Lecture #22
NP Complexity (2)

Algorithm

JBNU

Jinhong Jung

In This Lecture

- NP-Hard and NP-Complete
- Showing NP-Complete problems
  - Travelling salesman problem [TSP]
  - Longest path problem [LOGEST-PATH]
- NP-Hardness of optimization problem

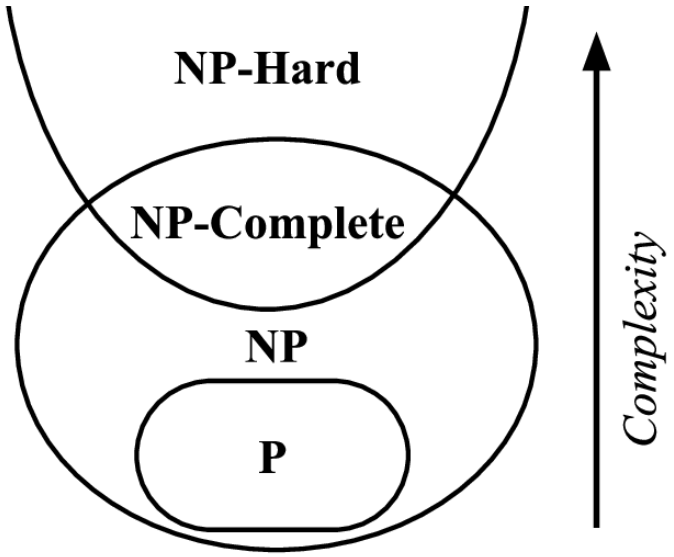
Outline

- ❑ Concept of NP
- ❑ Example - TSP
- ❑ Example – Longest Path
- ❑ NP-Hardness of optimization problem

Concept of NP

❑ P, NP, NP-Hard, NP-Complete

- P = set of decision problems s.t. can be solved in **poly-time**
- NP = set of decision problems s.t.
  - 1) Solved in poly-time by a non-deterministic computer
  - 2) Verified if "**the hint for Yes**" is correct or not in poly-time
- NP-Hard = set of problems s.t. each is reduced from every NP problem in poly-time
- NP-Complete = set of decision problems s.t. each is in NP-Hard as well as NP
  - Poly-algorithm for a problem in NP-$*$ is not yet known



How to Check NP-Hard

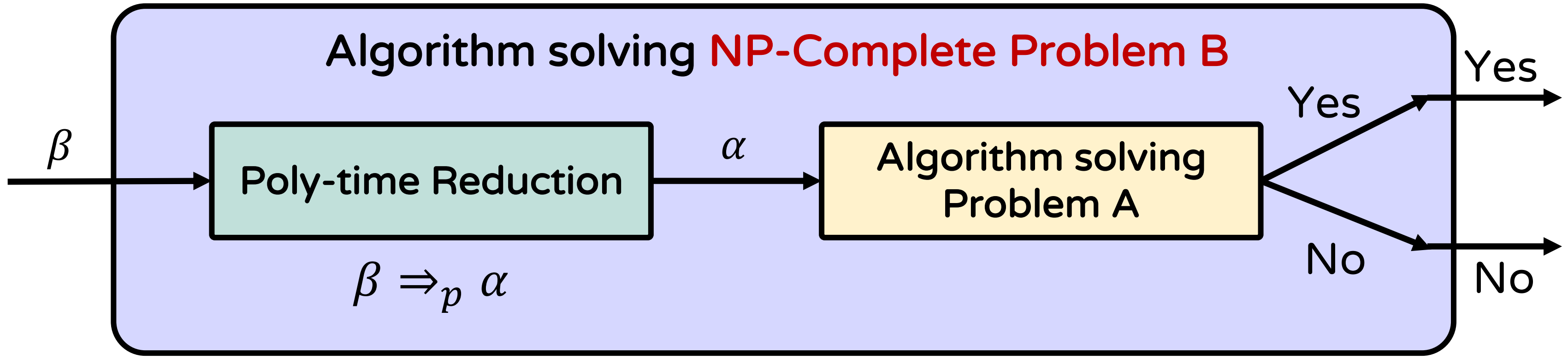
- Note that applying the original definition of NP-Hard is extremely difficult!
  - Because it is hard to manually check **all NP problems**

- Alternative approach to show NP-Hard
  - Problem A is **NP-Hard** if one NP-Hard problem $\mathrm{B} \Rightarrow_p \mathrm{A}$
  - Why?
    - $\mathrm{L} \Rightarrow_p \mathrm{B}\ \forall\ \mathrm{L} \in \mathrm{NP}$ because Problem B is NP-Hard **[by the original definition]**
    - $\mathrm{L} \Rightarrow_p \mathrm{B} \Rightarrow_p \mathrm{A}\ \forall\ \mathrm{L} \in \mathrm{NP}$ because $\mathrm{B} \Rightarrow_p \mathrm{A}$
    - Thus, $\mathrm{L} \Rightarrow_p \mathrm{A}\ \forall\ \mathrm{L} \in \mathrm{NP}$ meaning Problem A is NP-Hard

How to Check NP-Complete

❑ To show if Problem A is NP-Complete or not,

- **[NP-Hard]** Do poly-time reduction from a known NP-Complete (or NP-Hard) problem to Problem A



Algorithm solving NP-Complete Problem B
β
Poly-time Reduction
α
Algorithm solving Problem A
Yes
No
Yes
No
$\beta \Rightarrow_p \alpha$

Then, Problem A is also NP-Hard

- **[NP]** Show Problem A is in NP
  - Just check if "the hint of **Yes**" is correct or not in poly-time
  - For most cases, this is obviously easy

Beginning of NP-Complete Prob.

❑ Is there a known NP-Complete problem?

