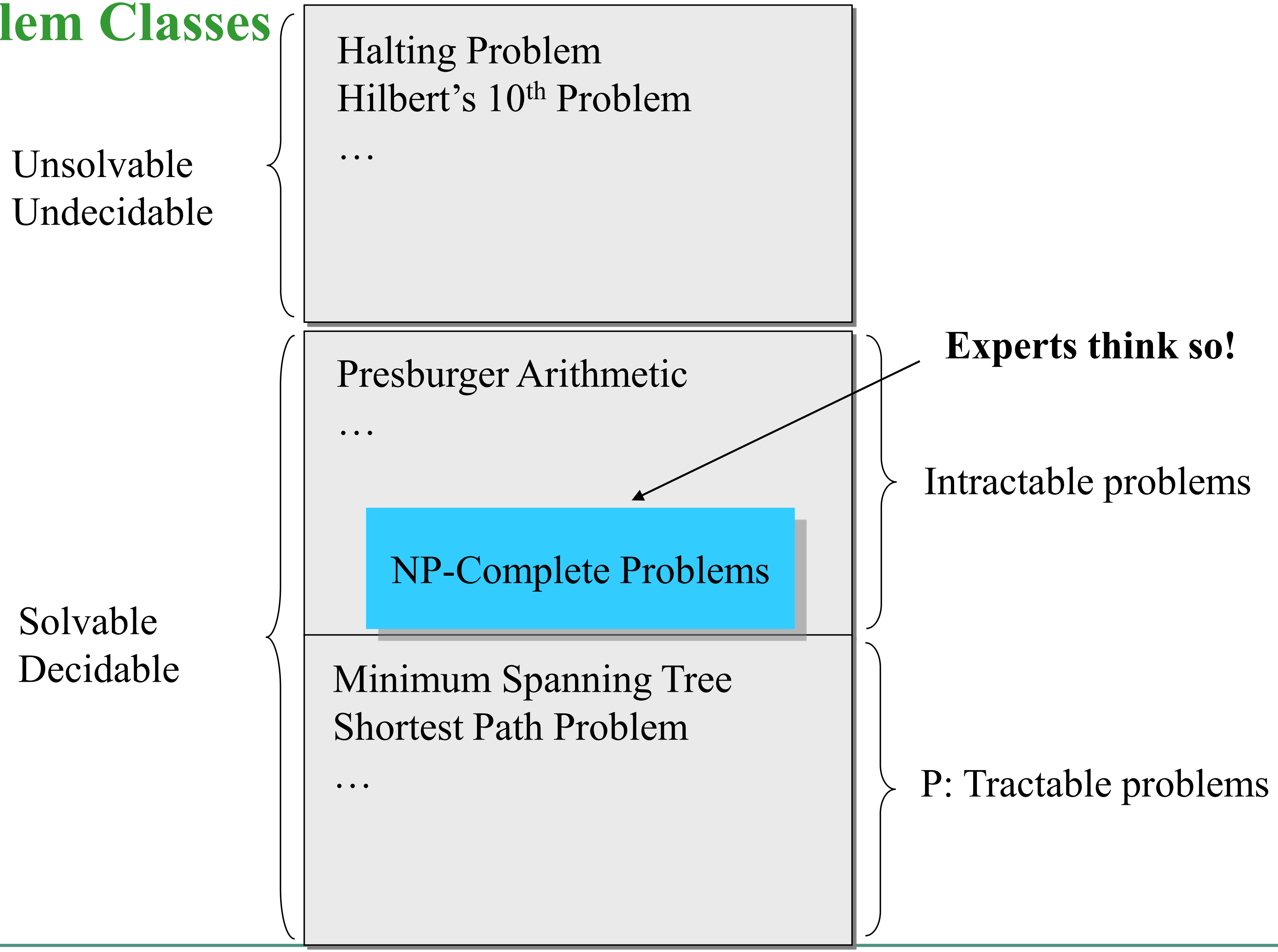
13. NP-Completeness

Goals

- Learn P and NP.
- Understand decision problems and optimization problems.
- Learn the definition of NP-completeness.
- Learn how to prove that a problem is NP-complete.

