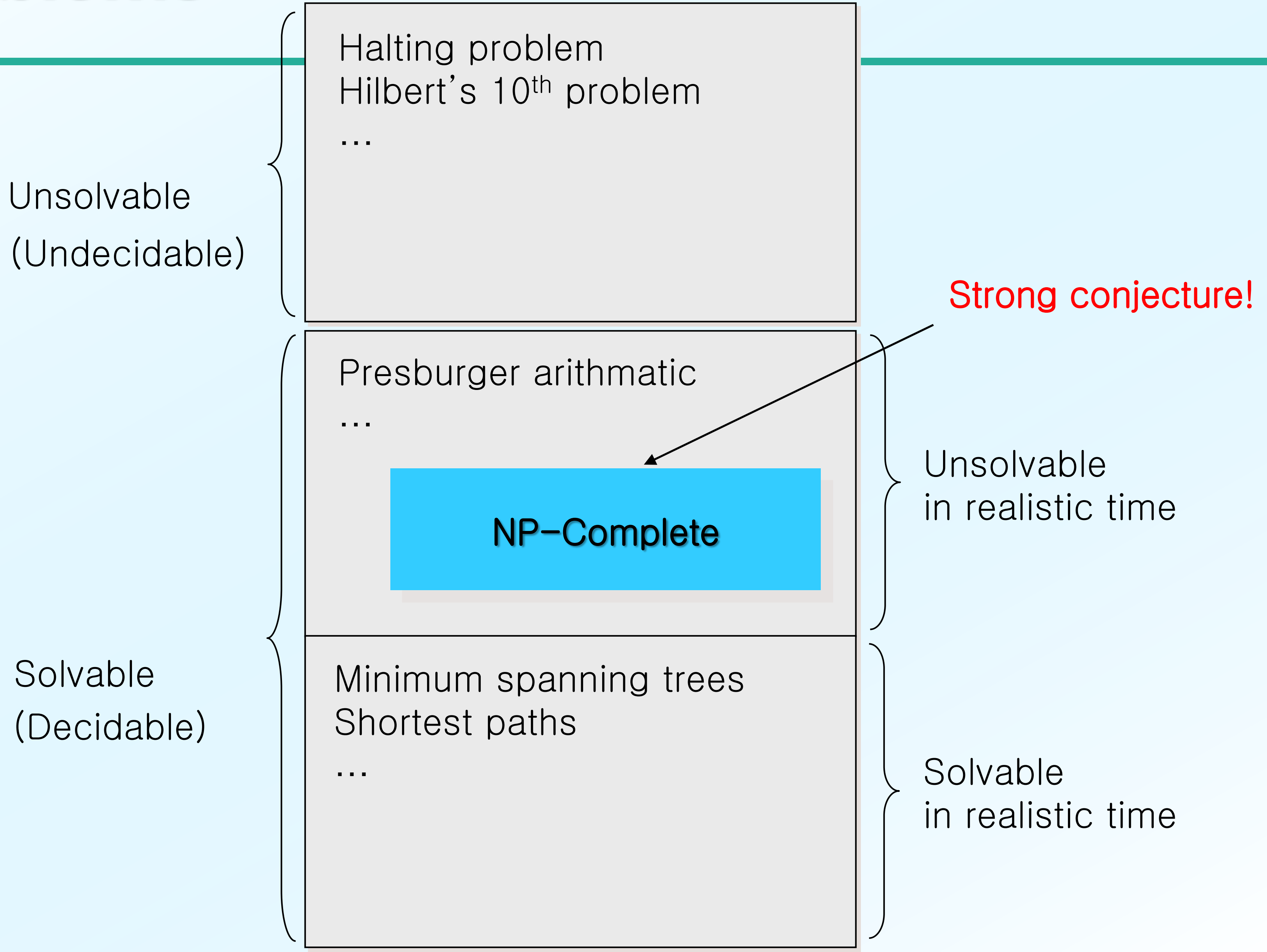
NP-Completeness

나는 그저 NP-Complete 이론이 흥미로운 발상이라고만 생각했다.
그것이 지닌 잠재적 영향력은 제대로 인식하지 못했다.
-- 스티븐 쿡

때로는 어떤 것이 불가능하다는 사실이 유용할 때도 있다.
-- 레오나드 레빈

Classes of Problems



Realistic, Tangible Time

- Means polynomial time
  - Represented by polynomials w.r.t. input size n
  - E.g., $3n^k + 5n^{k-1} + \ldots$
- Non-polynomials
  - Exponential
    - E.g., 2^n
  - Factorial
    - E.g., $n!$

Yes/No Problem : Optimization Problem

- Yes/No problem (Decision problem)
  - E.g.: Does a Hamiltonian cycle of length k or less in a graph *G* exist?
- Optimization problem
  - E.g.: What is the shortest Hamiltonial cycle length of in a graph *G*?

✓ The two classes of problems are closely related

NP-Complete

NP-Complete

- Restricts to decision problems (Yes/No problems)
  - But, says much about (twin) optimization problems
- About the possibility of solving in realistic time
- A huge class of problems
  - If a problem in this class is solved in realistic time,
    all problems in this class are solved in realistic time
  - Intuitively speaking, if a problem in this class is easy,
    all problems in this class become easy

So far..

- If a problem belongs to NP-Complete/NP-Hard

⇨ there is no realistic-time algorithm possible

to solve the problem (based on the research so far)

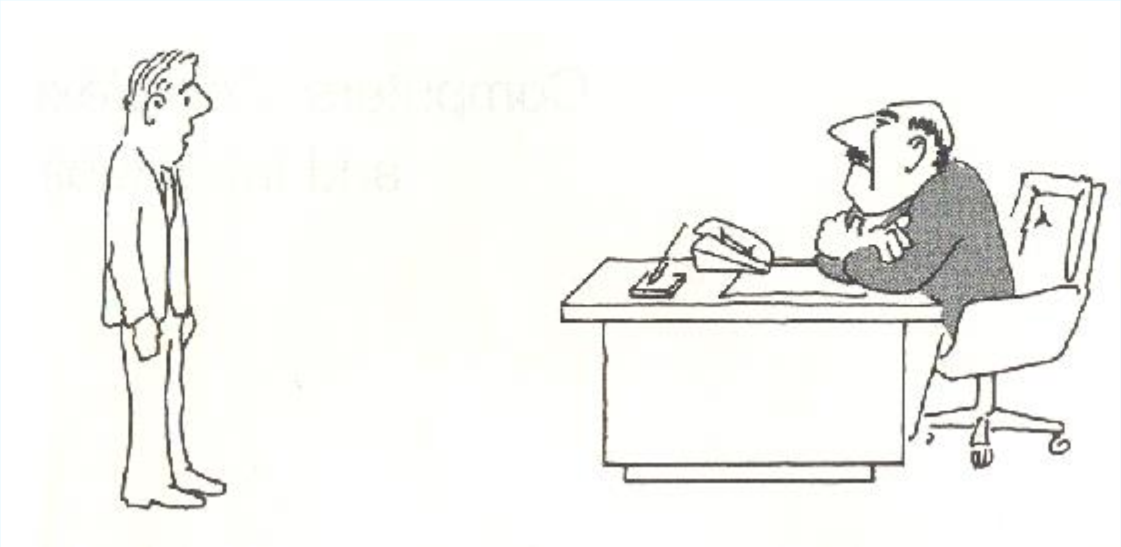
- But, not proven yet
- One of the seven Million-Dollar Problems of 21st Century

in Clay Mathematics Institute

- The problem of "P=NP?"