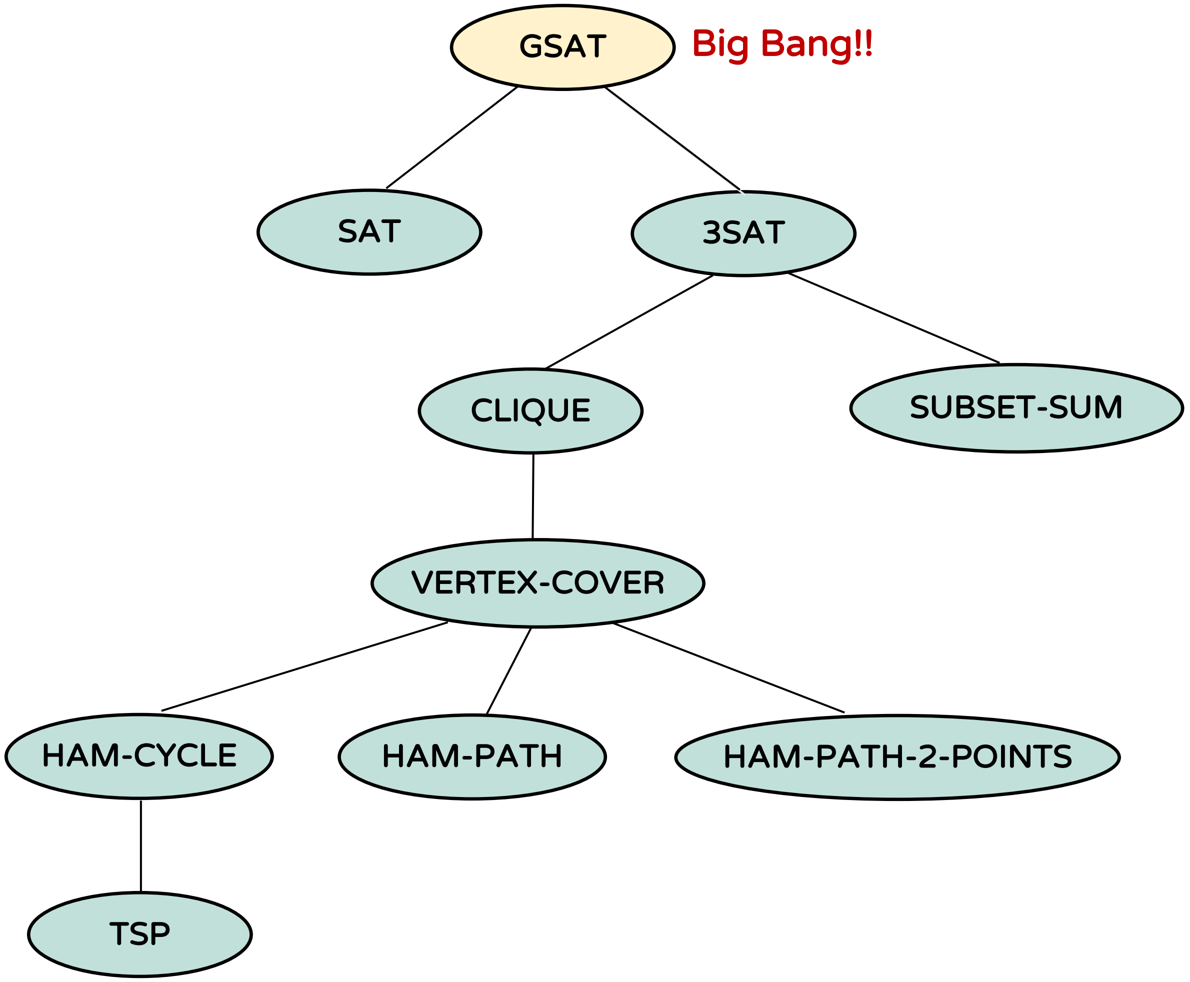
- Fortunately, Cook and Levin, inventors of NP theory, have shown a problem called **GSAT** is NP-Complete

❑ General satisfiability problem [GSAT]

- Boolean formula, e.g., $\left((a \vee b \vee c) \wedge d\right)$
  - If there is a configuration of variables that makes the formula **True**, then the Boolean formula is satisfiable
- Given an arbitrary Boolean formular, is it satisfiable?
- Proving GSAT is NP-Complete is out-of-scope
- Many NP-Complete problems are derived from GSAT!

NP-Complete Problems

❑ Lineage of NP-Complete family!



- NP-Hard indicates that the problems in NP-C shows the same difficulty.
- By definition of NP-Hard, every NP-Complete problem is reduced to each other.
- Poly-time algorithms for those are not yet known because they are in NP and P=NP is not yet proved.

Implication of NP-Complete (1)

❑ Why do we check if a problem is NP-Complete?

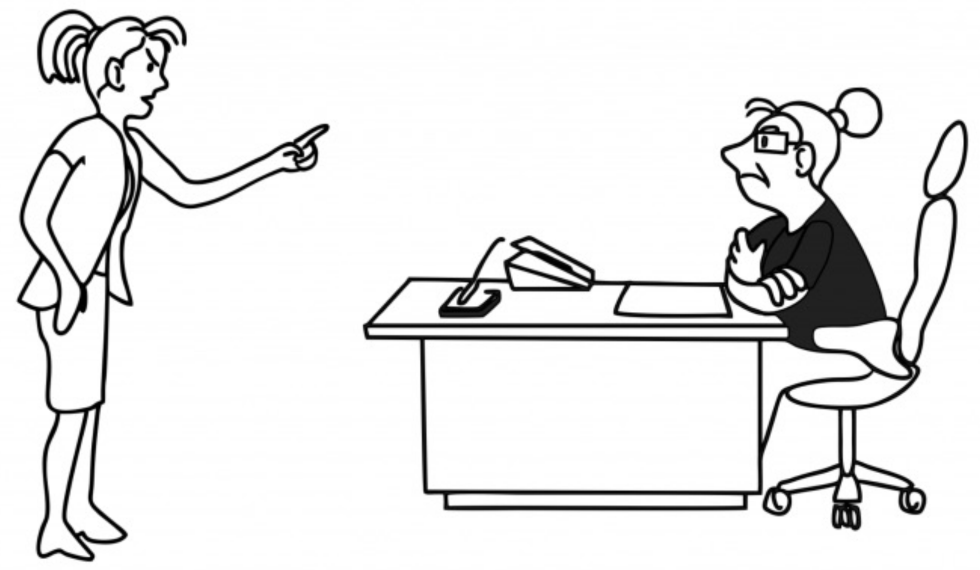
- Suppose your boss asked you for solving a hard problem; then, you might simply answer like

"I can't find an efficient algorithm, I guess I'm just too dumb."

