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

문제의 종류

풀 수 없는 문제들

(Unsolvable)

(Undecidable)

정지 문제

힐버트의 10번째 문제

…

여기에 속할 것이라고
강력히 추정!

Presburger 산술

…

NP-완비

현실적인 시간내에
풀 수 없는 문제들

풀 수 있는 문제들

(Solvable)

(Decidable)

최소 신장 트리 문제

최단 거리 문제

…

현실적인 시간내에
풀 수 있는 문제들

현실적인 시간

- 다항식 시간을 의미
  - 입력의 크기 n의 다항식으로 표시되는 시간
  - 예: $3n^k + 5n^{k-1} + \ldots$
- 비다항식 시간의 예
  - 지수 시간
    - 예: 2^n
  - 계승시간
    - 예: $n!$

Yes/No 문제와 최적화 문제

- Yes/No 문제 (Decision problem)
  - 예: 그래프 G에서 길이가 k 이하인 해밀토니안 경로가 존재하는가?
- 최적화 문제 (Optimization problem)
  - 예: 그래프 G에서 길이가 가장 짧은 해밀토니안 경로는 얼마인가?

✓ 두 문제는 동전의 앞뒷면

NP-Complete 이론

- Yes/No 의 대답을 요구하는 문제에 국한
  - 그렇지만 최적화 문제와 밀접한 관계를 갖고 있다
- 문제를 현실적인 시간에 풀 수 있는가에 관한 이론
- 거대한 군을 이룸
  - 이 중 한 문제만 현실적인 시간에 풀면 다른 모든 것도 저절로 풀리는 논리적 연결관계를 갖고 있다

