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!

Discussion today (mattw8)
Review Session today (PracticeQs)
Office hours
Friday's class
Decidable

P = NP

Finite

or

Finite

Decidable

NP

P

Finite
### P vs. NP

**Problems in P**
- (Membership in any) regular language
- (Membership in any) CFL
- PATH
- \( A_{DFA} \)
- \( E_{DFA} \)
- \( E_{Q_{DFA}} \)
- Addition, multiplication of integers

**Problems in NP**
- Any problem in P
- HAMPATH
- CLIQUE
- VERTEX-COVER
- TSP
- SAT
- ...
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!
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 computable function f such that, for all x, x is in A iff f(x) is in B.
Suppose $F: \Sigma^* \rightarrow \Sigma^*$

polynomial time computable?

$x \in A$ if $F(x) \in B$.

$x \in A$? Suppose we have $M$, a decider for $B$,

"On input $x$.

1. Compute $F(x)$

2. Run $M$ on $F(x)$. $x$ takes polynomial time steps.

3. Accept if $M$ accepts,

4. Reject if $M$ rejects."

A $\in$ P
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.
Reducions 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.

\[(x \lor \overline{y} \lor \overline{z}) \land (\overline{x} \lor y \lor z) \land (x \lor y \lor z)\]

Analogy: undecidable
First diagonalization
New: reduction.
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  

Sipser p. 302

**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

CLIQUE NP-complete
3-SAT to Clique example

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

Review for final exam

Please fill out CAPE, TA evaluations.

Seat charts for final exam on Piazza.