Today's learning goals

- Define NP-completeness
- Give examples of NP-complete problems
- Use polynomial-time reduction to prove NP-completeness

- Section 7.4, 7.5: NP-completeness

Start review!
Decidable

P = NP

Finite

or

Decidable

NP

P

Finite
<table>
<thead>
<tr>
<th>Problems in P</th>
<th>Problems in NP</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Membership in any) regular language</td>
<td>Any problem in P</td>
</tr>
<tr>
<td>(Membership in any) CFL</td>
<td>HAMPATH</td>
</tr>
<tr>
<td>PATH</td>
<td>CLIQUE</td>
</tr>
<tr>
<td>$A_{DFA}$</td>
<td>VERTEX-COVER</td>
</tr>
<tr>
<td>$E_{DFA}$</td>
<td>TSP</td>
</tr>
<tr>
<td>$EQ_{DFA}$</td>
<td>SAT</td>
</tr>
<tr>
<td>Addition, multiplication of integers</td>
<td>...</td>
</tr>
</tbody>
</table>
How to answer $P = NP$?

Are there hardest NP problems?

If so, finding an efficient solution for one of them would guarantee that all NP problems have efficient solutions.
Reductions to the rescue

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

**Definition** A language B is **NP-complete** if (1) it is in NP and (2) every A in NP is polynomial-time reducible to it.

Problem A polynomial-time reduces to problem B means there is a polynomial-time computation function f such that, for all x, x is in A iff f(x) is in B.
Reductions to the rescue

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

**Definition** A language B is **NP-complete** if (1) it is in NP and (2) every A in NP is polynomial-time reducible to it.

**Consequence** If an NP-complete problem has a polynomial time solution then all NP problems are polynomial time solvable.
Reductions to the rescue

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

Definition A language B is NP-complete if (1) it is in NP and (2) every A in NP is polynomial-time reducible to it.

Consequence If an NP-complete problem has a polynomial time solution then all NP problems are polynomial time solvable.

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
3-SAT

Cook-Levin Theorem: 3-SAT is NP-complete.

\[(x \lor \overline{y} \lor \overline{z}) \land (\overline{x} \lor y \lor z) \land (x \lor y \lor z)\]
Are other problems NP complete?

To prove that X is NP-complete

*From scratch*: Prove it is NP, and that all NP problems are polynomial-time reducible to it.

*Using reduction*: Show that a (known-to-be) NP complete problem reduces to it.
3SAT polynomial-time reduces to CLIQUE

Given: Boolean formula in CNF with exactly 3 literals/clause
- AND of ORs
- args in OR clauses: var or negated var

Desired Answer: Yes if satisfiable; No if unsatisfiable

Instead transform formula to graph so that graph has clique iff original formula is satisfiable
3SAT polynomial-time reduces to CLIQUE

Transform 3-CNF formula with $k$ clauses to graph $G$

- vertices are the literals in each clause
- edges between all vertices except
  - two literals in the same clause
  - literals that are negations of one another

Claim: formula is satisfiable iff $G$ has $k$-clique
3-SAT to Clique example

$$(x \lor \bar{y} \lor \bar{z}) \land (\bar{x} \lor y \lor z) \land (x \lor y \lor z)$$
Are other problems NP-complete?
Next time

Review for final exam

Please fill out CAPE, TA evaluations.

Seat charts for final exam on Piazza.