Problem Classes



Tractable

- Polynomial time
  - Time complexity: a polynomial in terms of input size *n*
  - Example: $3n^{k} + 5n^{k-1} + \ldots$
  - Class P: set of problems that are solvable in polynomial time
- Intractable
  - Exponential time
    - 2^{n}, n!
  - Polynomial space, doubly exponential

Decision Problem

- Decision problem (Yes/No problem)
  - Example: Is there a hamiltonian path of length at most k in graph G?
- Optimization problem
  - Example: What is the length of the shortest hamiltonian path in graph G?

✓ We can solve an optimization problem by solving the corresponding decision problem

Theory of NP-Completeness

- Focus on decision problems
  - But implications can be extended to optimization problems.
- Theory on the border between tractable and intractable
- Class NP-complete: a huge number of problems
  - All NP-complete problems are related: If one of these problems can be solved in polynomial time, then all problems in class NP-complete are solved in polynomial time.

Research So Far

- If a problem is proved to be NP-complete,

⇨ there is no known way of solving it in polynomial time.

- However, whether the problem can be solved in polynomial time or not (P=NP problem) is not solved yet.
- The P=NP problem is one of seven Millennium Prize Problems established by Clay Mathematics Institute.

NP-Completeness

Your boss asked you to find an efficient algorithm for
an NP-complete problem.

I can’t find an efficient algorithm. I guess I’m just too dumb.

(b)

I can’t find an efficient algorithm, because no such algorithm is possible.

(c)

I can’t find an efficient algorithm, but neither can all these famous people.