현재까지의 연구결과

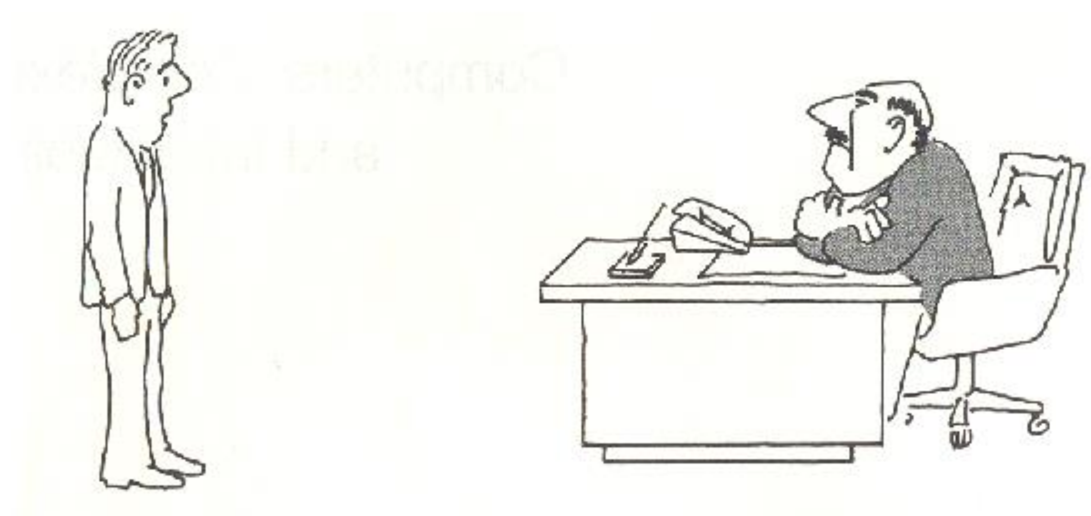
- 어떤 문제가 NP-Complete/NP-Hard임이 확인되면

⇨ 지금까지의 연구결과로는 이 문제를 현실적인 시간에 풀 수 있는 방법은 아직 없다

- 그렇지만 이 사실이 아직 증명은 되지 않음
- 클레이수학연구소의 21세기 7대 백만불짜리 문제 중의 하나
  - P=NP 문제

NP-Complete에 관한 비유

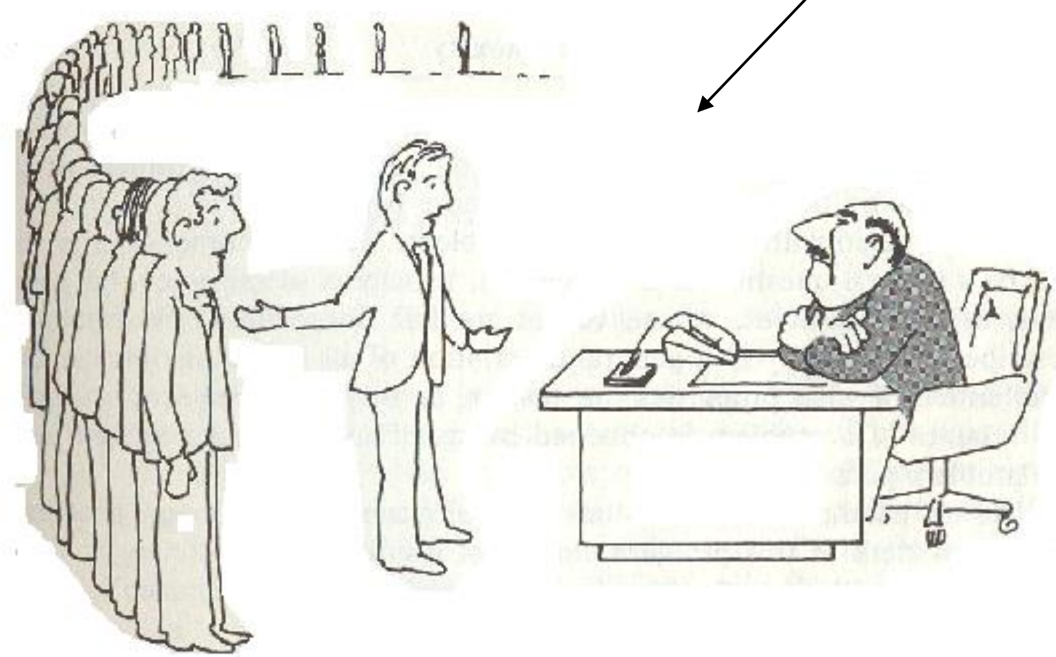
상사가 아주 어려운 문제를 해결하라고 지시했다



1. 나는 답을 구할 수가 없습니다. 아마 나는 멍청한 모양입니다.

2. 나는 답을 구할 수가 없습니다. 왜냐하면 이 문제는 답이 없어요.

NP-Complete 이론의 상황을 비유적으로 보여줌



3. 나는 답을 구할 수가 없습니다. 그렇지만 저렇게 수많은 천재들도
이 문제를 해결할 수 없습니다.

준비: 논리적 얼개

쉽다 = 현실적인 시간에 풀 수 있다

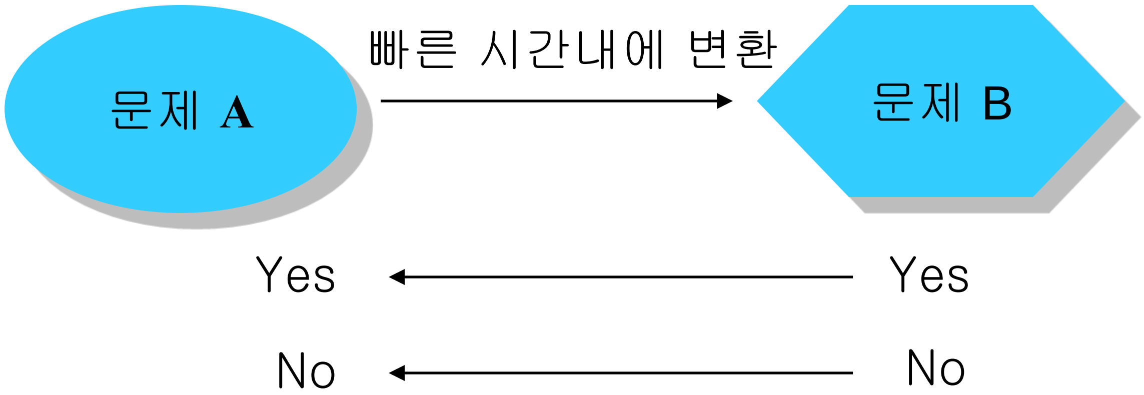
문제1: 정수 $x=x_1x_2\ldots x_n$은 3의 배수인가?

문제2: $x_1+x_2+\ldots+x_n$은 3의 배수인가?

