Chapter 34: NP-Completeness

About this Tutorial

- What is NP ?
  - How to check if a problem is in NP ?
- Cook-Levin Theorem
  - Showing one of the most difficult problems in NP
- Problem Reduction
  - Finding other most difficult problems

Polynomial time algorithm

- **Polynomial time algorithms:** inputs of size n, worst-case running time is $O(n^k)$.
- **Exponential time:** $O(2^n)$, $O(3^n)$, $O(n!)$, ...
- It is natural to wonder whether all problems can be solved in polynomial time.
- The answer is no. For example, the **"Halting Problem" cannot be solved by any computer no matter how long we allow it**.

Pseudo-Polynomial time

- For example: Factoring

```
function isPrime(n):
    for i from 2 to √n
        if (n mod i) = 0 return false
    return true
```

Problem size k = lg n, Execution time = $O(\sqrt{n})$ = $O(2^{k/2})$

Tractable vs. Intractable

- Shortest vs. Longest simple paths
- Euler tour vs. Hamiltonian cycle

➢ An Euler tour of a connected, directed graph $G = (V, E)$ is a cycle that traverses each edge of G exactly once. Time complexity = $O(E)$

➢ A Hamiltonian cycle of a directed graph $G = (V, E)$ is a simple cycle that contains each vertex in V.

Tractable vs. Intractable

- k-CNF (Conjunctive Normal Form): the AND of clauses of ORs of k variables or their negations. A Boolean formula is **satisfiable** if some assignment of the values 0 and 1 to its variable causes it to be 1.
  - ✓For example: a 2-CNF satisfiability problem: $(a \vee \sim b) \wedge (\sim a \vee c) \wedge (\sim b \vee \sim a)$

    Ans: a = 1, b = 0, c = 1
- 2-CNF satisfiability vs. 3-CNF satisfiability

Decision Problems

- When we receive a problem, the first thing concern is: whether the problem has a solution or not
- E.g., Peter gives us a map $G = (V,E)$, and he asks us if there is a path from A to B whose length is at most 100
- E.g., Your sister gives you a number, say 1111111111111111111 (19 one's), and asks you if this number is a prime

Decision Problems

- The problems in the previous page is called decision problems, because the answer is either YES or NO
- Some decision problems can be solved efficiently, using time polynomial to the size of the input (what is input size?)
- The input size can be measured in number of bits.
- We use P to denote the set of all these polynomial-time solvable problems

Decision Problems

e.g. For Peter's problem, there is an $O(V \log V + E)$-time algorithm that finds the shortest path from A to B;

→ we can first apply this algorithm and then give the correct answer

→ Peter's problem is in P

- Can you think of other problems in P?
- Can you think of other problems not in P?

Polynomial-Time Verifiable

- Another interesting classification of decision problems is to see if the problem can be verified in time polynomial to the size of the input
- Precisely, for such a decision problem, whenever it has an answer YES, we can :

1. Ask for a short proof, and

   /* short means : polynomial in size of input */

2. Be able to verify the answer is YES

Polynomial-Time Verifiable

e.g., In Peter's problem, if there is a path from A to B with length ≤ 100, we can :

1. Ask for the sequence of vertices (with no repetition) in any path from A to B whose length ≤ 100
2. Check if it is a desired path (in poly-time)

→ this problem is polynomial-time verifiable

Polynomial-Time Verifiable

More examples:

Given a graph $G = (V,E)$, does the graph contain a Hamiltonian path ?

Is a given integer x a composite number ?

Given a set of numbers, can be divide them into two groups such that their sum are the same ?

Polynomial-Time Verifiable

- Now, imagine that we have a super-smart computer, such that for each decision problem given to it, it has the ability to guess a short proof (if there is one)
- With the help of this powerful computer, all polynomial-time verifiable problems can be solved in polynomial time (how ?)