Polynomial-Time Reduction

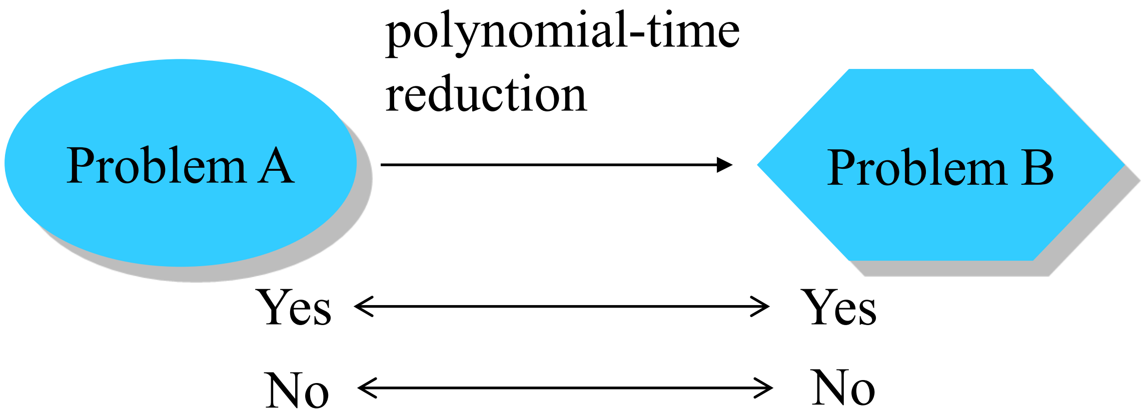
Exercise

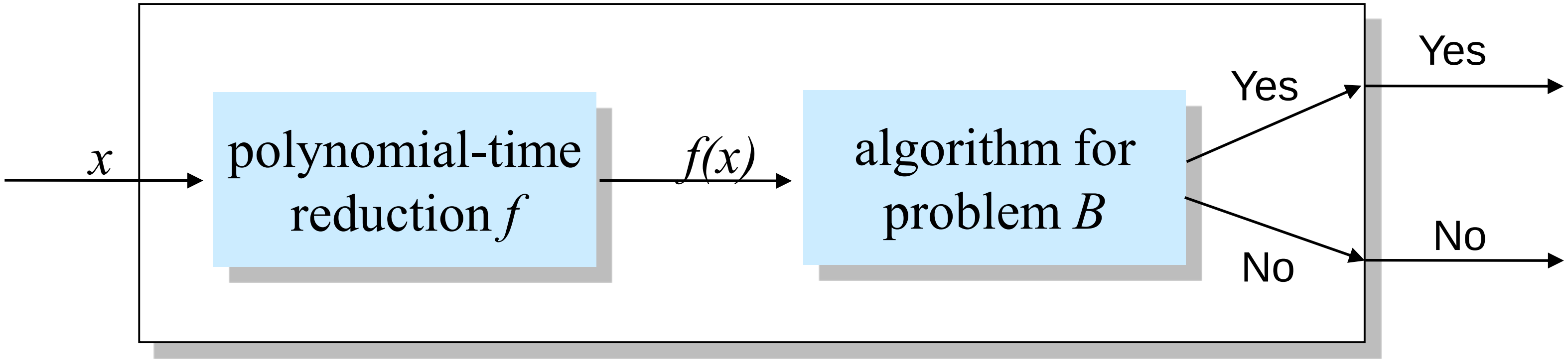
Problem 1: Is integer $x = x_1 x_2 \ldots x_n$ a multiple of 3?

Problem 2: Is $x_1 + x_2 + \ldots + x_n$ a multiple of 3?

- ✓ The answers to the two problems are the same.
  - ➢ Yes/No answers are the same.
- ✓ If problem 2 is an easy problem, the problem 1 is also easy.

- Polynomial-time reduction
  - a polynomial-time algorithm *f* that converts an instance *x* of problem *A* to an instance *f(x)* of problem *B* such that the (yes/no) answer to *x* is the same as the answer to *f(x)*.
  - denoted by $A \leq_P B$





algorithm for problem *A*

1. Convert an instance *x* of problem *A* to an instance *f(x)* of problem *B* in polynomial time.
2. Run an algorithm for problem *B* on instance *f(x)*.
3. Return the answer for *f(x)* as the answer for *x*.

– If *B* is solved in polynomial time, *A* can be solved in polynomial time.

– If *A* cannot be solved in polynomial time, *B* cannot be solved in polynomial time.

P and NP

- Complexity class P
  - Polynomial
  - Set of decision problems that can be solved in polynomial time
- Complexity class NP
  - Nondeterministic Polynomial
  - Set of decision problems that can be solved by nondeterministic Turing machine in polynomial time
  - Set of decision problems that can be verified in polynomial
- Proving that a problem belongs to NP is easy in most cases.
  - Hamiltonian cycle problem
  - Traveling salesman problem

NP-Complete/NP-Hard

- Problem L is NP-hard if
  - $A \leq_P L$ for every A in NP (every problem in NP is polynomial-time reducible to L)
- Problem L is NP-complete if
  1) L is in NP
  2) L is NP-hard

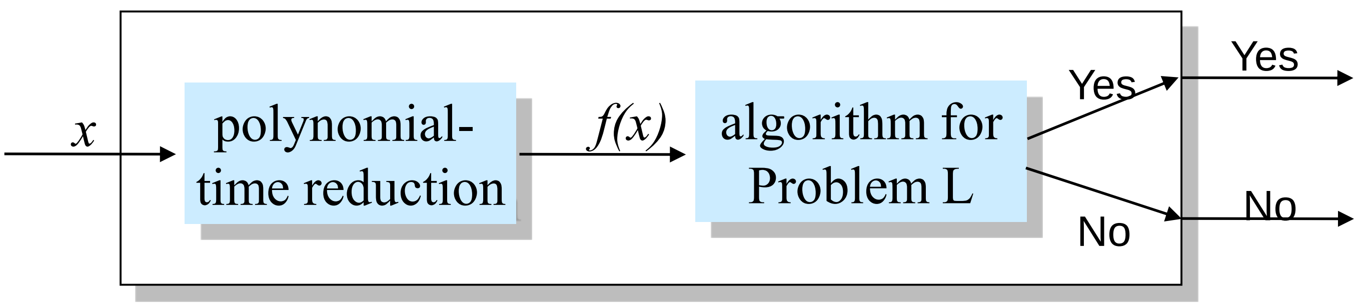
✓ Since condition 1 is easy in most cases, we focus on condition 2 in NP-completeness proof.