Current State of NP-Comple/Hard

The boss ordered to solve a very hard problem



1. I cannot solve it. Maybe, I'm stupid.

2. I cannot solve it. It has no solution.

An analogy to the state of NP-Complete

3. I cannot solve it. But those many genius people couldn't solve it.

Preparation: Logical Structure

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

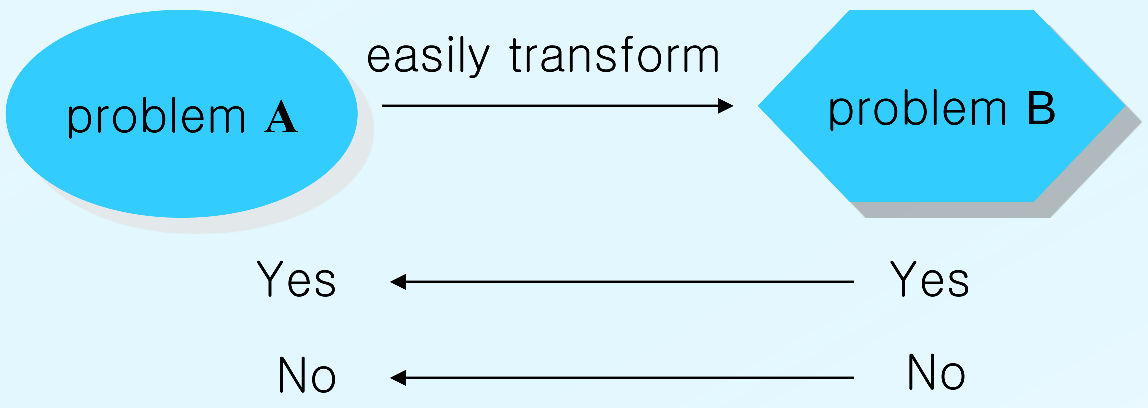
Problem 2: Is integer $X = x_1 + x_2 + \ldots + x_n$ a multiple of 3?

- ✓ The answers of the two problems are always the same
  - Answer: Yes or No

- ➢ If problem 2 is easy, then problem 1 is also easy.

easy = solvable in realistic time

- Situation
  - Problem B is easy
  - We can easily transform any instance of problem A
    to a matching instance of problem B
    where the answers(Yes or No) of the two instances are the same



✓ Is problem A also easy?

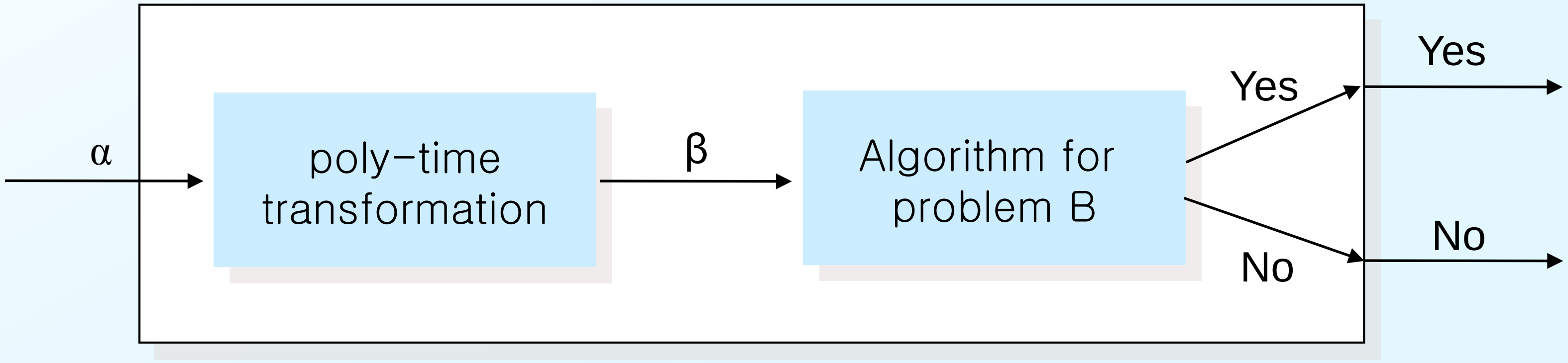
Poly-Time Reduction

$A \leq_p B$: A is polynomial-time reducible to B

if we can transform any instance α of problem A to a matching instance β of problem B satisfying the following:

1. The transformation is done in polynomial time
2. The answers(Yes or No) of the two instances are the same

polynomial time means $O(n^p)$



Algorithm for problem A

1. Transform problem A to problem B in poly time
2. Solve the transformed problem B
3. Return the answer of the transformed problem B

✓ If problem B is easy, problem A is also easy

Equivalent Definitions

HAM = {G: G has a Hamiltonian cycle} ← $G \in$ HAM

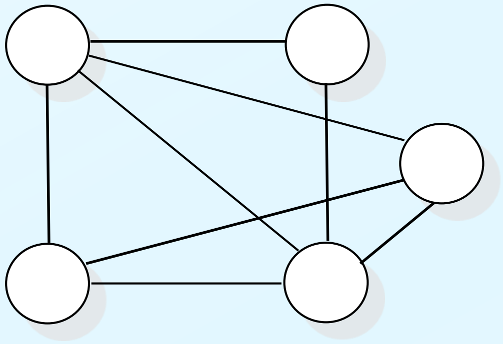
=

Or equivalently,

HAM:

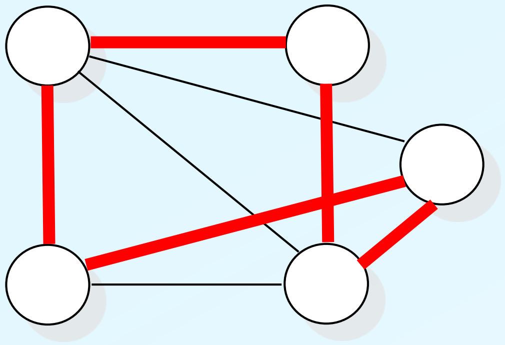
Input: a graph G

Question: Does G have a Hamiltonian cycle? ← Yes



Graph G

an instance of HAM



A Hamiltonian cycle in G

an example of $G \in$ HAM

a certificate of $G \in$ HAM

P and NP

a problem = a set of solutions

P and NP each is a set of problems

- P
  - Polynomial
  - Answers Yes or No in poly time

$L \in$ P: $\forall$ instance x, we can answer whether $x \in L$ or $x \notin L$ in poly time

- NP
  - Nondeterministic Polynomial
  - Note: it does not mean Non-Polynomial!
  - Given a certificate to answer Yes, we can verify in poly time that the certificate is right
  - No requirement for the case that the answer is No

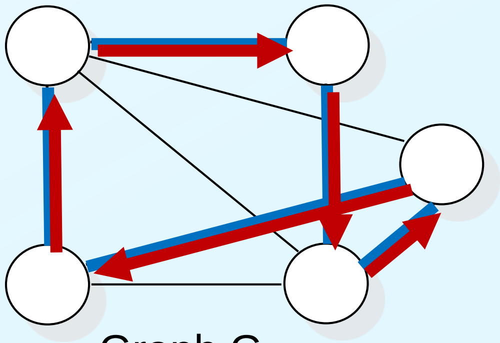
$L \in$ NP: $\forall$ instance $x \in L$, we can verify that $x \in L$ in poly time.

(given $x \in L$, check to see if $x \in L$)

a certificate

- Mostly, it is trivial to show a problem is in NP
  - A simple process in the proof of NP-Completeness

verify됨!