Implication of NP-Complete (2)

❑ Why do we check if a problem is NP-Complete?

- After few tries to solve the problem, but you failed, so you might answer like with pure rage



“I can’t find an efficient algorithm, because no such algorithm is possible!”

But, I don’t know why…

Implication of NP-Complete (3)

- Why do we check if a problem is NP-Complete?
  - You are true CS warrior, so found the problem is in NP-Complete; then, you can convince your boss like

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

Because this problem is NP-Complete

Instead, let's develop an approximation algorithm

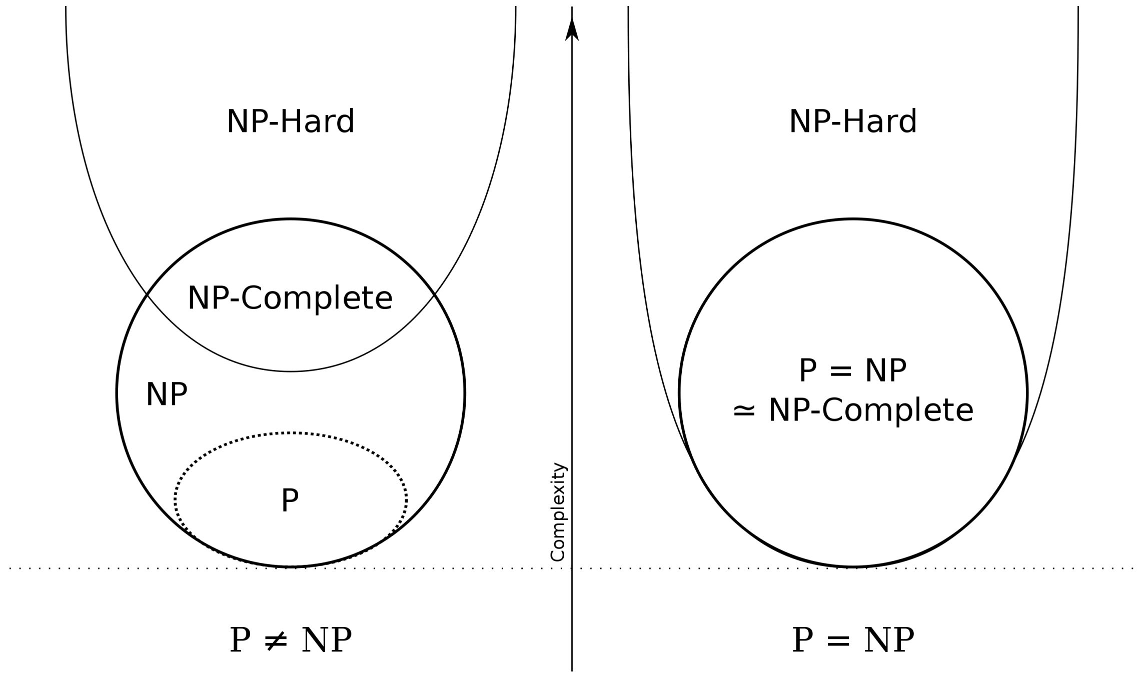
Now, You Can Say Like…

- Can we solve TSP quickly in polynomial time?
  - No, it can’t for now because **TSP is NP-complete**
    - i.e., it is NP-Hard as well as NP
  - NP=P has not yet proven, and polynomial algorithms for NP problems are not yet known (many geniuses have failed)
  - So TSP does!
  - Instead, let’s explore approximate algorithms!

Impact that P=NP

❑ P=NP implies that NP-Complete is also P

- If an efficient algorithm solves an NP-C problem in poly-time, then every NP problem can be solved in poly-time!
    - Because every NP-C problem is reduced to each other by the definition of NP-Hard



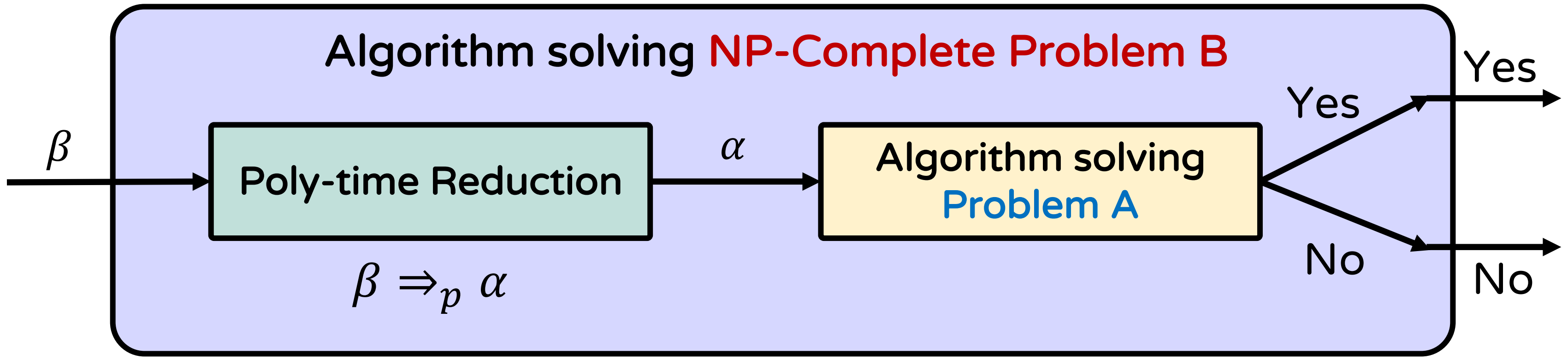
Outline

- ❑ Concept of NP
- ❑ Example - TSP
- ❑ Example – Longest Path
- ❑ NP-Hardness of optimization problem