The Class P and NP

- NP denote the set of polynomial-time verifiable problems
  - N stands for non-deterministic guessing power of our computer
  - P stands for polynomial-time "verifiable"
- NP: set of problems can be solved in polynomial time with non-deterministic Turing machine
- P: denote the set of problems that are polynomial-time solvable

P and NP

- We can show that a problem is in P implies that it is in NP (why?)
  - Because if a problem is in P, and if its answer is YES, then there must be an algorithm that runs in polynomial-time to conclude YES …
  - Then, the execution steps of this algorithm can be used as a "short" proof

P and NP

- On the other hand, after many people's efforts, some problems in NP (e.g., finding a Hamiltonian path) do not have a polynomial-time algorithm yet …
- Question: Does that mean these problems are not in P ??
- The question whether P = NP is still open

Clay Mathematics Institute (CMI) offers US$ 1 million for anyone who can answer seven problems, including if NP = P?

P and NP

- So, the current status is :
  1. If a problem is in P, then it is in NP
  2. If a problem is in NP, it may be in P
- In the early 1970s, Stephen Cook and Leonid Levin (separately) discovered that: a problem in NP, called SAT, is very mysterious …

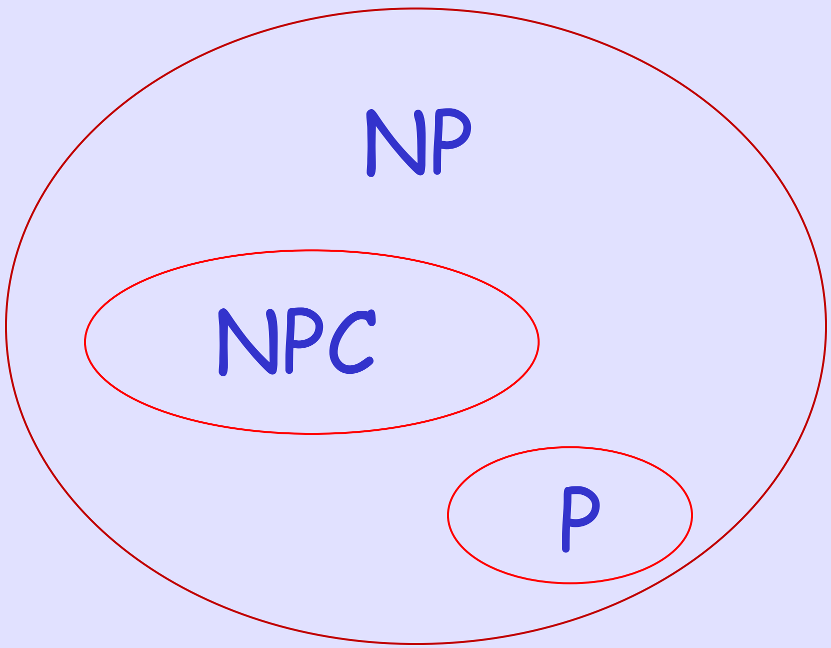
Cook-Levin Theorem

- If SAT is in P, then all problems in NP are also in P
  - i.e., if SAT is in P, then P = NP

  // Can Cook or Levin claim the money from CMI yet ?

- Intuitively, SAT must be one of the most difficult problems in NP
  - We call SAT an NP-complete problem (most difficult in NP)

All NP problems



NP = P ?

Satisfiable Problem

- The SAT problem asks:
  - Given a Boolean formula F, such as

    $$F = (x \vee y \vee \neg z) \wedge (\neg y \vee z) \wedge (\neg x)$$

    is it possible to assign True/False to each variable, such that the overall value of F is true ?

Remark: If the answer is YES, F is a satisfiable, and so it is how the name SAT is from

Other NP-Complete Problems

- The proofs made by Cook and Levin is a bit complicated, because intuitively they need to show that no problems in NP can be more difficult than SAT
- However, since Cook and Levin, many people show that many other problems in NP are shown to be NP-complete
  - How come many people can think of complicated proofs suddenly ??

