Today's learning goals

- Distinguish between polynomial and exponential DTIME
- Define nondeterministic running time
- Analyse an algorithm to determine whether it is in P
- Define the class NP
- Analyse a nondeterministic algorithm to determine whether it is in NP
- List some famous problems in P
- List some famous problems in NP
- State and explain $P=NP$?
- Define NP-completeness
- Explain the connection between $P=NP$? and NP-completeness
Announcements

• Final exam review
  • In class on Thursday
  • Evening session Thursday

• CAPE and TA evaluations open

• Final exam study guide + seat assignments on Ted
• Final Saturday June 4 11:30am-2:29pm PETERSON 108
  • 1 note sheet, may be 5x8, no magnifying glass!
Measuring time

- For a given **algorithm** working on a given **input**, how long do we need to wait for an answer? **Count steps!**

- For a **problem** decided by a given algorithm, how does the running time depend on the input in the worst-case? **average-case?** **Big-O**

- What's in common among all problems that are **efficiently solvable**? **Time(t(n))**
Time complexity classes

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

- **Exponential**
  \[ \text{EXPTIME} = \bigcup_{k} \text{TIME}(2^{n^k}) \]

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

- **Logarithmic**
  May not need to read all of input

Brute-force search

Invariant under many models of TMs
Which machine model?

deterministic computation

q0 → q_rej

non-deterministic computation

q0 → q_rej

q_rej → q_acc

q_rej → q_acc
Time complexity

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

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

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

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

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

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

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) \text{ a subset of } \text{DTIME}(n^2)\text{?}
A. Yes
B. No
C. Not enough information to decide
D. I don't know
"Feasible" i.e. P

\[ P = \bigcup \limits_{k} \text{TIME}(n^k) \]

- Can't use nondeterminism
- Can use multiple tapes

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

**Examples**

- PATH = \{<G,s,t> | G is digraph with n nodes there is path from s to t\}
- RELPRIME = \{ <x,y> | x and y are relatively prime integers\}

  Use Euclidean Algorithm to show in P

- L(G) = \{w | w is generated by G\} where G is any CFG

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

- Best known solution is brute-force
- Look for some "certificate" – if had one, could check if it works quickly

\[ NP = \bigcup_{k} NTIME(n^k) \]
Examples in NP for graphs

HAMPATH = \{ <G,s,t> | G is digraph with a path from s to t that goes through every node exactly once \}

CLIQUE = \{ <G,k> | G is an undirected graph with a k-clique \}

Complete subgraph with k nodes

VERTEX-COVER = \{ <G,k> | G is an undirected graph with a k-node vertex cover \}

Subset of k nodes s.t. each edge incident with one of them
Examples in NP for graphs

CLIQUE = \{ <G,k> | G is an undirected graph with a k-clique \}

Complete subgraph with k nodes

How many possible k-cliques are there? How long does it take to confirm "clique-ness"?

A. \(O(n^n), O(n^2)\)
B. \(O(2^n), O(n)\)
C. \(O(n!), O(\log n)\)
D. \(O(n), O(n)\)
E. I don't know
TSP = \{ <G,k> \mid G \text{ is complete weighted undirected graph where weight between node } i \text{ and node } j \text{ is "distance" between them; there is a tour of all cities with total distance less than } k \}\n
How many possible tours are there? How long does it take to check the distance of a single tour?

A. $O(n^2), O(n)$
B. $O(2^n), O(n^2)$
C. $O(n^n), O(\log n)$
D. $O(n!), O(n)$
E. I don't know
Examples in NP for numbers  

COMPOSITES = \{ x \mid x \text{ is an integer } > 2 \text{ and is not prime} \}

SUBSET-SUM = \{ <S,t> \mid S=\{x_1,\ldots,x_k\} \text{ and some subset sums to } t \}
Examples in NP for logic

SAT = \{ \langle \varphi \rangle \mid \varphi \text{ is a satisfiable Boolean formula} \}

Is \( \langle (x \lor \bar{y}) \lor (\bar{x} \land y) \rangle \) in SAT?
A. Yes
B. No
C. Not enough information to decide
D. I don't know
# 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>
Decidable

NP?

P

CF

Regular
Reductions to the rescue

1970s Stephen Cook and Leonid Levin independently and in parallel lay foundations of NP-completeness

Intuitively: if an NP-complete problem has a polynomial algorithm, then all NP problems are polynomial time solvable.

A language B is NP-complete if it is in NP and every A in NP is polynomial-time reducible to it.

Cook-Levin Theorem: SAT is NP-complete.
Reductions to the rescue

1970s Stephen Cook and Leonid Levin independently and in parallel lay foundations of NP-completeness.

Intuitively: if an NP-complete problem has a polynomial algorithm, then all NP problems are polynomial time solvable.

A language B is **NP-complete** if it is in NP and every A in NP is polynomial-time reducible to it.

**Cook-Levin Theorem**: SAT is NP-complete.

What would prove that P = NP?

A. Showing that a problem solvable by brute-force methods has a nondeterministic solution.
B. Showing that there are two distinct NP-complete problems.
C. Finding a polynomial time solution for an NP-complete problem.
D. Proving that an NP-complete problem is not solvable in polynomial time.
E. I don't know