Graph G

Problem: Given G, is there a Hamiltonian cycle? (say Yes or No)

A certificate: a Hamiltonian cycle in G

Poly-time verification: check to see in poly-time that the certificate is a Hamiltonian cycle

NP-Complete/Hard

NP: Given a certificate to answer Yes, we can verify in poly time that the certificate is right

Definition 1: (NP-Hard)

or L is in NP-Hard

A problem L is NP-Hard

if any problem in NP can be poly-time reducible to L. (i.e., any NP problem $\leq_p L$)

Definition 2: (NP-complete)

A problem L is NP-complete if:

1. L is NP, and
2. L is NP-Hard

✓ Since NP-Complete is a subset of NP-Hard,
we can say an NP-Complete problem as NP-Hard

✓ Mostly the property 1(NP) of NP-Completeness is trivial,
we focus on the property 2 (NP-Hardness)

Extremely Hard

Definition 1: (NP-Hard)

A problem L is NP-Hard

if any problem in NP can be poly-time reducible to L.

= all

Extremely hard to prove this.

We need an alternative to prove NP-Hardness.

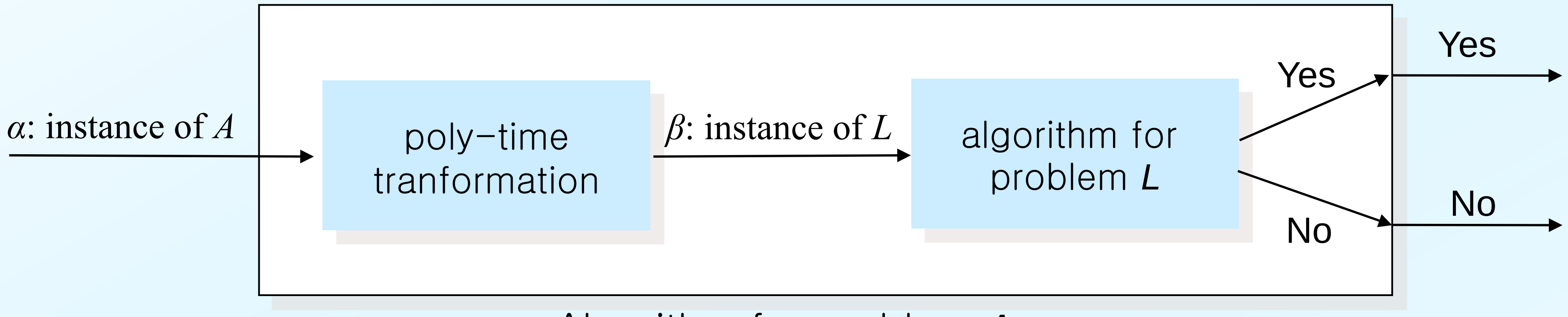
An Alternative to Prove NP-Hardness

Theorem 1:

A problem L is NP-Hard

if a known NP-Hard problem A can be poly-time reducible to L. (i.e., $A \leq_p L$)

<Proof>



Algorithm for problem A

This means $A \leq_p L$.

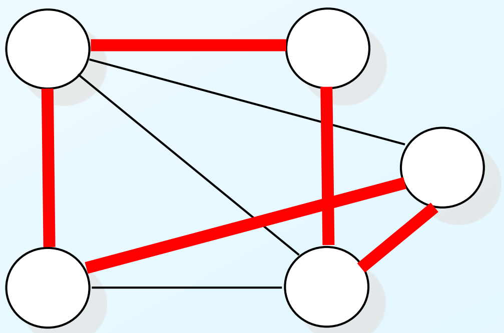
Since it is a given condition that any NP problem $\leq_p A$, any NP problem $\leq_p L$. (by transitivity) ■

✓ If problem L is easy, then problem A is also easy.

➔ Therefore, all NP problems are easy.

Example: NP-Hardness Proof

- Given fact: HAM (Hamiltonian cycle problem) is NP-Hard
- We can prove that TSP is NP-Hard using the given fact.



a Hamiltonian cycle

- Hamiltonian cycle (of an undirected graph)
  - A simple cycle to visit every vertex
- Hamiltonian cycle problem (HAM)

  Given: an undirected graph G

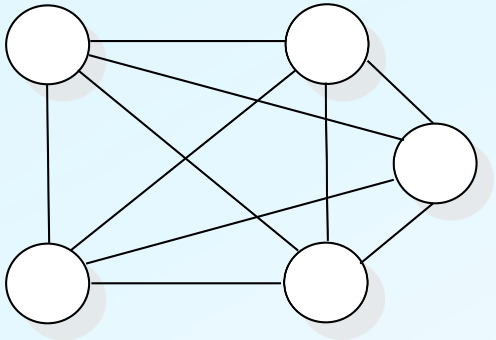
  Question: Is there a Hamiltonian cycle in G?

- TSP

  Given: a weighted undirected complete graph G

  Question: Is there a Hamiltonian cycle of length k or less in G?

edge between every pair of vertices



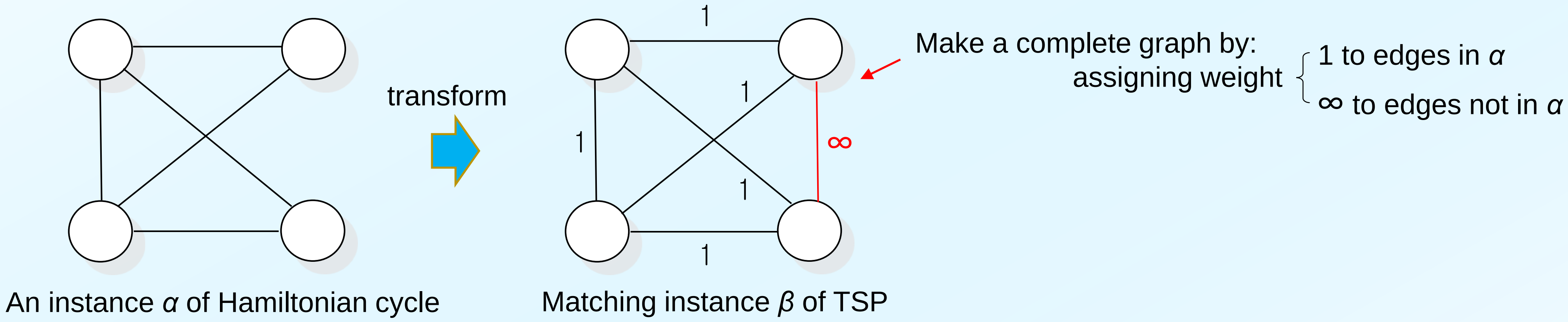
- Claim:

  TSP is NP-Hard → Proof: next page

HAM $\leq_p$ TSP

We say HAM is poly-time reducible to TSP

Transform an instance α of Hamiltonian cycle problem to an instance β of TSP in poly time as follows:



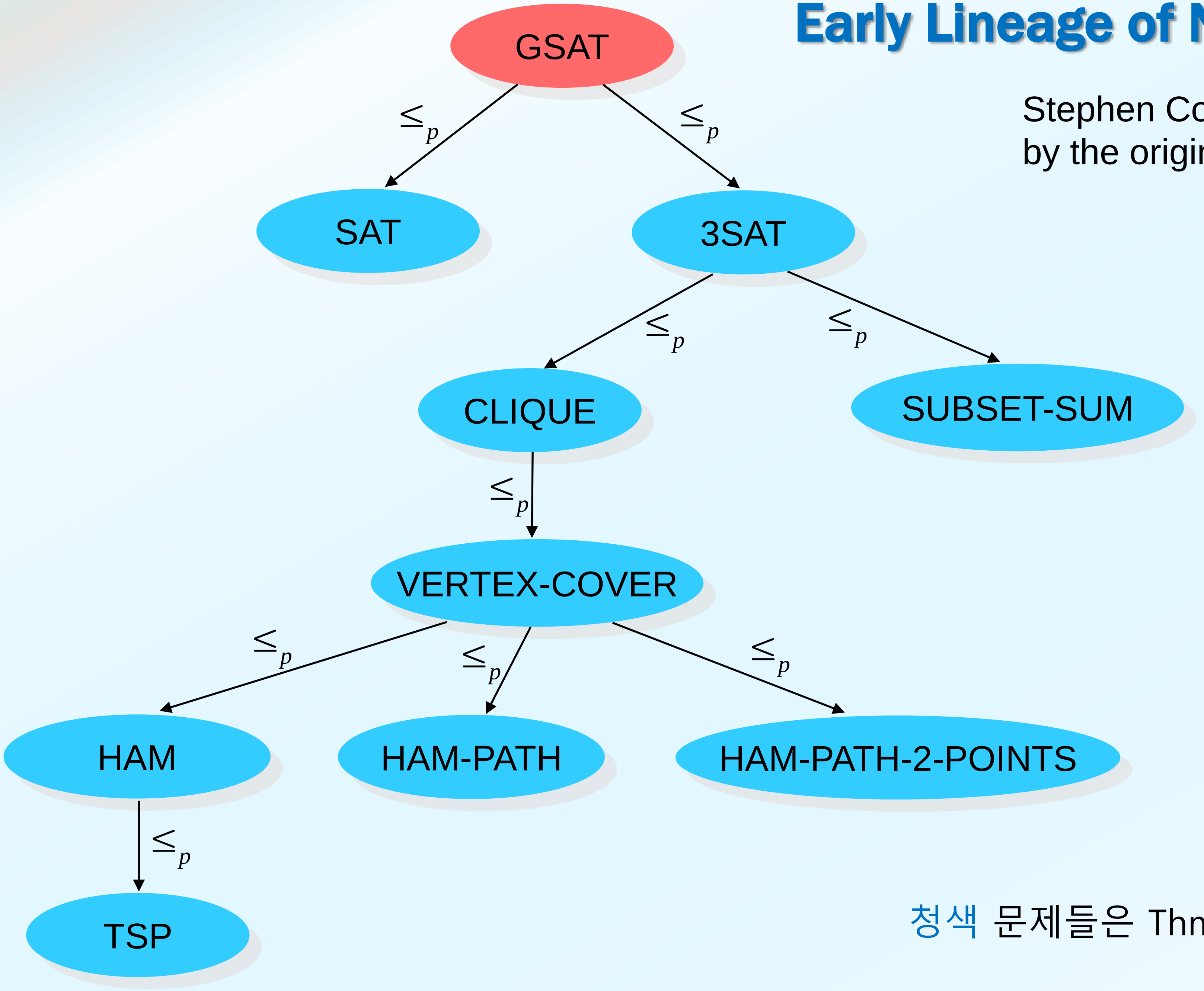
Instance α has a Hamiltonian cycle

⇔ Instance β has a Hamiltonian cycle of length n or less (n = 4 in this example)

# of vertices

Therefore, TSP is NP-Hard. ∎

Early Lineage of NP-Complete Problems



Stephen Cook proved that GSAT is NP-Complete by the original definition (Def 2)

청색 문제들은 Thm. 1에 의해 NP-Hard임을 증명

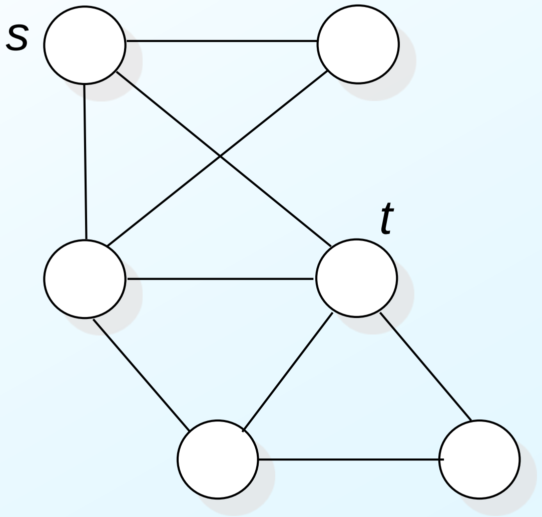
Counter-Intuitive Example of NP-Complete

- Shortest path problem
  - Shortest (simple) path from vertex *s* to *t*
  - Easy
- Longest path problem
  - Longest (simple) path from vertex *s* to *t*
  - NP-Hard

✓ Apparently similar, but extremely different! (Based on research so far)

- LONGEST-PATH
  - Longest path problem
  - (Optimization version) Given a weighted (undirected) graph,
    what is the longest simple path length from vertex *s* to vertex *t*?
  - (Yes/No version) Given a weighted (undirected) graph,
    is there any simple path from vertex *s* to vertex *t* of length *K* or greater?
- HAM-PATH-2-POINTS
  - Hamiltonian path problem between two vertices
  - Given a (undirected) graph, is there any Hamiltonian path from vertex *s* to vertex *t*?
  - Known to be NP-Complete

Property 1: LONGEST-PATH is NP



If we are provided a simple path from *s* to *t* of length *K* or greater (a certificate), we can verify in poly time that it is a simple path of length *K* or greater.

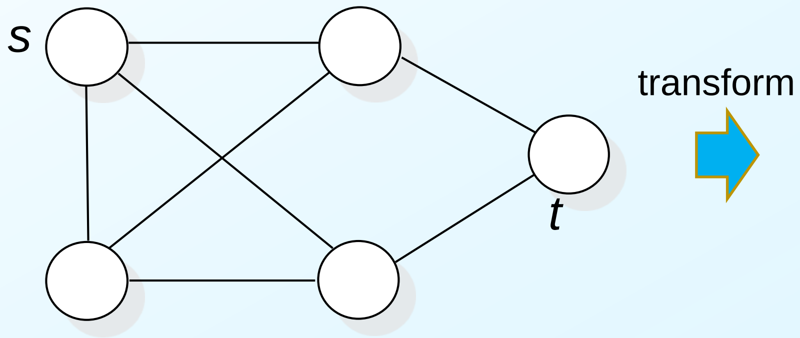
(We can check it in $O(n^2)$, a comfortable poly time)

It means LONGEST-PATH is NP. (Property 1 of NP-Completeness)

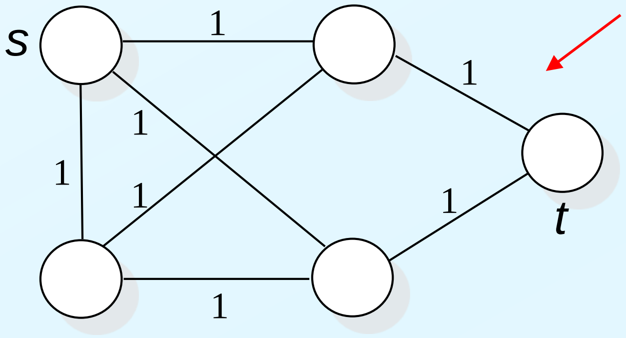
✓ Proving NP (Property 1) is this easy in most cases.
That’s why we focus on the NP-Hardness.