How To Check NP-Complete

❑ Claim. Problem A is NP-Complete

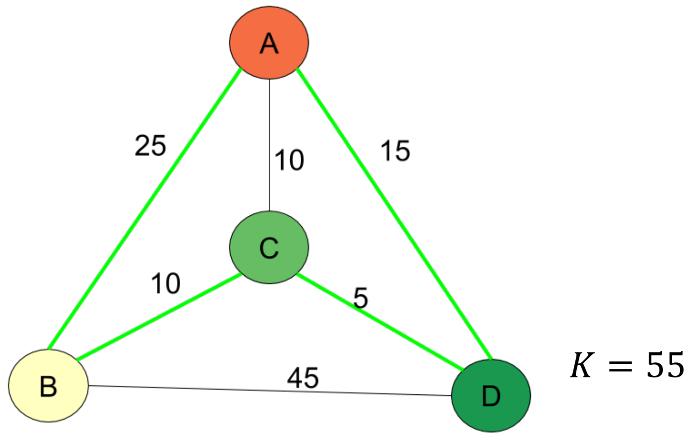
- **Step 1.** Show that Problem A is NP
- **Step 2.** Choose Problem B in NP-Complete (or NP-Hard)
- **Step 3.** Describe a poly-time algorithm that reduces an instance of Problem B to that of Problem A
- **Step 4.** Show both instances of A and B produces the same answer


Algorithm solving NP-Complete Problem B
β
Poly-time Reduction
$\beta \Rightarrow_p \alpha$
α
Algorithm solving Problem A
Yes
No
Yes
No

Is TSP NP-Complete?

❑ Travelling Salesman Problem [TSP]

- Given a weighted, undirected, & complete graph G, is there a Hamiltonian cycle such that its cost $\leq K$ in G?



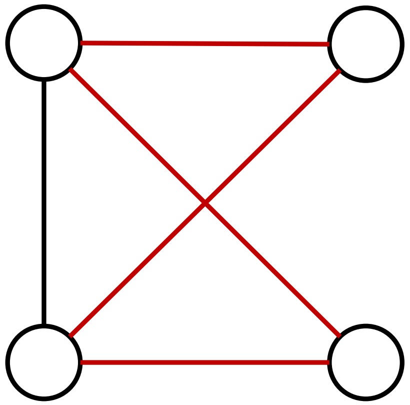
❑ To show NP-Completeness of TSP,

- **[NP-Hard]** Poly-time reduction from HAM-CYCLE to TSP
- **[NP]** Show TSP is in NP

Is TSP NP?

❑ Claim: TSP is in NP

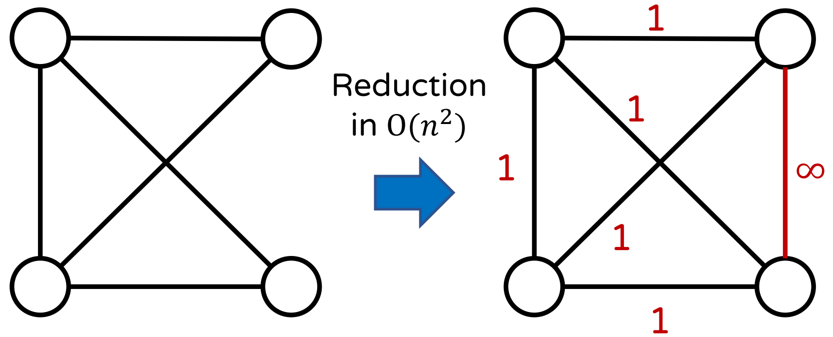
- TSP is a decision problem
- Instance and its hint certifying "**Yes**" is given
  - The graph (input) and the red colored path (hint)
- "The answer for the hint is **Yes**" can be verified in poly-time $O(n)$ by checking if the path is a cycle of length n



Is TSP NP-Hard?

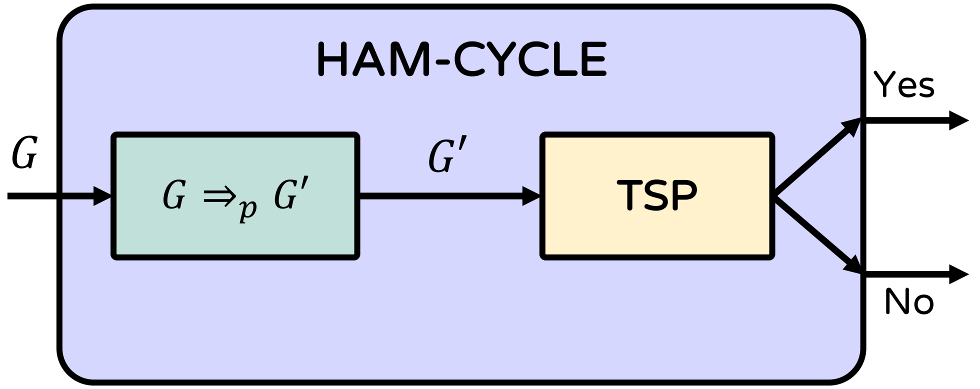
❑ Poly-time reduction from HAM-CYCLE to TSP

- Note that HAM-CYCLE is already proved as **NP-Complete**
- G is the graph of HAM-CYCLE, & G is reduced to G' of TSP
  - For existing edges, put weight 1 to each edge
  - For non-existing edges, fill each of them with weight ∞
  - This takes $O(n^2)$ time because there are $O(n^2)$ edges
- By $K = n$, the answer of TSP is that of HAM-CYCLE