- ✓ 위 두 문제의 대답은 같다
  - ➢ Yes/No 대답이 일치한다
- ✓ 문제 2가 쉬우면, 문제 1도 쉽다

- 상황
  - 문제 B는 쉽다
  - 문제 A는 Yes/No 대답이 일치하는 문제 B로 쉽게 변형된다

쉽다 = 현실적인 시간에 풀 수 있다



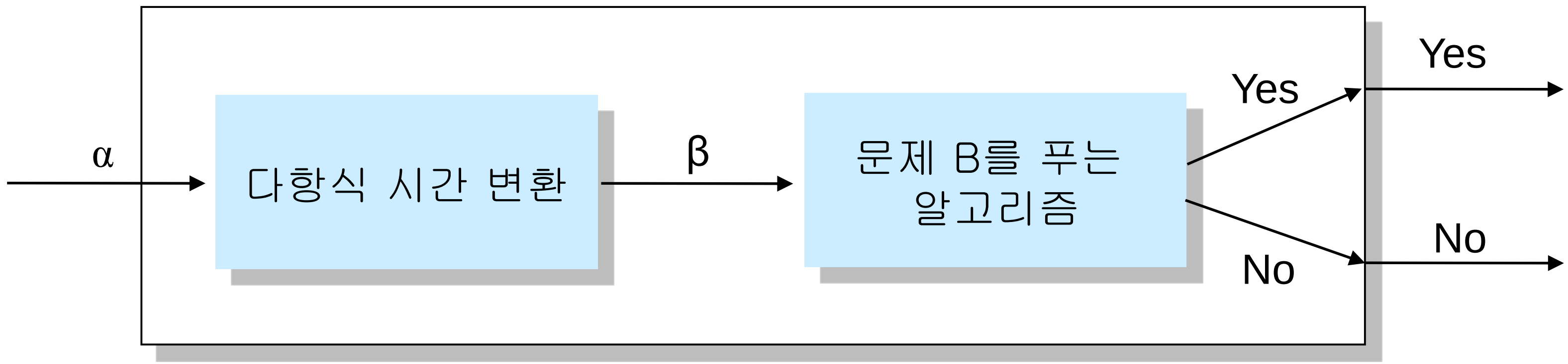
✓ 문제 A도 쉬운가?

Poly-Time Reduction (다항식 시간 변환)

- 문제 A의 사례 α를 문제 B의 사례 β로 바꾸되 아래 성질을 만족하면 polynomial-time reduction이라 하고, 이를 $\alpha \leq_p \beta$ 로 표기한다

① 변환은 다항식 시간에 이루어진다

② 두 사례의 답은 일치한다



문제 A를 푸는 알고리즘

1. 문제 A를 다항식 시간에 문제 B로 변환한다
2. 변환된 문제 B를 푼다
3. 문제 B의 대답이 Yes이면 Yes, No이면 No를 리턴한다

✓ 문제 B가 쉬운 문제라면 문제 A도 쉬운 문제이다

P와 NP

- P
  - Polynomial
  - 다항식 시간에 Yes 또는 No 대답을 할 수 있으면 P
- NP
  - Nondeterministic Polynomial
  - Non-Polynomial의 준말이 아님!
  - Yes 대답이 나오는 해를 제공했을 때, 이것이 Yes 대답을 내는 해라는 사실을 다항식 시간에 확인해줄 수 있으면 NP
- 어떤 문제가 NP임을 보이는 것은 대부분 아주 쉽다
  - NP-Complete 증명에서 형식적으로 확인하고 넘어가는 정도

NP-Complete/Hard

NP : Yes 대답이 나오는 해를 제공하면 이를 다항식 시간에 확인할 수 있으면 됨

[정의] 다음 성질을 만족하면 문제 L은 NP-Hard이다

- 모든 NP 문제가 L로 다항식 시간에 변환가능하다
  (즉 "모든 NP문제 $\leq_p L$")

[정의] 다음의 두 성질을 만족하면 문제 L은 NP-Complete이다

1) L은 NP이다.
2) L은 NP-Hard이다.