Property 2: HAM-PATH-2-POINTS $\leq_p$ LONGEST-PATH

Transform an instance *α* of HAM-PATH-2-POINTS
to an instance *β* of LONGEST-PATH in poly time as follows:



instance *α* of HAM-PATH-2-POINTS



Using the same graph,
assign weight 1 to all the edges in *α*

Matching instance *β* of LONGEST-PATH

Instance *α* has a Hamiltonian path between *s* and *t*

⇔ Instance *β* has a simple path between *s* and *t* of length n-1 or greater (in fact, exactly n-1)

(in this example, n-1 = 4)

➢ Therefore, LONGEST-PATH is NP-Hard. (Property 2 of NP-Complete)

By Properties 1 & 2, LONGEST-PATH is NP-Complete.

CLIQUE

- Input

  (unweighted, undirected) Graph $G = (V, E)$ and a positive integer K
- Question

  Is there any complete subgraph (clique) of size K in G?

- Claim: CLIQUE is NP-Complete

- 3SAT
  - Input: a Boolean formula ϕ in 3-CNF
  - Question: Is ϕ satisfiable?
  - Known to be NP-Complete

literals

. . .

Example: Boolean formula ϕ: $(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3} \vee x_4)$ This is in 3-CNF

clauses

Clause: literals are combined by ‘$\vee$’

CNF(Conjunctive Normal Form): clauses are combined by ‘$\wedge$’

3-CNF: clauses in CNF each has exactly 3 literals

Satisfying assignment: assignment (0 or 1) of literals that evaluates the formula to true.
If there is a satisfying assignment of ϕ, we say ϕ is satisfiable.

3SAT

- Input: a Boolean formula ϕ in 3-CNF
- Question: Is ϕ satisfiable?
- Known to be NP-Complete

3SAT $\leq_p$ CLIQUE

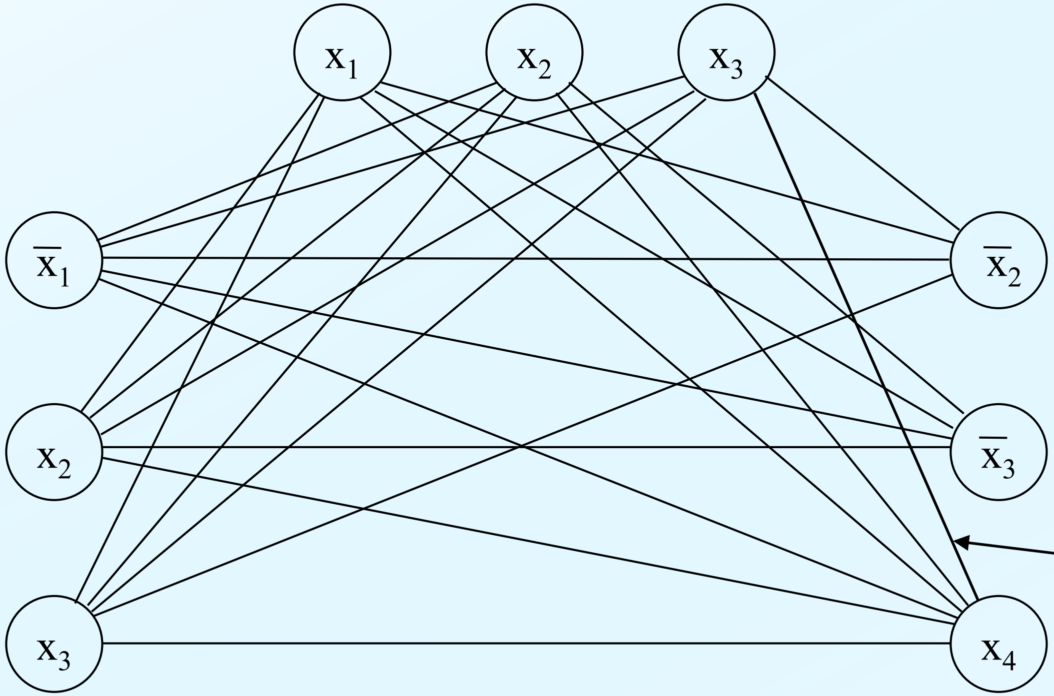
Example instance ϕ: $(x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4)$

Question: Is there a satisfiable assignment of x_i's that evaluates ϕ to true?

Make a graph(an instance of CLIQUE) by:

1. generating a vertex for each literal
2. set an edge between two vertices s.t.
   the corresponding two literals are consistent and
   they are not in the same clause.

$(x_1 \vee x_2 \vee x_3)$



$(\overline{x}_1 \vee x_2 \vee x_3)$

$(\overline{x}_2 \vee \overline{x}_3 \vee x_4)$

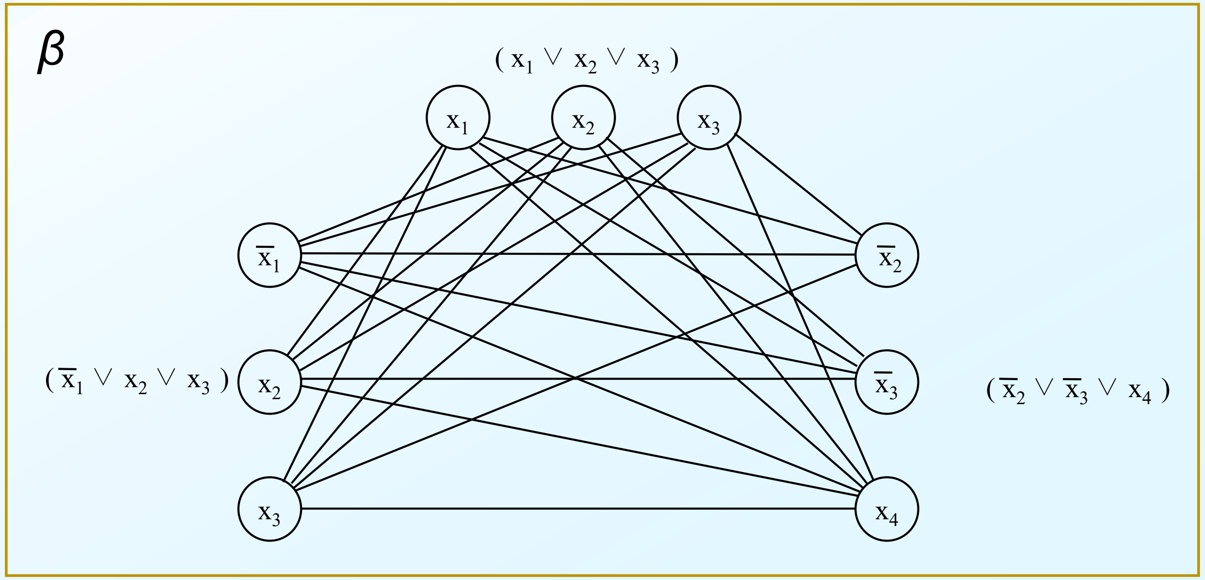
x_3 and x_4 are consistent, and
x_3 and x_4 are not in the same clause

Matching instance β of CLIQUE

3SAT $\leq_p$ CLIQUE

ϕ: $(x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4)$

m: # of clauses (3 in this example)



There is a satisfiable assignment of x_i's that evaluates ϕ to true
iff
there is a clique of size m in β

Obvious that the transformation takes a poly time.