Instance G of HAM-CYCLE

Instance G' of TSP
$K = n$



Now, You Can Say Like…

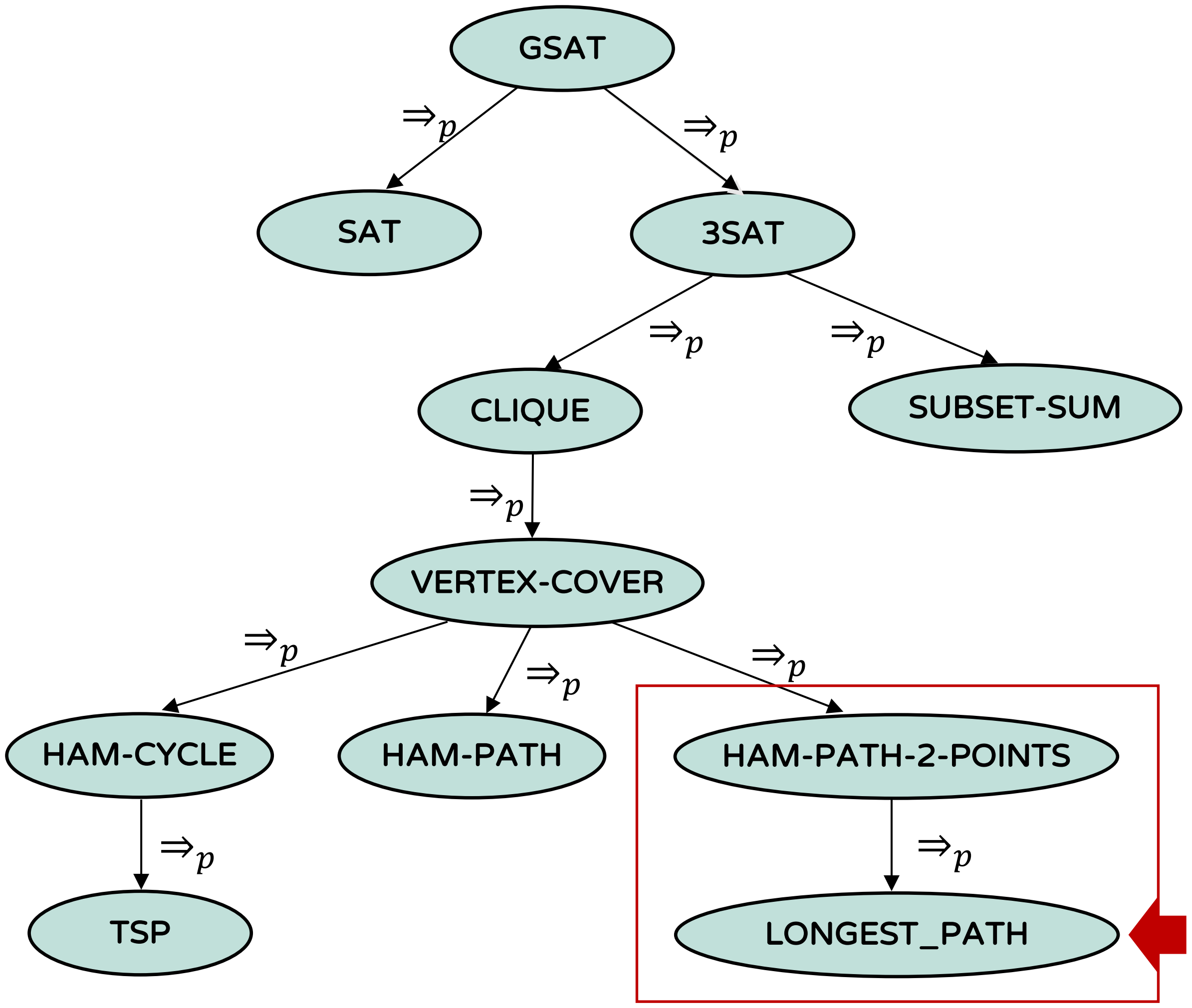
- Can we solve TSP quickly in polynomial time?
  - No, it can’t for now because TSP is NP-complete
    - i.e., it is NP-Hard as well as NP
  - NP=P has not yet proven, and polynomial algorithms for NP problems are not yet known (many geniuses have failed)
  - So does TSP!
  - Instead, let’s explore approximate algorithms!
    - Approximate algorithm for TSP is out-of-scope (see the textbook)

Outline

- ❑ Concept of NP
- ❑ Example - TSP
- ❑ Example – Longest Path
- ❑ NP-Hardness of optimization problem

NP-Complete Problems

❑ Lineage of NP-Complete family



Is it NP-Complete?

Longest Path Problem

Longest path problem [LONGEST-PATH]

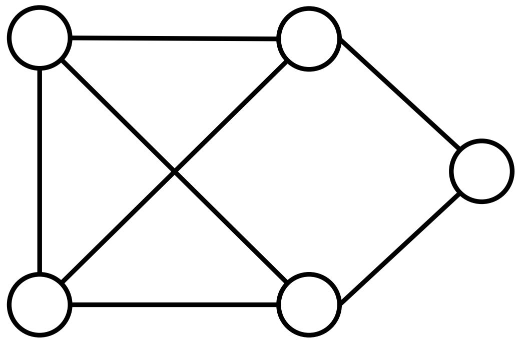
- Inputs
  - A positively weighted and undirected graph G
  - Two nodes s and t
  - Positive constant K