NP-Complete Problems

- First NP-complete problem
  - SAT (Boolean formula satisfiability problem)
  - Very difficult to prove
  - Done by Stephen Cook in 1971
- Other NP-complete problems
  - Proved by polynomial-time reduction

Theorem 1

- Problem L is NP-hard if it satisfies the following:
  - A known NP-hard problem A is polynomial-time reducible to L.



algorithm for problem A

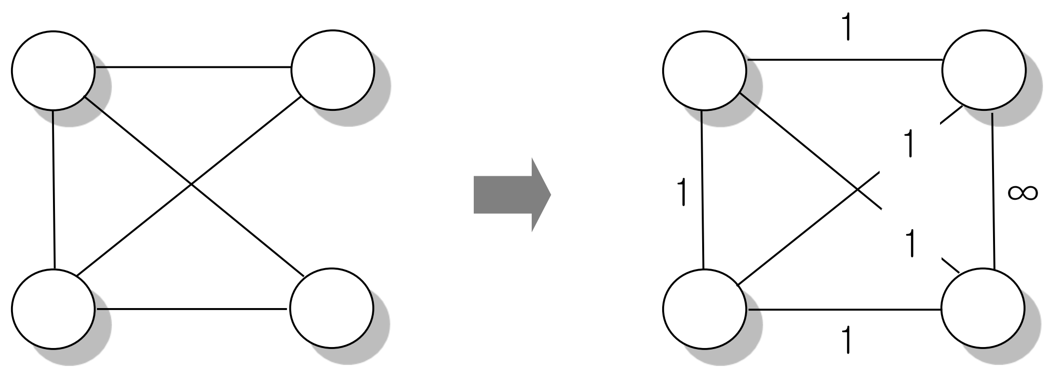
✓ For every problem B in NP, $B \leq_P A$. Since $A \leq_P L$, we have $B \leq_P L$.

NP-Completeness Proof

- Under the assumption that Hamiltonian cycle problem is NP-hard, prove that TSP is NP-hard.

- Hamiltonian cycle
  - simple cycle that contains each vertex in V
- Hamiltonian cycle problem
  - Given an undirected graph G, does G have a hamiltonian cycle?
- Traveling salesman problem (TSP)
  - Given a complete graph G with weights and K, does G have a hamiltonian cycle with weight $\leq K$?

- Polynomial-time reduction algorithm takes as input an instance x of Hamiltonian cycle problem, and outputs instance $f(x)$ of TSP.



