Today's learning goals

• Distinguish between computability and complexity
• Articulate motivation questions of complexity

• Section 7.1: time complexity, asymptotic upper bounds.
• Section 7.2: polynomial time, P
• Section 7.3: NP, polynomial verifiers, nondeterministic machines.
In the "real world", computers (and Turing machines) don't have infinite tape, and we can't afford to wait unboundedly long for an answer.

"Decidable" isn't good enough – we want "Efficiently decidable"
Not just decidable …

- For a given algorithm working on a given input, how long do we need to wait for an answer? How does the running time depend on the input in the worst-case? average-case? Expect to have to spend more time on larger inputs.
Measuring time

- For a given algorithm working on a given input, how long do we need to wait for an answer? Count steps! How does the running time depend on the input in the worst-case? average-case? Big-O

Can we detect problems that are efficiently solvable?
Time complexity

For M a deterministic decider, its **running time** or **time complexity** is the function

\[ f : \mathbb{N} \rightarrow \mathbb{R}^+ \]

given by

\[ f(n) = \text{maximum number of steps } M \text{ takes before halting,} \]

over all inputs of length \( n \).

**Plus, instead of calculating precisely, estimate** \( f(n) \) **by using big-O notation.**
Time complexity classes

\[ \text{TIME}(t(n)) = \{ L \mid L \text{ is decidable by a TM running in } O(t(n)) \} \]

- **Exponential**
- **Polynomial**
  \[ P = \bigcup_{k} \text{TIME}(n^k) \]
- **Logarithmic**

  - Brute-force search
  - Invariant under many models of TMs
  - May not need to read all of input
Why is it okay to group all polynomial running times?

- Contains all the "feasibly solvable" problems.
- Invariant for all the "usual" deterministic TM models
  - multitape machines (Theorem 7.8)
  - multi-write
Which Turing machine model?

deterministic computation

\[
\begin{array}{c}
q_0 \\
\downarrow \\
q_{\text{rej}}
\end{array}
\]

non-deterministic computation

\[
\begin{array}{c}
q_0 \\
\quad \downarrow \\
q_{\text{rej}} \\
\quad \downarrow \\
q_{\text{acc}} \\
\quad \downarrow \\
q_{\text{acc}}
\end{array}
\]
Time complexity

For $M$ a deterministic decider, its **running time** or **time complexity** is the function $f: \mathbb{N} \rightarrow \mathbb{R}^+$ given by

$$f(n) = \text{maximum number of steps } M \text{ takes before halting, over all inputs of length } n.$$  

For $M$ a **nondeterministic decider**, its **running time** or **time complexity** is the function $f: \mathbb{N} \rightarrow \mathbb{R}^+$ given by

$$f(n) = \text{maximum number of steps } M \text{ takes before halting on any branch of its computation, over all inputs of length } n.$$
Time complexity classes

$\text{DTIME} ( \ t(n) \ ) = \{ L \mid L \text{ is decidable by } O( t(n) ) \}
\text{deterministic, single-tape TM }$

$\text{NTIME} ( \ t(n) \ ) = \{ L \mid L \text{ is decidable by } O( t(n) ) \}
\text{nondeterministic, single-tape TM }$

Is $\text{DTIME}(n^2)$ a subset of $\text{DTIME}(n^3)$?

A. Yes
B. No
C. Not enough information to decide
D. I don't know
Time complexity classes

$\text{DTIME}(t(n)) = \{ L \mid L \text{ is decidable by } O(t(n)) \}
\text{deterministic, single-tape TM}\}

$\text{NTIME}(t(n)) = \{ L \mid L \text{ is decidable by } O(t(n)) \}
\text{nondeterministic, single-tape TM}\}

Is $\text{DTIME}(n^2)$ a subset of $\text{NTIME}(n^2)$?

A. Yes
B. No
C. Not enough information to decide
D. I don't know
Time complexity classes

\[ \text{DTIME} \left( t(n) \right) = \{ L \mid L \text{ is decidable by } O(t(n)) \text{ deterministic, single-tape TM} \} \]

\[ \text{NTIME} \left( t(n) \right) = \{ L \mid L \text{ is decidable by } O(t(n)) \text{ nondeterministic, single-tape TM} \} \]

Is \( \text{NTIME}(n^2) \) a subset of \( \text{DTIME}(n^2) \)?

A. Yes
B. No
C. Not enough information to decide
D. I don't know
P vs. NP
"Feasible" i.e. $P$

- Can't use nondeterminism
- Can use multiple tapes

Often need to be "more clever" than naïve / brute force approach

Examples

$\text{PATH} = \{ <G,s,t> \mid G \text{ is digraph with } n \text{ nodes there is path from } s \text{ to } t \}$

Use breadth first search to show in $P$

$\text{RELPRIME} = \{ <x,y> \mid x \text{ and } y \text{ are relatively prime integers} \}$

Use Euclidean Algorithm to show in $P$

$L(G) = \{ w \mid w \text{ is generated by } G \}$

where $G$ is any CFG

Use Dynamic Programming to show in $P$
"Verifiable" i.e. NP

- Can be decided by a nondeterministic TM in polynomial time
- Best known deterministic solution is brute-force
- Solution can be verified by a deterministic TM in polynomial time

P = NP?
Examples in NP

Solution can be *verified* by a deterministic TM in polynomial time

\[
\text{HAMPATH} = \{ <G,s,t> \mid G \text{ is digraph with } n \text{ nodes there is path from } s \text{ to } t \text{ that goes through every node exactly once} \}\]

\[
\text{VERTEX-COVER} = \{ <G,k> \mid G \text{ is an undirected graph with } n \text{ nodes that has a } k \text{-node vertex cover} \}\]

\[
\text{CLIQUE} = \{ <G,k> \mid G \text{ is an undirected graph with } n \text{ nodes that has a } k \text{-clique} \}\]

\[
\text{SAT} = \{ <X> \mid X \text{ is a satisfiable Boolean formula with } n \text{ variables} \}\]
Examples in NP

Claim: HAMPATH is in NP
Decidable

P = NP

Finite

or

Decidable

NP

P

Finite
# P vs. NP

<table>
<thead>
<tr>
<th>Problems in P</th>
<th>Problems in NP</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Membership in any) CFL</td>
<td>Any problem in P</td>
</tr>
<tr>
<td>PATH</td>
<td>HAMPATH</td>
</tr>
<tr>
<td>$E_{DFA}$</td>
<td>CLIQUE</td>
</tr>
<tr>
<td>$EQ_{DFA}$</td>
<td>VERTEX-COVER</td>
</tr>
<tr>
<td>Addition, multiplication of integers</td>
<td>TSP</td>
</tr>
<tr>
<td>…</td>
<td>SAT</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
Next time

Pre-class reading skim Chapter 7