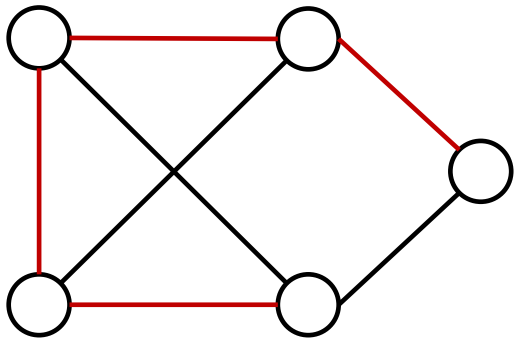
- Question
  - Is there a simple path of cost $\geq K$ from node s to node t in G?
    - A simple path if it does not have any repeated nodes
    - The cost is the sum of all edge weights of the path

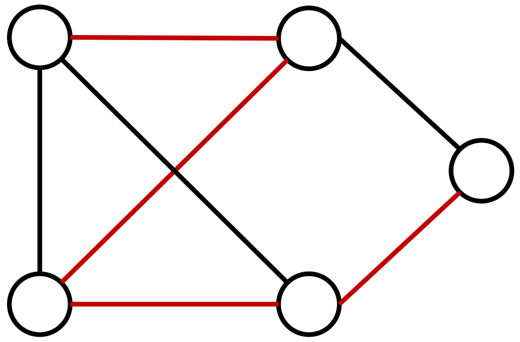
Hamiltonian Path Problem

❑ Hamiltonian path between two nodes [HAM-PATH-2-POINTS]

- **Inputs**
  - An undirected graph G & two nodes s and t
- **Question**
  - Is there a Hamiltonian path from node s to node t?
    - Hamiltonian path is a simple path visiting all nodes exactly once


Input graph G


Hamiltonian path 1


Hamiltonian path 2

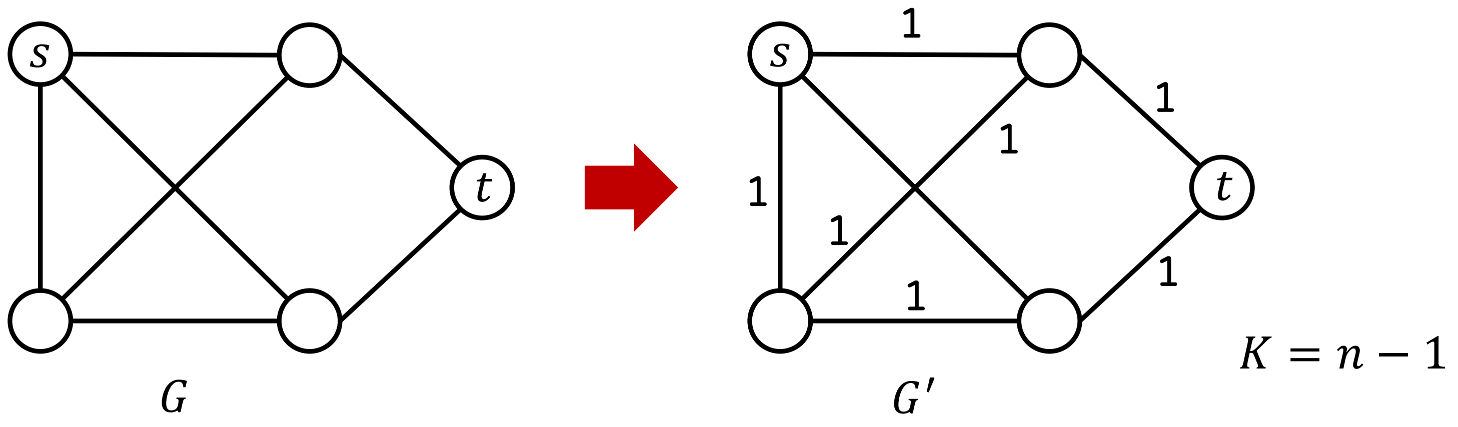
Proof of NP-Completeness

❑ Claim. LONGEST-PATH is NP-Complete

- **Step 1.** Show that the problem is NP
- 1) The problem is a decision problem
- 2) Suppose a hint of "Yes" is given as follows
  - Instance: a graph
  - Hint: a simple path of cost $\geq K$ from node s to node t
- 3) It is easily verifiable by comparing the sum of all of edge weights following the path with K
  - Done in $O(n)$ (note that # of edges of a simple path $< n$) where n is # of nodes

Proof of NP-Completeness

- **Claim.** LONGEST-PATH is NP-Complete
  - **Step 2.** Choose HAM-PATH-2-POINTS (NP-Hard)
  - **Step 3.** Do a poly-time reduction from HAM-PATH-2-POINTS to LONGEST-PATH
    - Let a graph G be an instance of HAM-PATH-2-POINTS
    - Simply put weight 1 to all edges of G, resulting in a new graph G'
      - Done in $O(n^2)$ time where n is # of nodes



Instance of HAM-PATH-2-POINTS   Instance of LONGEST-PATH

Proof of NP-Completeness

- **Step 4.** Show both instances produces the same answer
  - B: There is a Hamiltonian path from s to t in G $\Leftrightarrow$
  - A: There is a simple path of cost $\geq n-1$ from s to t in G'
- Proof that B $\Rightarrow$ A
  - If there is a Hamiltonian path from s to t, the path consists of $n-1$ edges in G
  - The cost of this (simple) path is $n-1$ in G'
  - Thus, there is a simple path of cost $\geq n-1$ from s to t in G'
- Proof that B $\Leftarrow$ A
  - Suppose there is a simple path of cost $\geq n-1$ from s to t in G'
  - A simple path cannot have more than $n-1$ edges; thus, its cost $n-1$
  - The corresponding path visits all nodes exactly once in G