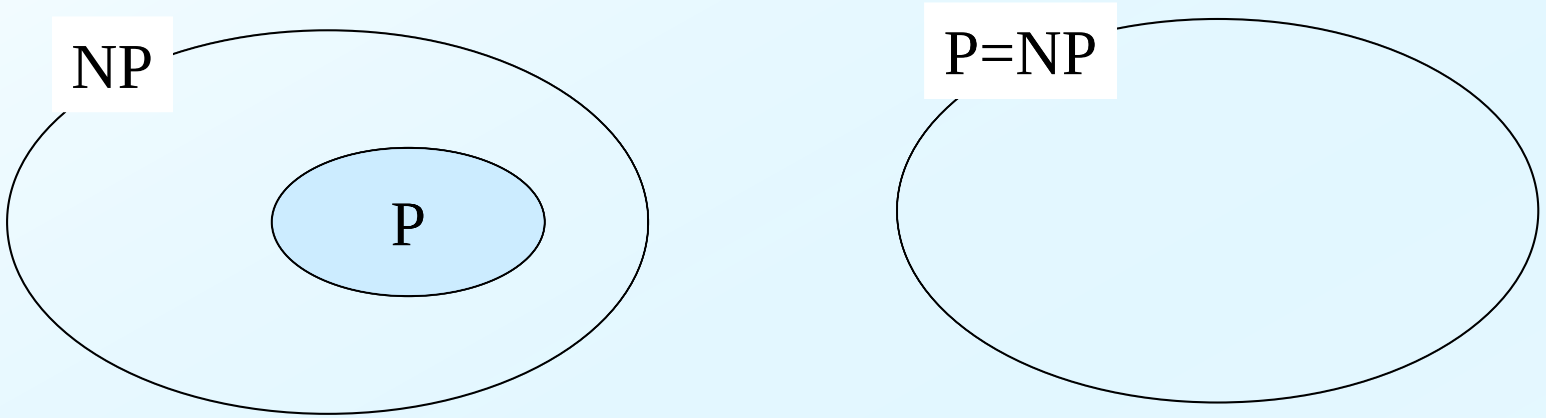
Therefore, 3SAT $\leq_p$ CLIQUE. ■

Is the Theory of NP-Hard any Useful?

- A problem is proved to be NP-Complete/Hard
  - ⇨ Stop to develop an easy algorithm
  - ⇨ Develop a heuristic algorithm that tries to find a suboptimal solution in the given time budget

Remind: 때로는 어떤 것이 불가능하다는 사실이 유용할 때도 있다.
-- Leonard Levin

Relationship of P and NP



(a)

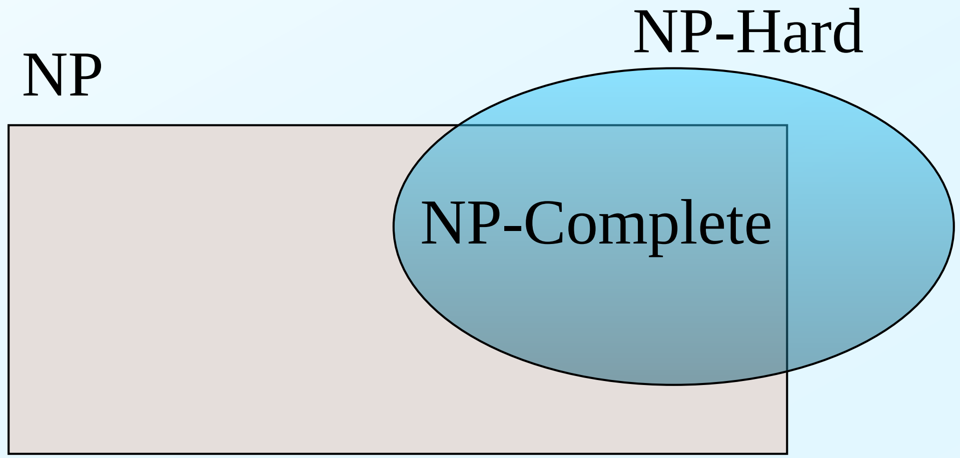
(b)

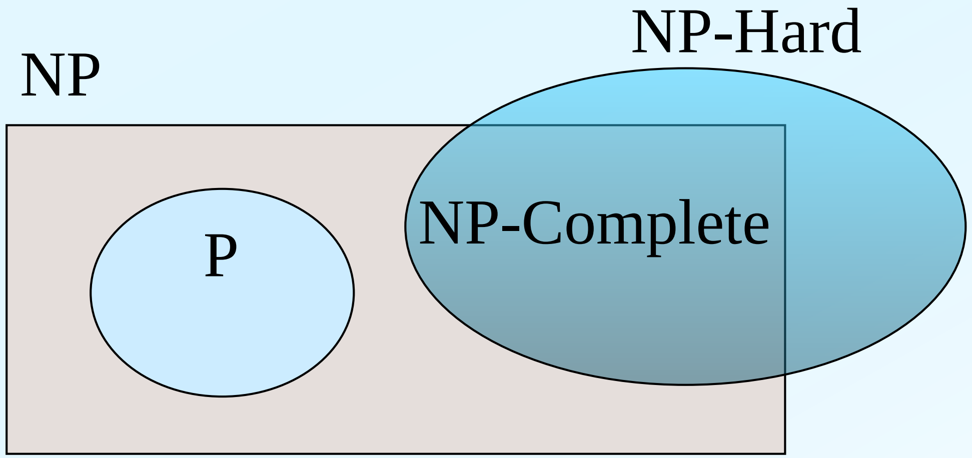
- ✓ Not determined which of (a) or (b) is right.
- ✓ Prize money $1,000,000 at stake (Clay Institute)
- ✓ Strongly conjectured to be case (a)

Relationship of NP, NP-Hard, and NP-Complete

Intuitive meaning of NP-Hard: as hard as any NP problem

NP-Complete: NP subset of NP-Hard





✓ The area of P is a conjecture

Extending NP-Hard to Optimizaztion Problems

Extending NP-Hard to Optimizaztion Problems

- By definition, NP-Complete is restricted to <u>decision problems</u>
  (Yes/No problems)
- But, NP-Hard is not restricted to decision problems

[Reminder] L is NP-Hard if every NP problem $\leq_p L$

(An NP-Hard problem need not to be NP;

thus, it need not to be a decision problem)

Redefinition of Poly-Time Reduction

Definition (Yes/No version) : Transform any instance α of problem A to a matching instance β of problem B satisfying the following:

1. The transformation is done in polynomial time (easy)
2. The answers of the two instances are the same

Extended Definition : …

1. …
2. We can get α's answer using β's solution

the same as the above

오히려 이것이 poly-time reduction의 본질에 더 가깝다

Optimization Version of TSP

Known to be NP-Hard

[Reminder] TSP (Yes/No version)

Input: undirected (positive) weighted complete graph G = (V, E)
positive number K

Question: Is there a Hamiltonian cycle of length K or less in G?

NP-Hard since TSP $\leq_p$ OPT-TSP

[OPT-TSP]

Input: undirected (positive) weighted complete graph G = (V, E)

Question: What is the length of the shortest Hamiltonian cycle in G?