instance of Hamiltonian cycle problem

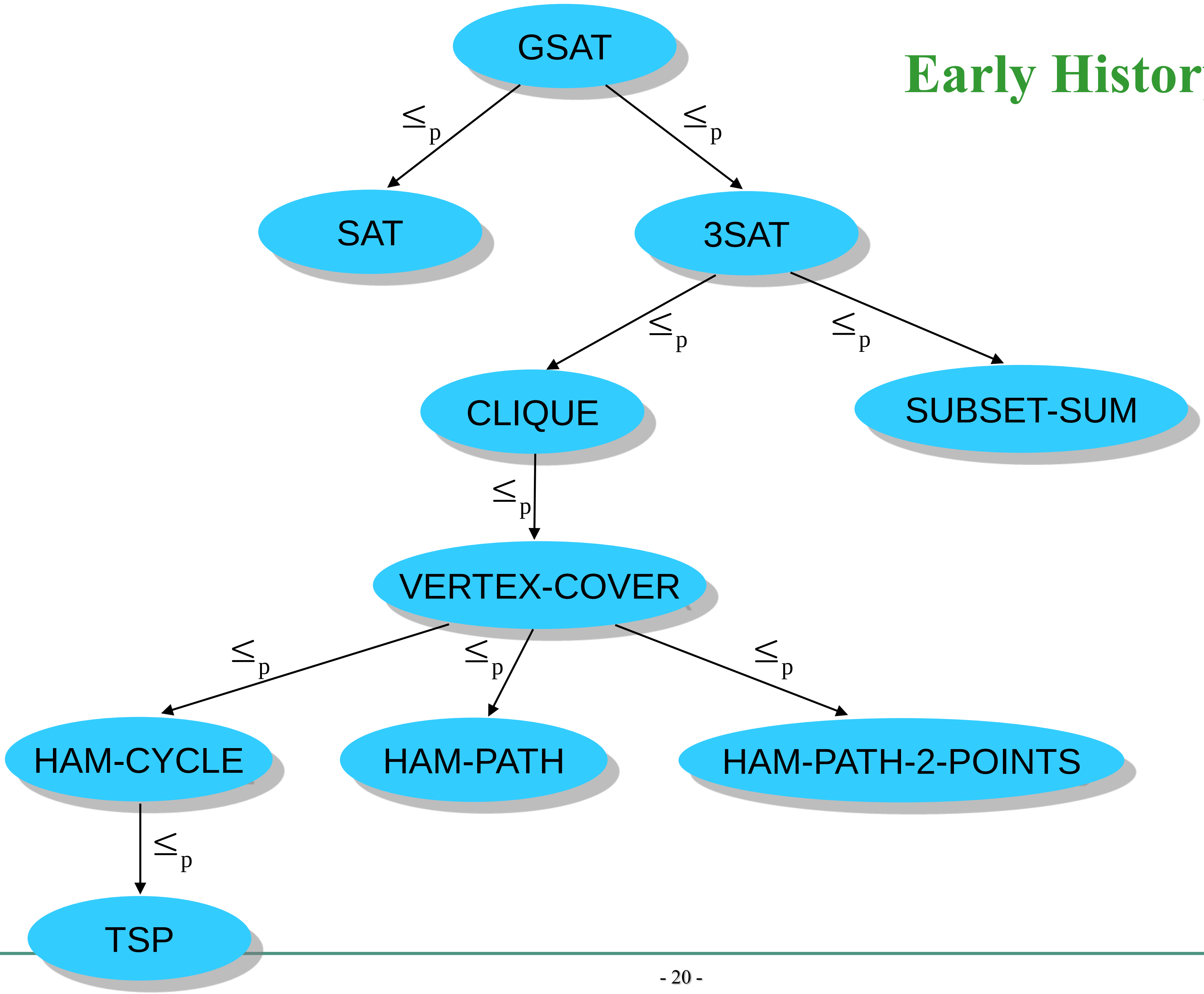
instance of TSP ($K = n$)

Instance x has a hamiltonian cycle

⇔ Instance $f(x)$ has a hamiltonian cycle with weight $\leq n$

➢ Therefore, TSP is NP-hard.