Outline

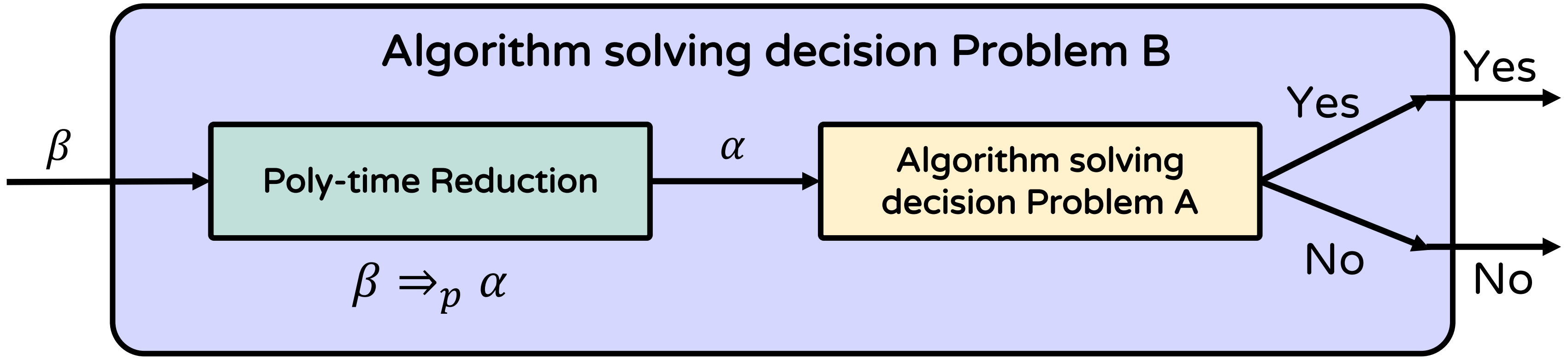
- ❑ Concept of NP
- ❑ Example - TSP
- ❑ Example – Longest Path
- ❑ NP-Hardness of optimization problem

NP-Hardness of Opt. Problems

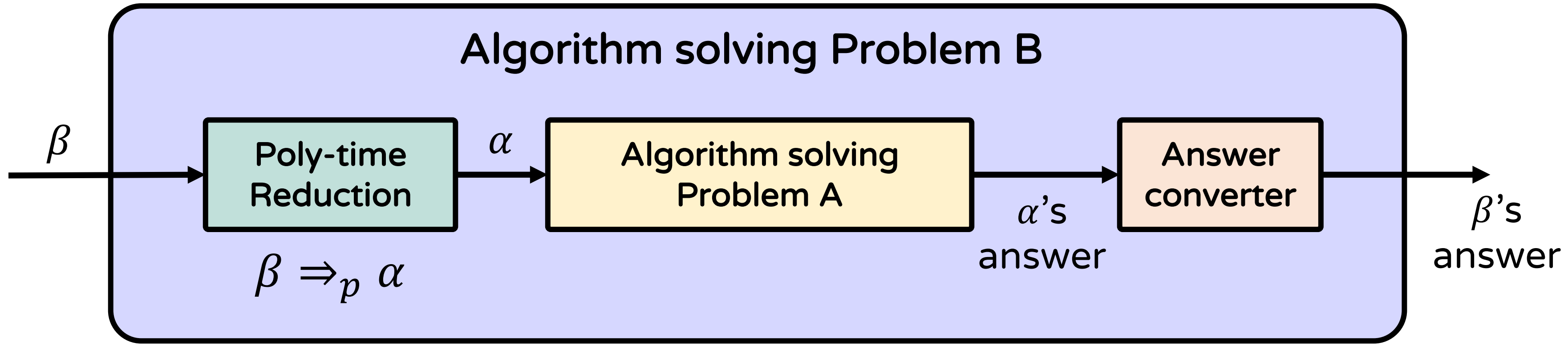
- We only considered decision problems for now..
  - **P**, **NP**, and **NP-Complete** contain only **decision problems**
  - As mentioned before, **NP-Hard** contains **decision problems** as well as **optimization problems**
- Q. How to show if an optimization problem is NP-Hard?

Poly-time Reduction

❑ Poly-time reduction for decision problems


Algorithm solving decision Problem B
β
Poly-time Reduction
α
Algorithm solving decision Problem A
Yes
No
Yes
No
$\beta \Rightarrow_p \alpha$

❑ Poly-time reduction for general problems



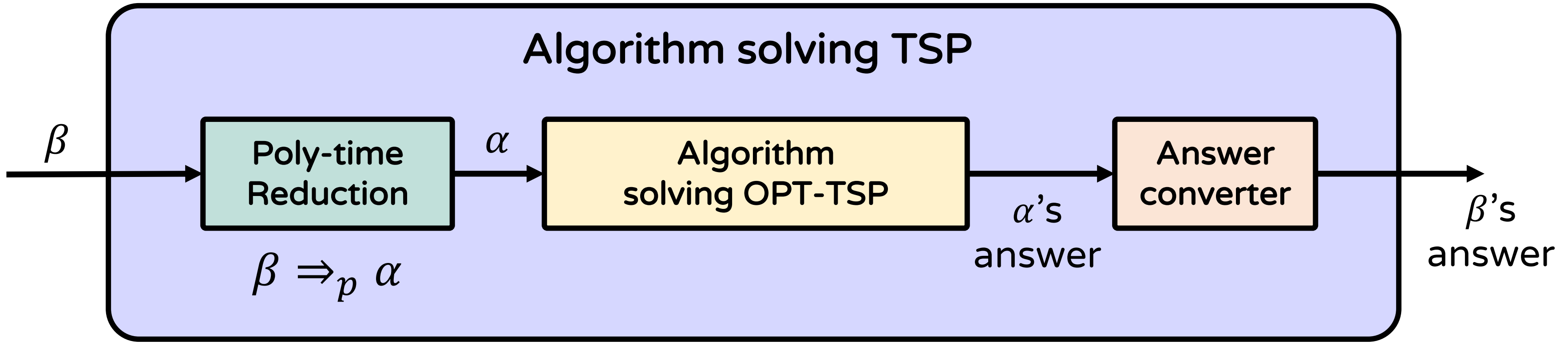
NP-Hardness of Opt. Problems

- In general, NP-Hardness of a decision problem is extended to that of its optimization problem
  - [OPT-TSP] The optimization version of TSP
    - Given a positively weighted, undirected, and complete graph G, what is the distance of the shortest Hamiltonian cycle in G?
  - [TSP] The decision version of TSP
    - Given G and positive constant K, is there a Hamiltonian cycle such that its cost $\leq K$ in G?
  - Let's check how to prove OPT-TSP is NP-Hard