OPT-TSP is NP-Hard

Claim: TSP $\leq_p$ OPT-TSP

Proof:

i) Use the instance α of TSP as it is. ($\alpha \equiv \beta$. zero transformation time)

ii) Solve OPT-TSP with α ($\equiv \beta$) to the optimum. (get the shortest Ham cycle length M)

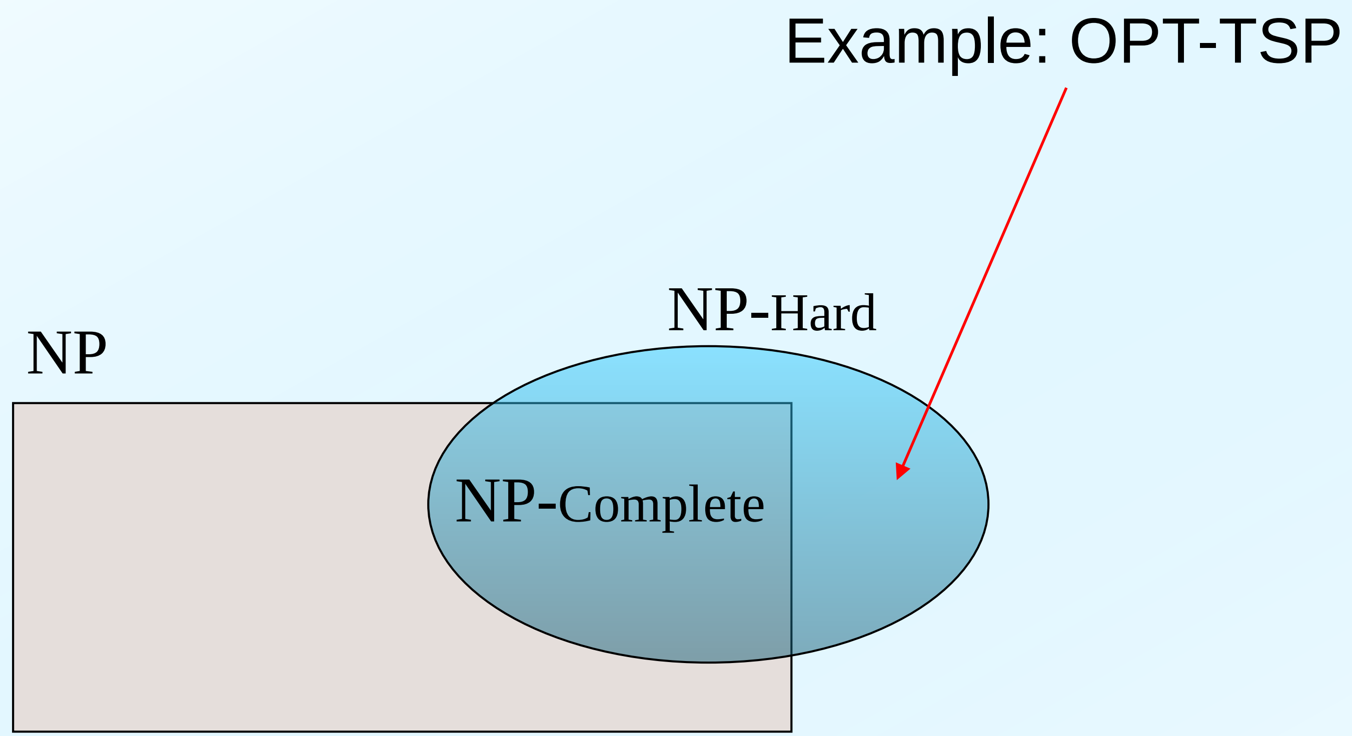
iii) $\begin{cases} \text{If } M \leq K \rightarrow \text{answer to } \alpha \text{ is Yes} \\ \text{Otherwise} \rightarrow \text{answer to } \alpha \text{ is No} \end{cases}$

The above means that we can get α's answer using β's solution.

$\therefore$ TSP $\leq_p$ OPT-TSP ∎

- ✓ Therefore, OPT-TSP is NP-Hard
- ✓ Usually, a Yes/No problem is closely related this way to the corresponding optimization problem.

It is Possible to be not NP but NP-Hard



Approximation

with OPT-TSP as an Example

Approximation

- When a problem is proved to be NP-Complete/Hard
  - Stop to develop an easy algorithm
  - Develop a heuristic algorithm that tries to find a suboptimal solution in the given time budget

Ratio bound $\rho(n)$ of an algorithm for a minimization problem:

$$\frac{C}{C^*} \leq \rho(n)$$

where n: the problem size

C^*: the optimal solution cost

C : the cost of a solution produced by the algorithm

OPT-TSP w/ Triangle Inequality (Metric TSP)

$w_{ij} \leq w_{ik} + w_{kj}$, $\forall$vertices i, j, k

1. NN (Nearest Neighbor Algorithm)
   - Start at a random vertex, keep visiting the nearest unvisited neighbor
   - [Thm] The ratio bound $\rho(n)$ for NN

     $\rho(n) = \frac{1}{2}(\lceil \log_2 n \rceil + 1)$ for all instances

     $\frac{C}{C^*} > \frac{1}{3}(\log_2(n+1) + \frac{4}{3})$ for some large instances

   - Guarantees almost nothing (just for theoretical interest. Practically, little attraction.)

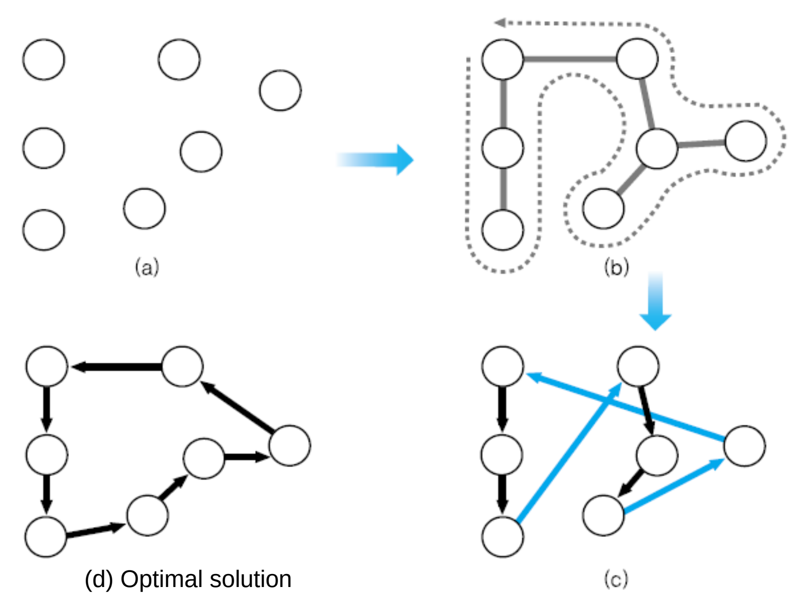
2. MST (Minimum Spanning Tree Algorithm)

- Construct a min. sp. tree T starting at a random vertex
- Return the Ham. Cycle H that visits the vertices in the order of a preorder traverse of T
- [Thm] The ratio bound $\rho(n)$ for MST

$$\rho(n) = 2$$

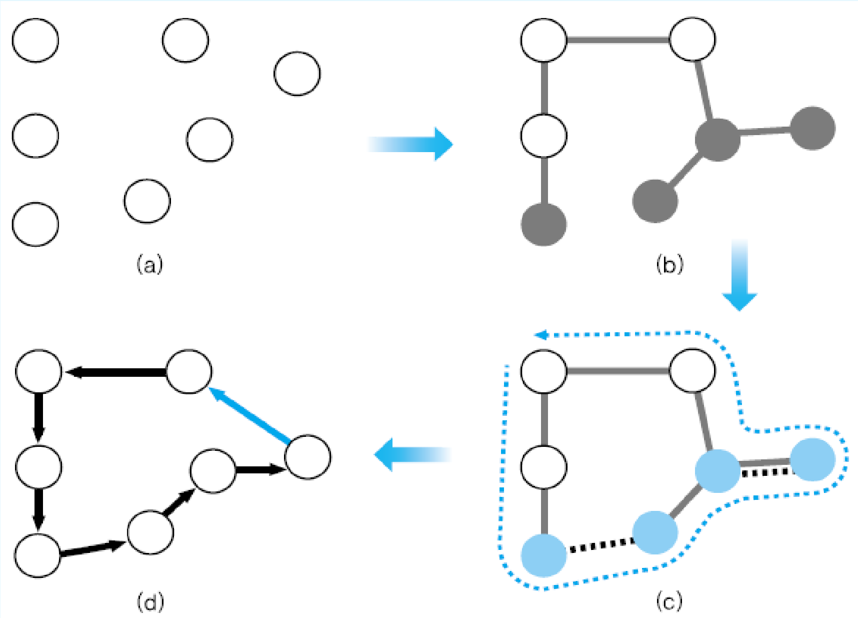
<Proof>

$$C = \text{cost}(H) \leq 2\text{cost}(T) \leq 2C^*$$



3. MM (Minimum Matching Algorithm)

- Find a min. sp. tree T starting at a random vertex
- Let V' be the set of odd-degree vertices in T
- Find a matching of V' which has maximum cardinality and minimum weight, say M
- Add M to T (to get an Eulerian cycle C_1 of T')
- Convert C_1 into a TSP tour C_2 (a Ham. cycle) by using short cuts.