Early History



Satisfiability

- General Boolean formula (GSAT)
  - $\left((x_1 \wedge \overline{x_2}) \vee (x_3 \wedge x_4)\right) \wedge \overline{x_5}$
- Conjunctive normal form (SAT)
  - $(x_1 \vee \overline{x_2}) \wedge (x_2 \vee x_3 \vee x_4)$
- 3-CNF
  - $(x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee \overline{x_3} \vee x_4)$

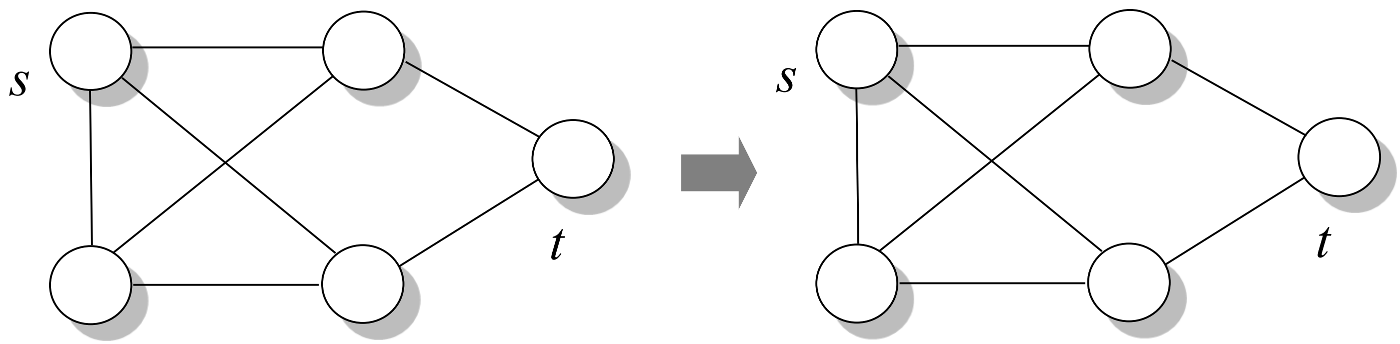
NP-hardness contrary to intuition

- Unweighted (directed or undirected) graph G
- Shortest path in G
  - Given two vertices s and t, find a shortest path from s to vertex t.
  - Belongs to P
- Longest simple path in G
  - Given two vertices s and t, find a longest simple path from s to t.
  - NP-hard

- Longest path problem (LONGEST-PATH)
  - Given graph G, vertices s, t, and integer k, is there a simple path from s to t in G of length $\geq k$?

- Hamiltonian path problem (HAMILTONIAN-PATH)
  - Given graph G and vertices s, t, is there a hamiltonian path from s to t in G?
  - NP-complete

- Polynomial-time reduction algorithm takes as input an instance *x* of Hamiltonian path problem, and outputs instance *f(x)* of Longest path problem.



instance of HAMILTONIAN-PATH instance of LONGEST-PATH ($k = n$-1)

Instance *x* has a hamiltonian path from *s* to *t*

⇔ Instance *f(x)* has a simple path from *s* to *t* of length $\geq k$

➢ Therefore, LONGEST-PATH is NP-Hard.

