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

CP

P

Refl

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) 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\textsubscript{DFA}</td>
<td>VERTEX-COVER</td>
</tr>
<tr>
<td>E\textsubscript{DFA}</td>
<td>TSP</td>
</tr>
<tr>
<td>EQ\textsubscript{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?
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.
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

[Diagram showing reductions to an NP-complete problem]
3-SAT = \begin{cases} \emptyset & \text{if } \varphi \text{ is in CNF, } 3 \text{ literals/clause} \\ \text{true} & \text{if } \varphi \text{ is satisfiable} \end{cases}

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

\varphi = \left( x \lor \overline{y} \lor z \right) \land \left( \overline{x} \lor y \lor z \right) \land \left( x \lor y \lor z \right)

\begin{align*}
x &= T \\
y &= F \\
z &= F
\end{align*}

ex. doesn't mean \( \varphi \) is unsatisfiable
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

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

Truth assignment:
1. \(x\)
2. \(y\)
3. \(z\)
4. \(\overline{x}\)
5. \(y\)
6. \(\overline{z}\)
7. \(z\)
8. \(\overline{x}\)

3-clique:
keep going...
Are other problems NP-complete?
Next time

Review for final exam

Please fill out CAPE, TA evaluations.

Seat charts for final exam on Piazza.
Lecture B: PCYNH109