot n^3$</td>
<td>double the input</td>
</tr>
<tr>
<td>$c \cdot n^k$</td>
<td>double the input</td>
</tr>
<tr>
<td>$c \cdot 2^n$</td>
<td>double the input</td>
</tr>
</tbody>
</table>
Common Big Oh classes and their names

Constant: \( O(1) \)

Logarithmic: \( O(\log n) \)

Square-root: \( O(\sqrt{n}) = O(n^{0.5}) \)

Linear: \( O(n) \)

Loglinear: \( O(n \log n) \)

Quadratic: \( O(n^2) \)

Polynomial: \( O(n^k) \)

Exponential: \( O(k^n) \)
Big Oh
How much smaller is log \( n \) compared to \( n \)?

<table>
<thead>
<tr>
<th>( n )</th>
<th>( \log n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>128</td>
<td>7</td>
</tr>
<tr>
<td>1024</td>
<td>10</td>
</tr>
<tr>
<td>1,048,576</td>
<td>20</td>
</tr>
<tr>
<td>1,073,741,824</td>
<td>30</td>
</tr>
<tr>
<td>1,152,921,504,606,846,976</td>
<td>60</td>
</tr>
</tbody>
</table>

\( \sim 1 \) quintillion
How much smaller is \( n \) compared to \( 2^n \)?

<table>
<thead>
<tr>
<th>( 2^n )</th>
<th>( n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>128</td>
<td>7</td>
</tr>
<tr>
<td>1024</td>
<td>10</td>
</tr>
<tr>
<td>1,048,576</td>
<td>20</td>
</tr>
<tr>
<td>1,073,741,824</td>
<td>30</td>
</tr>
<tr>
<td>1,152,921,504,606,846,976</td>
<td>60</td>
</tr>
</tbody>
</table>
If your algorithm has exponential running time
e.g. $\sim 2^n$

No hope of being practical.
Exponential running time: Example

Given a list of integers, determine if there is a subset of the integers that sum to 0.

| 4 | -3 | -2 | 7 | 99 | 5 | 1 |
Exponential running time: Example

Given a list of integers, determine if there is a subset of the integers that sum to 0.

| 4 | -3 | -2 | 7 | 99 | 5 | 1 |

Exhaustive Search (Brute Force Search):
Try every possible subset and see if it sums to 0.

Number of subsets is $2^n$
So running time is at least $2^n$
\[\log n \lll \sqrt{n} \lll n < n \log n \lll n^2 \lll n^3 \lll 2^n \lll 3^n\]

The theoretical divide between efficient and inefficient:

If it is not \( O(n^k) \) for some constant \( k \)

(if it does not have polynomial complexity)

then it is inefficient.
Some exotic functions

1

log* n

log log n

log n

\sqrt{n}

n / log n

n

n log n

n^2

n^3

n^{O(1)}

n^{\log n}

2^n

3^n

n!

n^n

2^{2^n}

2^{2^{2^{\cdots}}}

n \text{ times}

Fastest algorithm for multiplication:

\[ n \cdot (\log n) \cdot 2^{O(\log^* n)} \]
If $O(\cdot)$ is like $\leq$

$\Omega(\cdot)$ is like $\geq$

$O(\cdot)$

**Informal:** An upper bound that suppresses constant factors and ignores small n.

$\Omega(\cdot)$

**Informal:** A lower bound that suppresses constant factors and ignores small n.
Big Omega

\( \Omega(\cdot) \)

**Informal:** A lower bound that suppresses constant factors and ignores small \( n \).

**Formal Definition:**

For \( f, g : \mathbb{N}^+ \to \mathbb{R}^+ \), we say \( f(n) = \Omega(g(n)) \) if there exists constants \( c, n_0 > 0 \) such that

\[ f(n) \geq cg(n) \quad \text{for all} \quad n \geq n_0. \]

(\( c \) and \( n_0 \) cannot depend on \( n \).)
Formal Definition:

For $f, g : \mathbb{N}^+ \to \mathbb{R}^+$, we say $f(n) = \Omega(g(n))$ if there exists constants $c, n_0 > 0$ such that

$$f(n) \geq cg(n)$$

for all $n \geq n_0$.

($c$ and $n_0$ cannot depend on $n$.)

---

**Big Omega**
Big Omega

Some Examples:

\[ 10^{-10} n^4 \text{ is } \Omega(n^3) \]

\[ 0.001n^2 - 10^{10} n - 10^{30} \text{ is } \Omega(n^2) \]

\[ n^{0.0001} \text{ is } \Omega(\log n) \]

\[ n^{1.0001} \text{ is } \Omega(n \log n) \]
If \( \Theta(\cdot) \) is like \( \leq \)

and \( \Omega(\cdot) \) is like \( \geq \)

\( \Theta(\cdot) \) is like \( = \)
Theta

**Formal Definition:**

For \( f, g : \mathbb{N}^+ \to \mathbb{R}^+ \), we say \( f(n) = \Theta(g(n)) \) if

\[
f(n) = O(g(n)) \quad \text{and} \quad f(n) = \Omega(g(n)).
\]

**Equivalently:**

There exists constants \( c, C, n_0 \) such that

\[
 cg(n) \leq f(n) \leq Cg(n) \quad \text{for all} \quad n \geq n_0.
\]
Back to intrinsic complexity
Defining intrinsic complexity

With a specific computational model and resource in mind:

The **intrinsic complexity** of a problem is the complexity of the most efficient algorithm solving it.
Intrinsic complexity

If you give an algorithm that solves a problem

upper bound on the intrinsic complexity

How to show a lower bound on the intrinsic complexity?