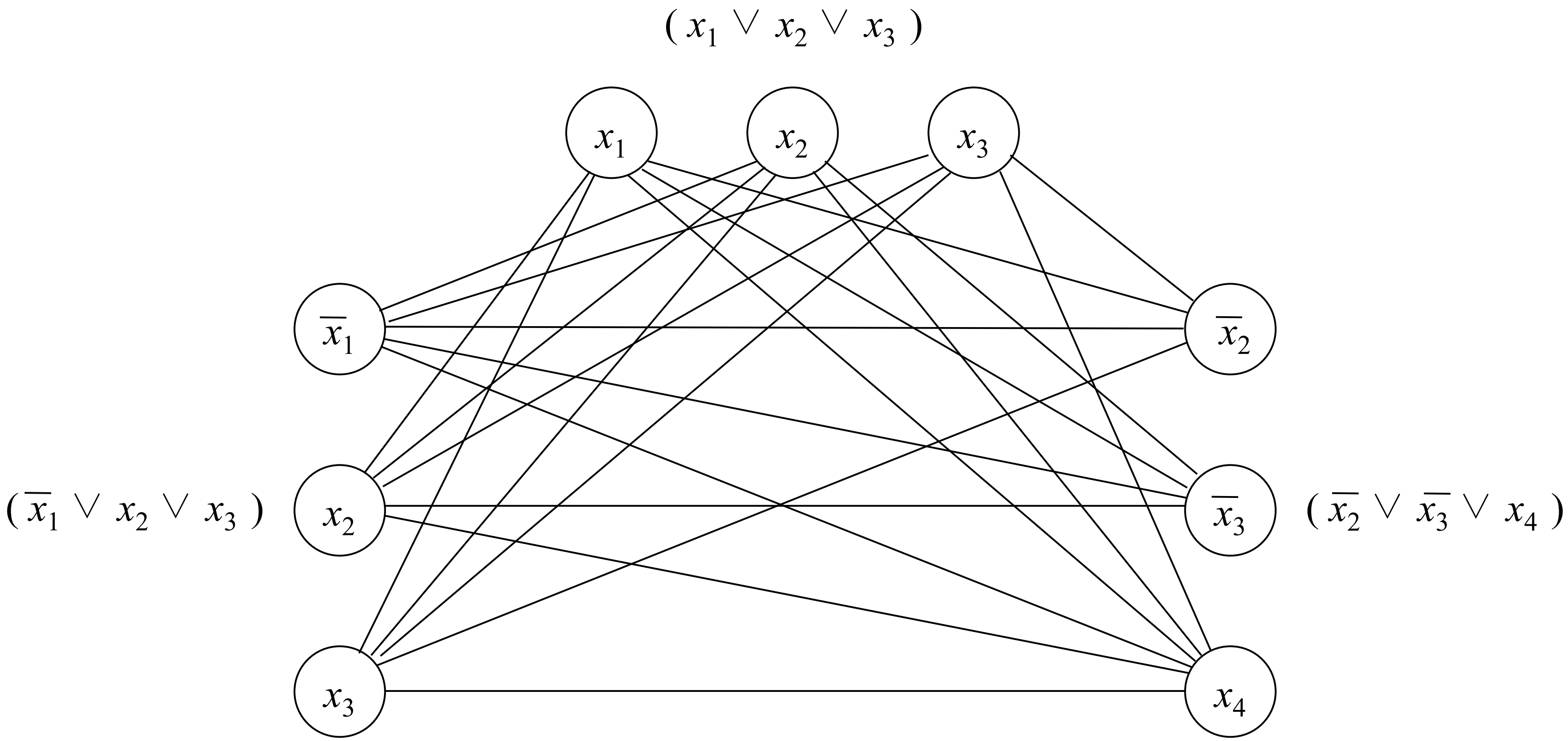
CLIQUE (Complete Subgraph)

- Input
  - Graph G = (V, E), integer k
- Problem
  - Is there a clique (complete subgraph) in G of size $\geq k$?

- CLIQUE is NP-complete.

3SAT $\leq_p$ CLIQUE

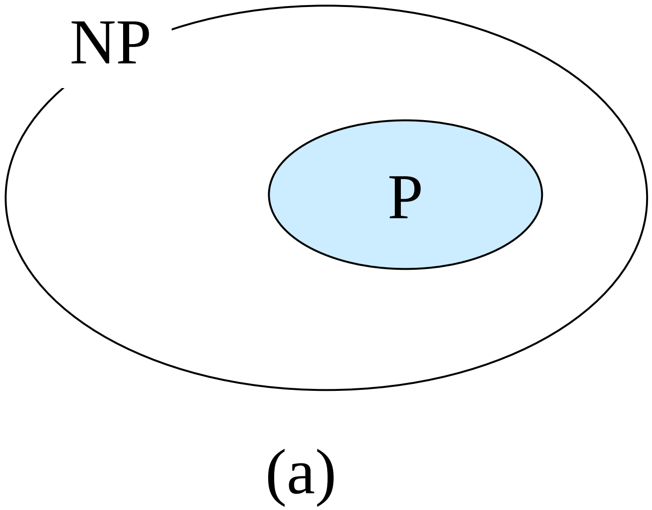
Instance of 3SAT: $(x_1 \lor x_2 \lor x_3) \land (\overline{x_1} \lor x_2 \lor x_3) \land (\overline{x_2} \lor \overline{x_3} \lor x_4)$