- [Thm] The ratio bound $\rho(n)$ for MM

$$\rho(n) = \frac{3}{2}$$

<Proof>

Note: $\text{cost}(T) \leq C^*$

$\text{cost}(C_1) = \text{cost}(T) + \text{cost}(M)$

Claim: $\text{cost}(M) \leq \frac{1}{2} C^*$

<Proof>

An optimal tour C_{opt} (cost C^*) can be changed to a tour C'_{opt} with only vertices in V' by using short cuts.

$\text{cost}(C'_{opt}) \leq C^*$.

Take alternate edges in C'_{opt} and then we have two matchings of V'.

Then the smaller of the two matchings must have $\text{cost} \leq \frac{1}{2} cost(C'_{opt})$.

Since M is the min. weight matching of V',

$$\begin{aligned} \text{cost}(M) &\leq \text{the smaller matching above} \\ &\leq \frac{1}{2} cost(C'_{opt}) \\ &\leq \frac{1}{2} C^* \end{aligned}$$

Therefore, $\text{cost}(C_2) \leq \text{cost}(C_1) = \text{cost}(T) + \text{cost}(M) \leq \frac{3}{2} C^*$

OPT-TSP without Triangle Inequality

true even for any ratio bound function $\rho(n)$

Theorem: If P≠NP, there exists no poly-time approximation algorithm with a constant ratio bound ρ

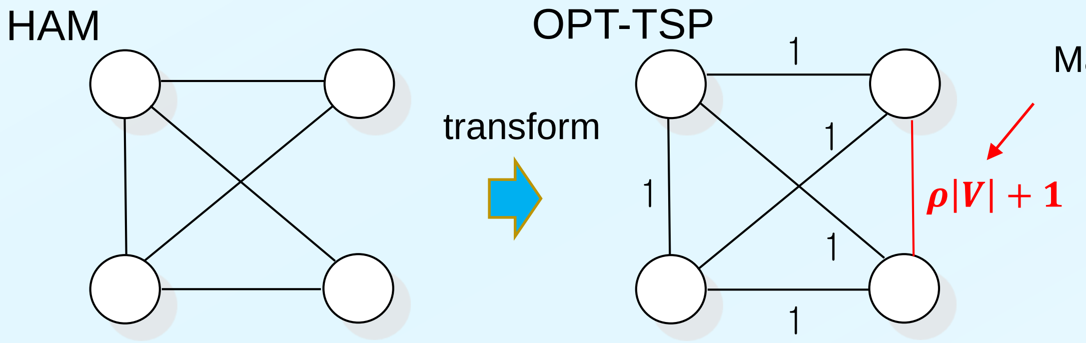
<Proof>

Show that if there is a poly-time approximation algorithm A with a ratio bound ρ, then we show that the Hamiltonian cycle problem is in P.

Given a HAM instance *G*=(*V*,*E*), we construct an instance of TSP *G*'=(*V*,*E*') as follows:

$$w_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \\ \rho|V| + 1 & \text{otherwise} \end{cases}$$

$\rho \geq 1$



An instance α of Hamiltonian cycle

Matching instance β of OPT-TSP

Make a complete graph by:

assigning weight $\begin{cases} 1 \text{ to the edges in } \alpha \\ \boldsymbol{\rho|V| + 1} \text{ to the edges not in } \alpha \end{cases}$

This is a poly-time transformation.

Run the (poly-time) algorithm *A* on *G*';

if *A* returns a tour of length $\leq \rho|V|$, then there exists a Ham cycle in *G*,

otherwise, there is no Ham cycle in G.

Thus, if there is such a poly-time approximation for OPT-TSP,
HAM can be solved in poly time. Therefore NP=P. ∎

✓ We say this type of technique as 'amplification.'

이론적 근사 알고리즘의 실용성은 낮다

OPT-TSP에 대해 ratio bound 1.5가 실용적 의미가 있는가?

TABLE II
EXPERIMENTAL RESULTS UNDER THE MIXED FRAMEWORK

Graph	EA	Best	Ave(%)	$\sigma/\sqrt{n}$	Gen(CPU)
lin318 (42029)	DGLS	**42029**	**42029.00**(0.000)	0.00	791(27)
	NGLS	**42029**	**42029.00**(0.000)	0.00	388(20)
att532 (27686)	DGLS	**27686**	27692.64(0.024)	0.72	3984(86)
	NGLS	**27686**	**27690.01**(0.014)	0.72	4270(171)
dsj1000 (18659688)	DGLS	**18659688**	18660177(0.003)	115	5190(1075)
	NGLS	**18659688**	**18659894**(0.001)	18	4898(1331)
d2103 (80450)	DGLS	**80450**	**80471.33**(0.027)	4.45	3008(512)
	NGLS	**80450**	80480.51(0.038)	5.32	2520(448)
pcb3038 (137694)	DGLS	**137698**	137792.25(0.071)	5.91	18246(830)
	NGLS	137699	**137770.08**(0.055)	5.68	27371(1504)
fnl4461 (182566)	DGLS	182620	182742.53(0.097)	5.79	69300(2906)
	NGLS	**182602**	**182694.65**(0.070)	4.36	99740(6421)
rl11849 (923288)	DGLS	923786	924371.90(0.117)	74.19	95507(18542)
	NGLS	**923656**	**924072.60**(0.085)	58.64	123614(36631)

이론적 보장은 없지만 genetic algorithm(GA)으로 찾은 OPT-TSP의 솔루션 품질
Optimal solution을 찾거나 평균 0.1% 미만으로 근접한다

Co-NP

Co-NP

Co-NP: set of problems L s.t. $\overline{L} \in$ NP   Equivalently, $L \in$ Co-NP if $\overline{L} \in$ NP

In the universe(set) of all unweighted undirected graphs

HAM = {graph G: G has a Hamiltonian cycle}

$\overline{\mathrm{HAM}}$ = {graph G: G doesn't have a Hamiltonian cycle}

$\mathrm{HAM} \in$ NP

$\overline{\mathrm{HAM}} \in$ Co-NP ← $\overline{\overline{\mathrm{HAM}}} = \mathrm{HAM} \in$ NP

Equivalently,
given a certificate of $\overline{\overline{\mathrm{HAM}}}$(=HAM) (a Ham cycle of a graph $G \in$ HAM),
we can verify that the certificate is <u>not in $\overline{\mathrm{HAM}}$</u> (in HAM) in poly time.

If NP ≠ Co-NP, then P ≠ NP

Equivalently, if P = NP, then NP = Co-NP

<Proof>

If P = NP, then NP = P = Co-P = Co-NP.

Hence, NP = Co-NP.

Equivalently, if NP ≠ Co-NP, then P ≠ NP.

Co-P: set of problems L s.t. $\bar{L} \in$ P

Obvious that P = Co-P

✓ "NP = Co-NP?" is also a million-dollar problem