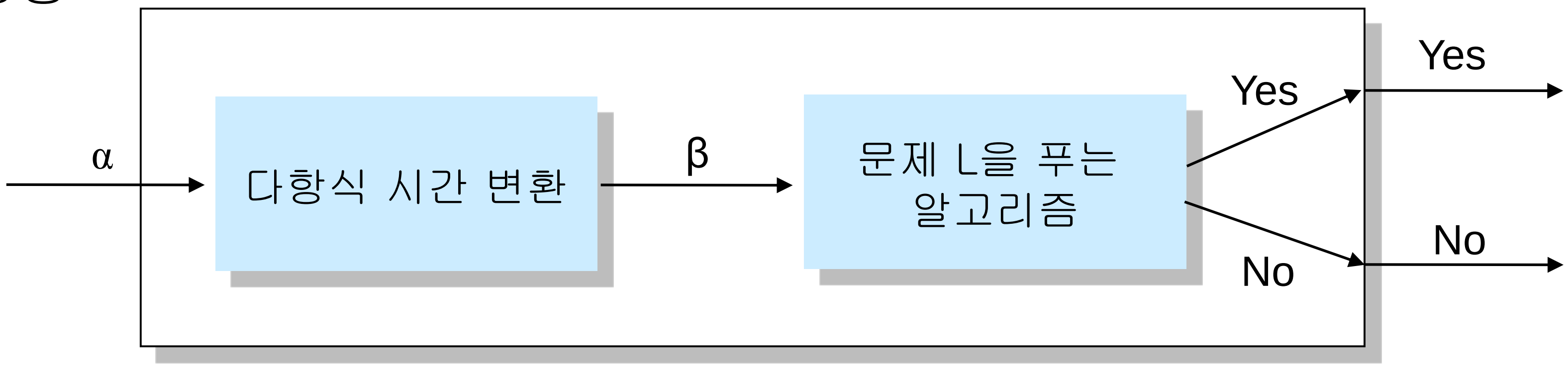
✓ NP-Complete는 NP-Hard의 일부이므로 NP-Complete인 문제를 NP-Hard이라고 불러도 맞다

✓ NP-Complete의 성질 1)은 대부분 자명하므로 핵심에 집중하기 위해 NP-Hard에 초점을 맞추자

[정리] 문제 L이 다음의 성질을 만족해도 NP-hard이다

알려진 임의의 NP- Hard 문제 A로부터 문제 L로 다항식 시간에 변환가능하다. (즉 $A \le_p L$)

<증명>



문제 A를 푸는 알고리즘

이것은 $A \le_p L$임을 뜻한다.

주어진 조건이 “모든 NP문제 $\le_p A$”이므로, 모든 NP문제 $\le_p L$이 된다.

(by transitivity) ■

✓ 만일 문제 L이 쉬운 문제라면 문제 A도 쉬운 문제이다

➔ 그러므로 모든 NP 문제도 쉬운 문제이다

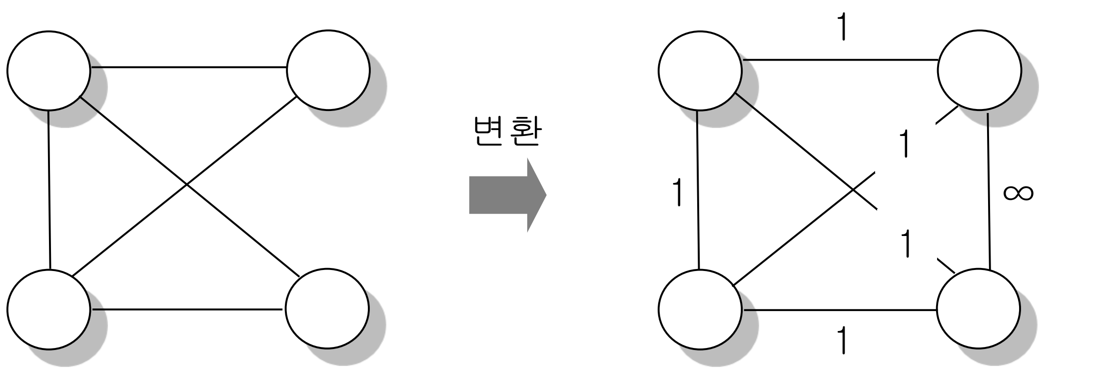
NP-Hard 증명의 예

- 해밀토니안 싸이클 문제는 NP-Hard이다
- 이를 이용해서 TSP 문제가 NP-Hard임을 보일 수 있다

- 해밀토니안 싸이클Hamiltonian cycle
  - 그래프의 모든 정점을 단 한번씩 방문하고 돌아오는 경로
- 해밀토니안 싸이클 문제
  - 주어진 그래프에서 해밀토니안 싸이클이 존재하는가?

- TSP 문제
  - 주어진 완전 그래프complete graph에서 길이 k 이하인 해밀토니안 싸이클이 존재하는가?