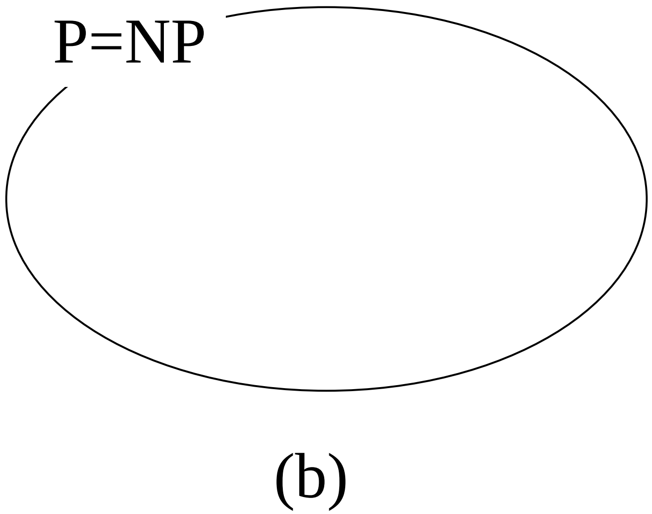


Edge (u,v) if u and v are in different clauses and u is not a negation of v. k=#clauses

$x_2 = \overline{x_1} = x_4 = 1 \quad \Leftrightarrow \quad$ clique of size 3

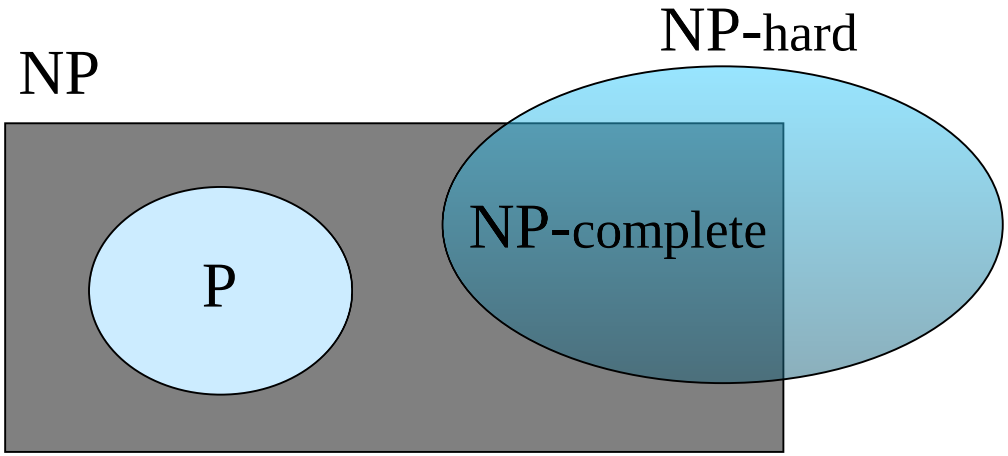
Relationship between P and NP





✓ It is an open problem whether (a) or (b) holds

NP, NP-Complete, NP-Hard



✓ If $P \neq NP$

Usefulness of NP-completeness Theory

- If a problem is proved to be NP-complete/NP-hard,
  - ⇨ stop efforts to find polynomial-time algorithms
  - ⇨ focus on finding algorithms (heuristics) appropriate for the application

- Solve a restricted problem (poly time)
- Find approximation algorithms (poly time)
- Branch-and-bound (state space search)

Thank you