Argue against all possible algorithms that solves the problem.

The dream: Get a matching upper and lower bound.
Example

\[ L = \{0^k 1^k : k \geq 0\} \]

def twoFingers(s):
    lo = 1
    hi = len(s)
    while (lo < hi):
        if (s[lo] != 0 or s[hi] != 1):
            return False
        lo += 1
        hi -= 1
    return True

In the RAM model:

\[ O(n) \]

Could there be a faster algorithm?

e.g. \[ O(n / \log n) \]
Example

\[ L = \{0^k1^k : k \geq 0\} \]

**Fact:** Any algorithm that decides \( L \) must use \( \geq n \) steps.

**Proof:**

Suppose there is an algorithm \( A \) that decides \( L \) in \( < n \) steps.

Let \( I \) be instance (input) \( a^k b^k \)

When \( A \) runs on input \( I \), there must be some index \( j \) such that \( A \) never reads \( I[j] \).

Let \( I' \) be the same as \( I \), but with j’th coordinate reversed. (\( I' \) is a NO instance)

When \( A \) runs on \( I' \), it has the same behavior as it does on \( I \)

But then \( A \) does not decide \( L \).  \textit{Contradiction}. 

Example

$$L = \{0^k 1^k : k \geq 0\}$$

**Fact:** Any algorithm that decides $L$ must use $\geq n$ steps.

This shows the intrinsic complexity of $L$ is $\Omega(n)$.

But we also know the intrinsic complexity of $L$ is $O(n)$.

The dream achieved. Intrinsic complexity is $\Theta(n)$. 
Representation of the input
How you represent the input matters

Technically, how the input is represented/encoded should be part of the problem description.

If it is not specified, input length is the number of bits needed to represent the input.

You should be careful about this!
How you represent the input matters

**Multiplication Problem**

**Input:** 2 numbers \( s \) and \( t \)

**Output:** the product of \( s \) and \( t \)

*Obvious algorithm:* Add \( s \) to itself \( t \) times.

How is the input represented?

\[
s = 111\ldots1 \quad (s \text{ many } 1s) \quad \quad \quad \quad n = s + t \\
t = 111\ldots1 \quad (t \text{ many } 1s)
\]

Running time: \( O(st) \quad O(n^2) \)
How you represent the input matters

**Multiplication Problem**

**Input:** 2 numbers $s$ and $t$

**Output:** the product of $s$ and $t$

**Obvious algorithm:** Add $s$ to itself $t$ times.

How is the input represented?

$s = \text{binary representation of } s$

t = \text{binary representation of } t$

We’ll do $s+s$ in binary, so it is $O(\log s)$ steps.

Running time: $O(t \log s)$, $O(2^n)$
How you represent the input matters

**Multiplication Problem**

**Input:** 2 numbers $s$ and $t$

**Output:** the product of $s$ and $t$

**Obvious algorithm:** Add $s$ to itself $t$ times.

This algorithm actually sucks!

When dealing with problems with integer inputs:

we want to be able to deal with numbers with say a million binary digits.

So numbers of magnitude $2^{10^6}$. 
How you represent the input matters

**Multiplication Problem**

**Input:** 2 numbers \( s \) and \( t \)

**Output:** the product of \( s \) and \( t \)

**Obvious algorithm:** Add \( s \) to itself \( t \) times.

Is there a more efficient algorithm?

\[
\begin{array}{c}
5 \ 6 \ 7 \ 8 \\
\times \ 1 \ 2 \ 3 \ 4 \\
\hline
2 \ 2 \ 7 \ 1 \ 2 \\
1 \ 7 \ 0 \ 3 \ 4 \\
1 \ 1 \ 3 \ 5 \ 6 \\
\hline
5 \ 6 \ 7 \ 8 \\
+ \ 5 \ 6 \ 7 \ 8 \\
\hline
7 \ 0 \ 0 \ 6 \ 6 \ 5 \ 2 \\
\end{array}
\]

\( O(n^2) \)

Can we do better?
Strong Church Turing Thesis
Church-Turing Thesis:
The intuitive notion of “computable” is captured by functions computable by a Turing Machine.

Physical Church-Turing Thesis:
Any computational problem that can be solved by a physical device, can be solved by a Turing Machine.

Strong Church-Turing Thesis:
The intuitive notion of “efficiently computable” is captured by functions efficiently computable by a TM.
Strong Church Turing Thesis

Experience suggests it is true for all deterministic models.

First main challenger in 1970s:

Randomized computation.

In light of research from 1980s, we believe SCCT holds even with randomized computation.

Second main challenger in 1980s:

Quantum computation.

In light of research from 1990s, we believe SCCT is not true!
Challenge all ideas!