Problem Reduction

- How these new problems are shown to be NP-complete rely on a new technique, called reduction (problem transformation)
- Basic Idea:
  - Suppose we have two problems, A and B
  - We know that A is very difficult
  - However, we know if we can solve B, then we can solve A
  - What can we conclude ??

Problem Reduction

- e.g., A = Finding median, B = Sorting
- We can solve A if we know how to solve B

  ➔ sorting is as hard as finding median

- eg., A = Topological Sort, B = DFS
- We can solve A if we know how to solve B

  ➔ DFS is as hard as topological sort

Problem Reduction

- Now, consider

  A = an NP-complete problem (e.g., SAT)

  B = another problem in NP

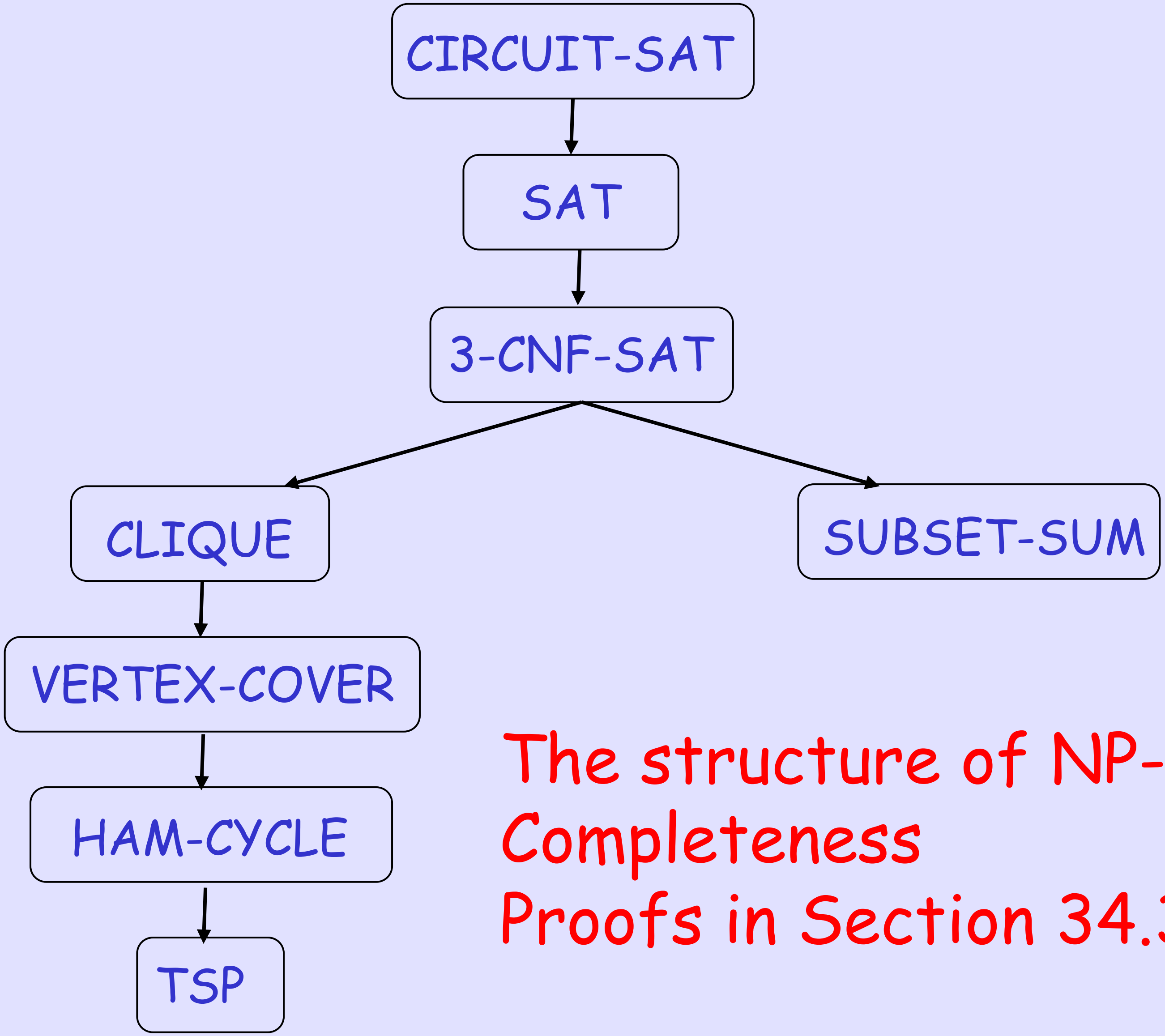
- Suppose that we can show that:

1. we can transform a problem of A into a problem of B, using polynomial time
2. We can answer A if we can answer B

➔ Then we can conclude B is NP-complete

(Can you see why??)

Problem Reduction



The structure of NP-Completeness
Proofs in Section 34.3

Problem Reduction

- 3-CNF satisfiability problem (3SAT):
  $(a \vee b \vee c) \wedge (a \vee d \vee e) \wedge (b \vee f \vee a)$
- All satisfiability problems can be reduced to 3-SAT problem in polynomial time.
- For example,
  - ✓ $(x1 \vee x2) \rightarrow (x1 \vee x2 \vee y1) \wedge (x1 \vee x2 \vee \neg y1)$
  - ✓ $\neg x3 \rightarrow (\neg x3 \vee y1 \vee y2) \wedge (\neg x3 \vee \neg y1 \vee y2) \wedge(\neg x3 \vee y1 \vee \neg y2) \wedge(\neg x3 \vee \neg y1 \vee \neg y2)$
  - ✓ $(x1 \vee x2 \vee x3 \vee x4 \vee x5) \rightarrow (x1 \vee x2 \vee y1) \wedge (\sim y1 \vee x3 \vee y2) \wedge (\sim y2 \vee x4 \vee x5)$
- Since 3-SAT is in NP, 3-SAT is an NP-Complete problem

Problem Reduction

- Let us define two problems as follows :
- The CLIQUE problem:

  Given a graph $G = (V,E)$, and an integer k, does the graph contains a complete graph with at least k vertices?
- The VERTEX-COVER problem:

  Given a graph $G = (V,E)$, and an integer k, does the graph contain k vertices such that all edges are covered by them?

Problem Reductin

- Questions:
  1. Are both problems decision problems?
  2. Are both problems in NP?

- In fact, CLIQUE is NP-complete (3-SAT can polynomial time reduce to CLIQUE)
- Can we use reduction to show that VERTEX-COVER is also NP-complete?

  [transform which problem to which?]

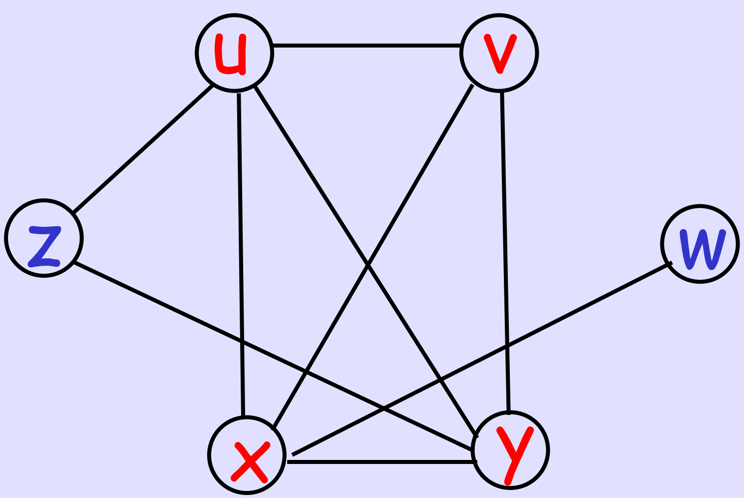
Problem Reduction

- Theorem: The VERTEX-COVER problem is NP-complete
- Proof:

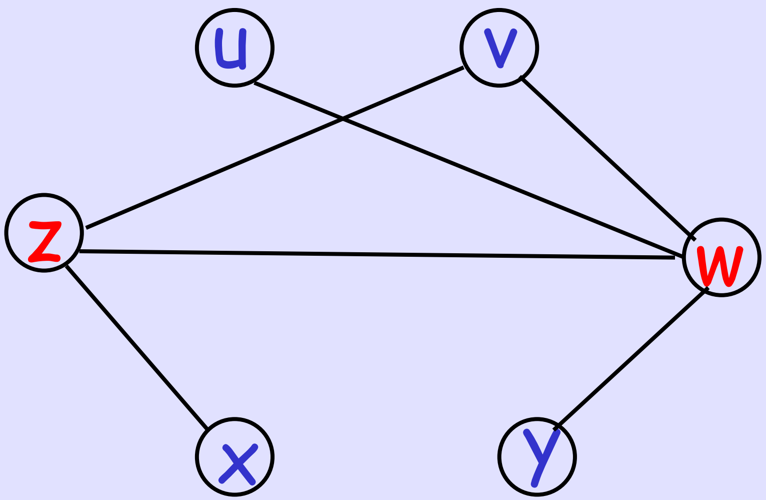
1. VERTEX-COVER $\in$ NP
2. Show that CLIQUE $\leq_p$ VERTEX-COVER

Given an undirected graph $G = (V, E)$, we define the complement of G as $G' = (V, E')$, where $E' = \{(u, v): u, v, \in V, u \neq v$, and $(u, v) \notin E\}$

Suppose that G' has a vertex cover $V' \subseteq V$. Then for all $u, v \in V$, if $(u, v) \in E'$, then $u \in V'$ or $v \in V'$ or both. This implies that for all $u, v \in V$, If $u \notin V'$ and $v \notin V'$, then $(u, v) \in E$. In other words, $V - V'$ is a clique, and its size $|V| - |V'| = k$



$G = (V, E)$



$G' = (V, E')$ $V' = \{z, w\}$

Problem Reduction

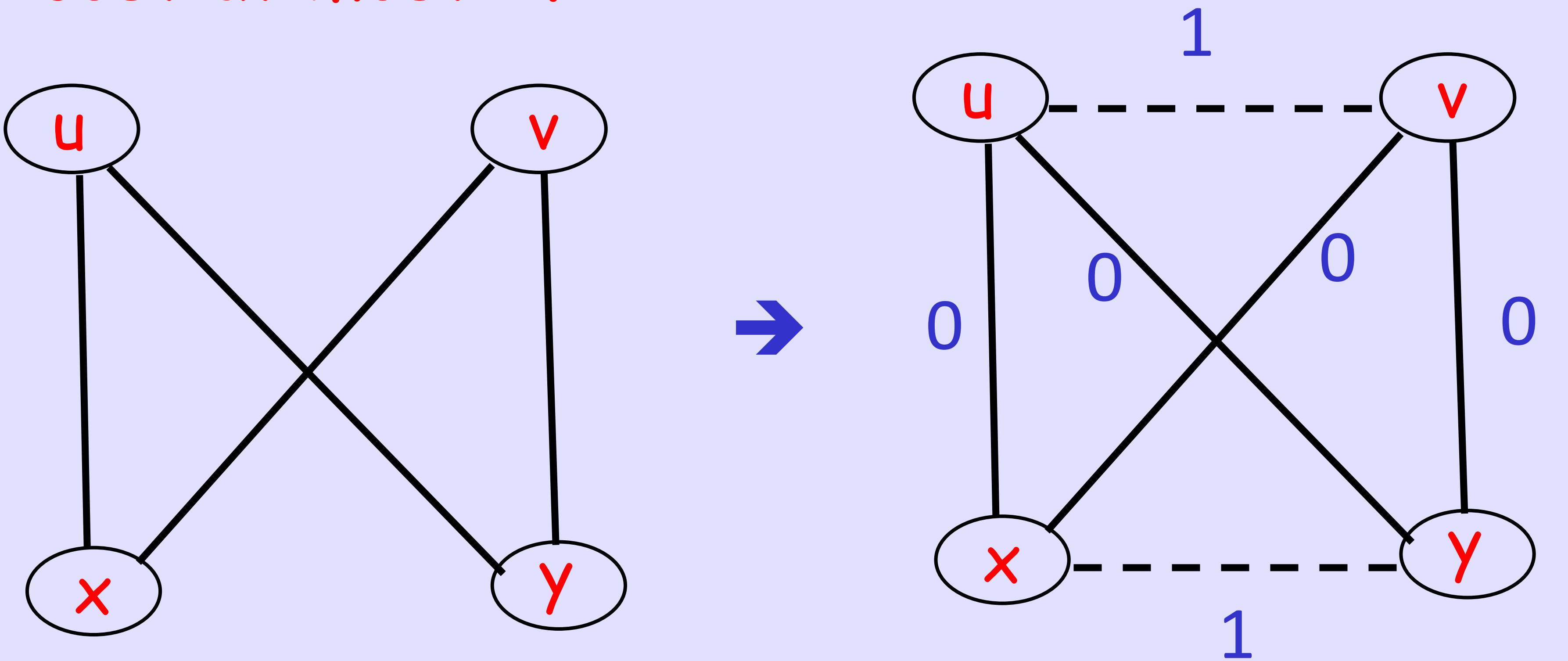
- Theorem: The traveling-salesman problem (TSP) is NP-complete
- Proof:

1. TSP $\in$ NP
2. Show that HAM-CYCLE $\leq_p$ TSP

Let G = (V, E) be an instance of HAM-CYCLE. We construct an instance of TSP as follows. We form a complete graph G' = (V, E'), where $E' = \{(i, j) \mid i, j, \in V, \text{ and } i \neq j\}$ and we define the cost function

$$C(i,j) = \begin{cases} 0 \; if \; (i,j) \in E \\ 1 \; if \; (i,j) \notin E \end{cases}$$

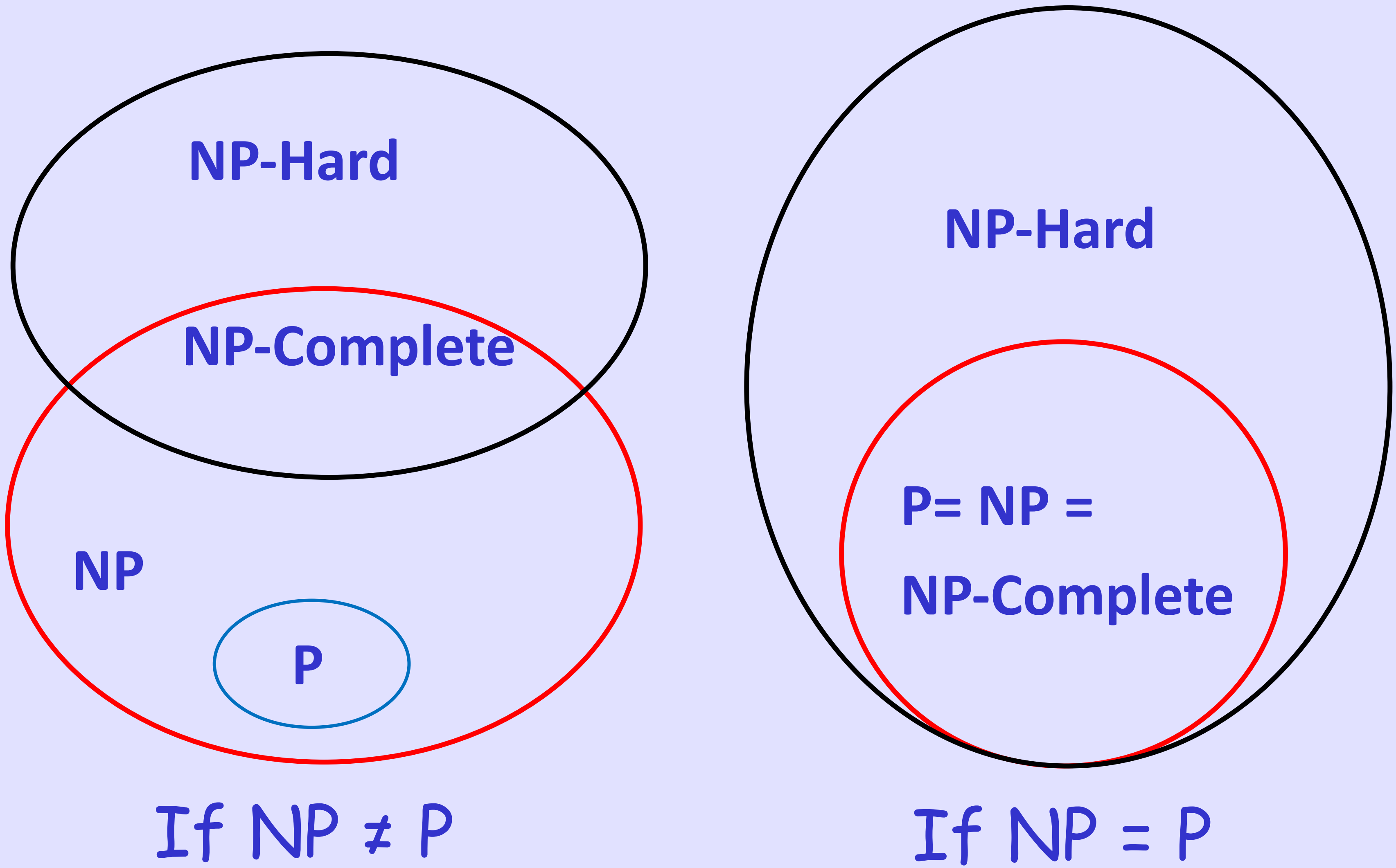
We can show that graph G has a Hamiltonian cycle iff graph G' has a tour cost at most 0.



NP-Complete & NP-Hard

- NP-Complete: a problem *A* is in NPC iff (i) *A* is in NP, and (ii) any problem in NPC can be reduced to it in $O(n^k)$ time.
- If any problem in NPC can be solved in $O(n^k)$ time, then P=NP. It is believed (but not proved) that $P \neq NP$.
- NP-Hard: a problem *A* is in NPH iff a problem in NPC can be reduced to *A* in $O(n^k)$ time.

NP-hard, NP, Np-Complete and P



Examples of NP-Complete

- The subset-sum problem: Given a positive integer set of S, we ask whether a subset $S' \subseteq S$ whose elements sum to t exists.
- The k-graph coloring problem ($k \geq 3$)
- The traveling-salesperson problem (TSP)
- The vertex-cover problem

Homework

- Exercises: 34.1-4, 34.2-2, 34.2-7
- Give a polynomial algorithm to solve the 2-CNF satisfiability problem

True or False

- If a problem is NPC, it cannot be solved by any polynomial time algorithm in the worst cases.
- If a problem is NPC, we have not found any polynomial time algorithm to solve it in the worst cases until now.
- If a problem is NPC, then it is unlikely that a polynomial time algorithm can be found in the future to solve it in the worst cases.

True or False

- If we can prove that the lower bound of an NPC problem is exponential, then we have proved that NP $\neq$ P.
- Any NP-hard problem can be solved in polynomial time if an algorithm can solve the satisfiability problem in polynomial time.

Quiz

- Suppose that all edge weights in a graph are integers ranging from 1 to $|V|$. How fast can you make Prim's algorithm run?
- How do you solve the single-source shortest paths problem in directed acyclic graphs (DAGs)?