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
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>$A_{DFA}$ regular languages</td>
<td>SAT</td>
</tr>
<tr>
<td>context-free languages</td>
<td>NP-complete.</td>
</tr>
</tbody>
</table>

A path through each vertex exactly once.
How to answer $P = NP$?

Suppose $G \in NP$ is a problem such that for all $X \in NP$ in polynomial time, $X$ reduces to $G$.

This property is called $NP$-completeness.
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.

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

Cook-Levin Theorem: 3-SAT is NP-complete
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
3SAT reduces to Clique example

- \((x \lor \overline{y} \lor \overline{z}) \land (x \lor y \lor z) \land (x \lor y \lor z)\)

3 variables. \(2^3 = 8\) possible assignments.

Satisfies:
- \(x = \text{True}\)
- \(y = \text{True}\)
- \(z = \text{True}\)

Satisfies:
- \(x = \text{False}\)
- \(y = \text{False}\)
- \(z = \text{False}\)
3SAT reduces to Clique example

- \((x \lor \overline{y} \lor z) \land (\overline{x} \lor y \lor z) \land (x \lor y \lor z)\)

clauses:

- \(x, \overline{z}, y\)

\(x = \text{false}\)
\(z = \text{false}\)
\(y = \text{true}\)

\(\Rightarrow\) CLIQUE
\(\in\) NP-complete.
The famous logician Kurt Gödel asked the famous computer scientist, mathematician, and economist John von Neumann the P vs. NP question in a private letter, written shortly before von Neumann’s death.
in the soviet union

• S.V. Yablonkski invents the term
• “perebor” or “brute force search”
to describe the combinatorial
explosion limiting algorithms, especially
for circuit design problems (1959)
In 1965, Jack Edmonds gives the first polynomial time algorithm for perfect matching on general graphs. To explain the significance to referees, he introduces a section defining P, NP and posing the P vs. NP question.
NP-completeness

- In 1971, Steve Cook defines NP-completeness and proves that several problems from logic and combinatorics are NP-complete, meaning that P=NP if and only if any of these problems are polynomial time solvable.
Following Cook’s work, Richard Karp showed that a large number of the most important optimization problems from all sub-areas (scheduling, graph theory, number theory, logic, puzzles and games, packing, coding, ...) are NP-complete.
Back in the USSR

- Leonid Levin, a student of Kolmogorov’s, publishes similar results to
- Cook and Karp’s in his thesis,
- but needs to be careful
- to disguise what he’s claiming, since it might be interpreted as
Questioning earlier work on perebor.
Garey and Johnson’s classic Textbook (1979) includes an Appendix listing hundreds of NP-complete problems.
NP-complete problems everywhere

Since then, thousands of NP-complete problems have been identified in pretty much any area. With computational problems in physics, biology, chemistry, economics, sociology, linguistics, games, engineering, ….
Some of our favorites

- Sudoku
Tetris
Minesweeper
Next time

Pre-class reading skim Chapter 7