Example of OPT-TSP

❑ Claim. OPT-TSP is NP-Hard

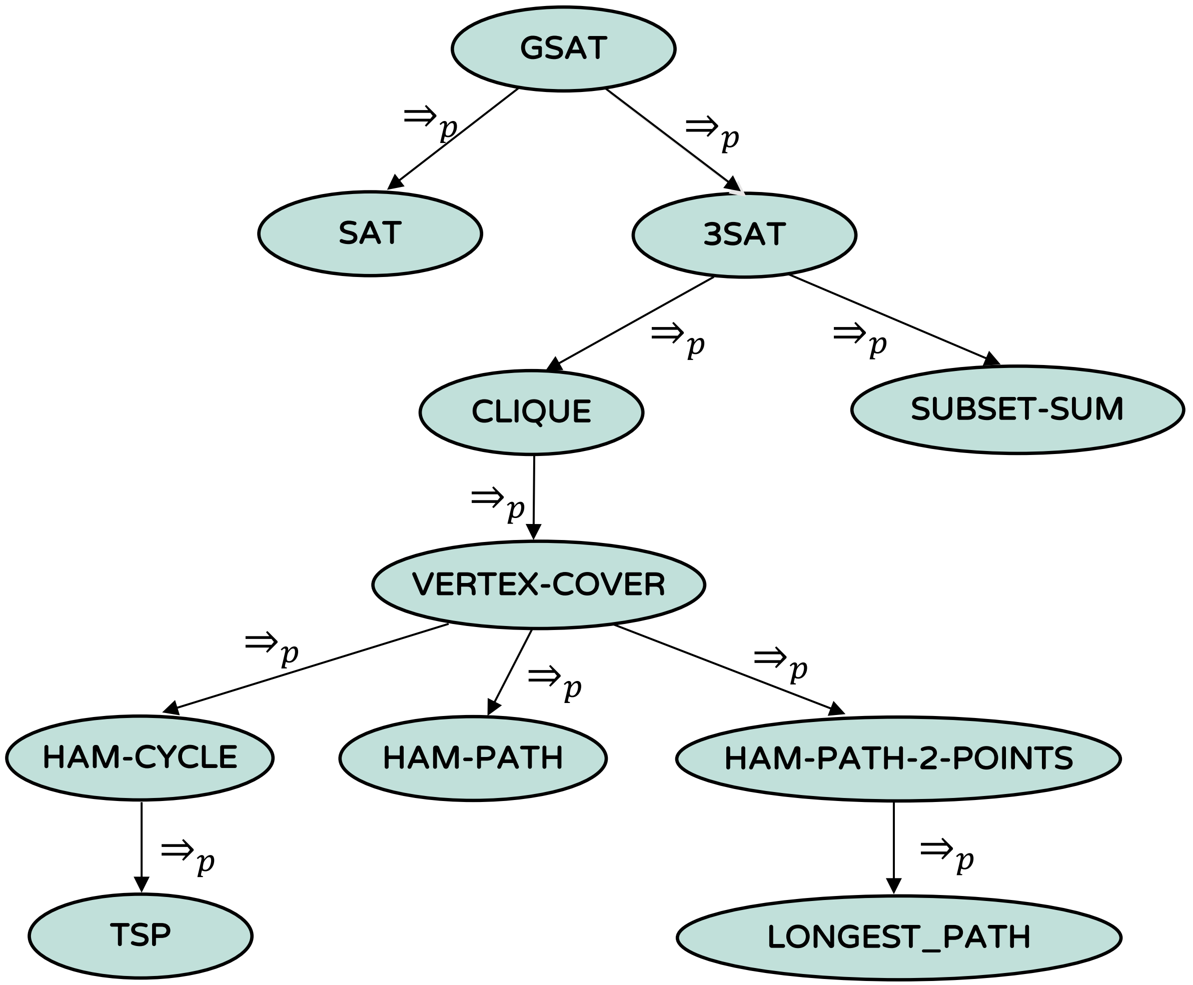
- Show TSP $\Rightarrow_p$ OPT-TSP where TSP is NP-Hard



- **Poly-time reduction**
  - **OPT-TSP** and **TSP** use the same graph (done in $O(1)$)
- **Answer converter**
  - Let M be the shortest distance obtained by **OPT-TSP**
  - If $M \leq K$, then **TSP**'s answer is "Yes"; otherwise, it is "No"

Other NP-Complete Problems

- Optimization versions of other NP-Complete problems are NP-Hard similarly



Not Covered Topics

❑ Other NP-Complete problems (p444~p452)

- GSAT $\Rightarrow_p$ SAT, SAT $\Rightarrow_p$ CLIQUE, …

❑ Approximate technique (p456~460)

- How to approximate an NP-Complete problem?

❑ Concept of Co-NP (p461)

- Co-NP: set of decision problems that "**the hint of No**" is verifiable in poly-time

What You Need To Know

❑ To prove NP-Completeness of Problem A

- **Step 1.** Show that Problem A is NP
- **Step 2.** Choose another Problem B which is NP-Hard
- **Step 3.** Describe a poly-time algorithm that reduces an instance of Problem B to that of Problem A
- **Step 4.** Show both instances of A and B produces the same answer

❑ To prove NP-Hardness of optimization problems

- **Step 1.** Show a decision problem is NP-Hard
- **Step 2.** Show its optimization problem is NP-Hard
    - As TSP $\Rightarrow_p$ OPT-TSP

In Next Lecture

❑ State Space Tree

- Concept of state space tree
- Backtracking
- Bounded branch (or branch and bound)

Thank You