- 해밀토니안 싸이클 문제의 instance(사례) α를 아래와 같이 TSP 문제의 instance β로 다항식 시간에 변환한다



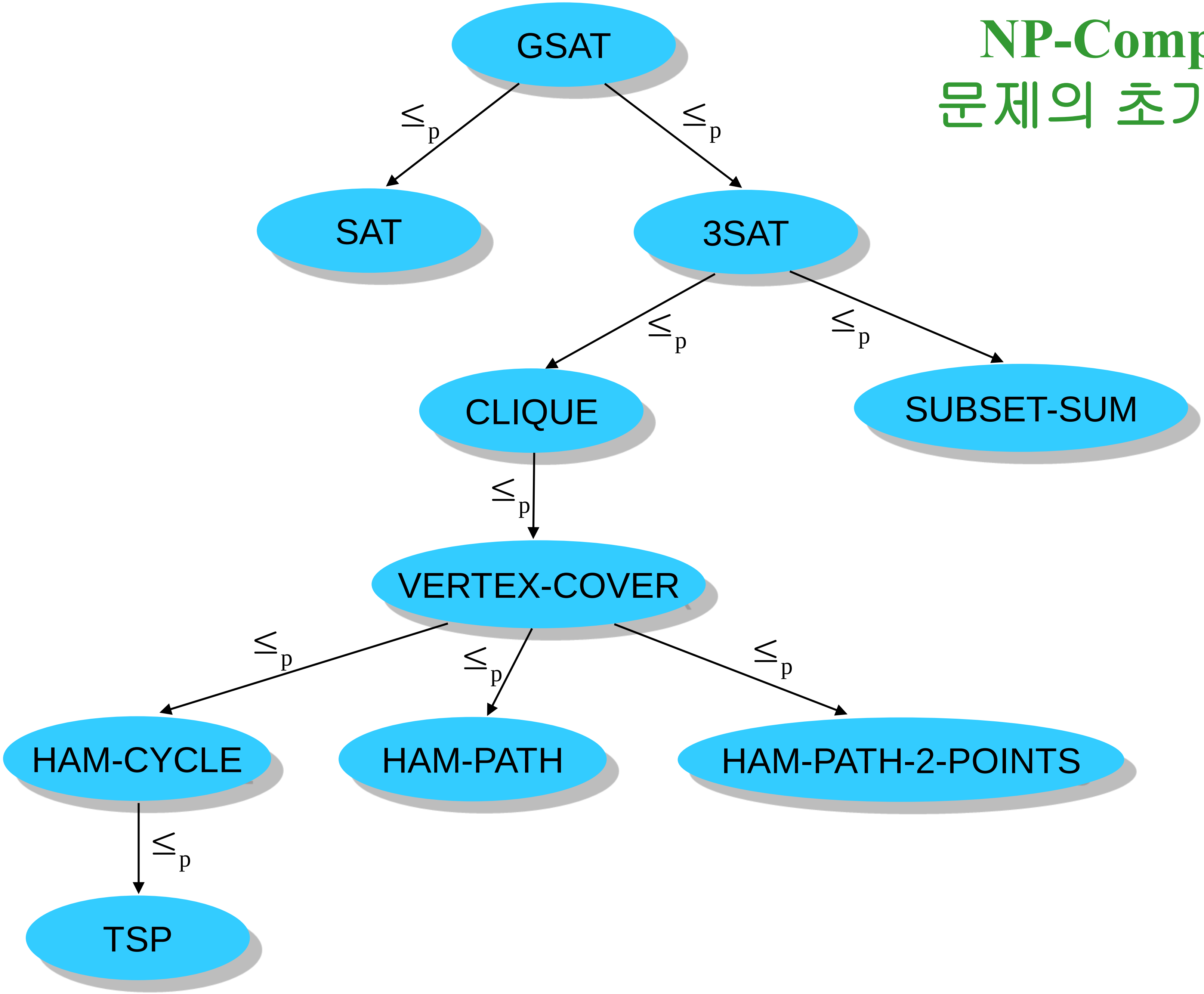
해밀토니안 싸이클 문제의 instance α TSP 문제의 instance β

Instance α가 해밀토니안 싸이클을 갖는다
⇔ Instance β가 길이 n 이하인 해밀토니안 싸이클을 갖는다
(이 예에서 n은 4)

➢ 그러므로 TSP는 NP-Hard이다. ■

Vertex 수와 일치

NP-Complete 문제의 초기 역사



직관과 배치되는 NP-Complete 문제의 예

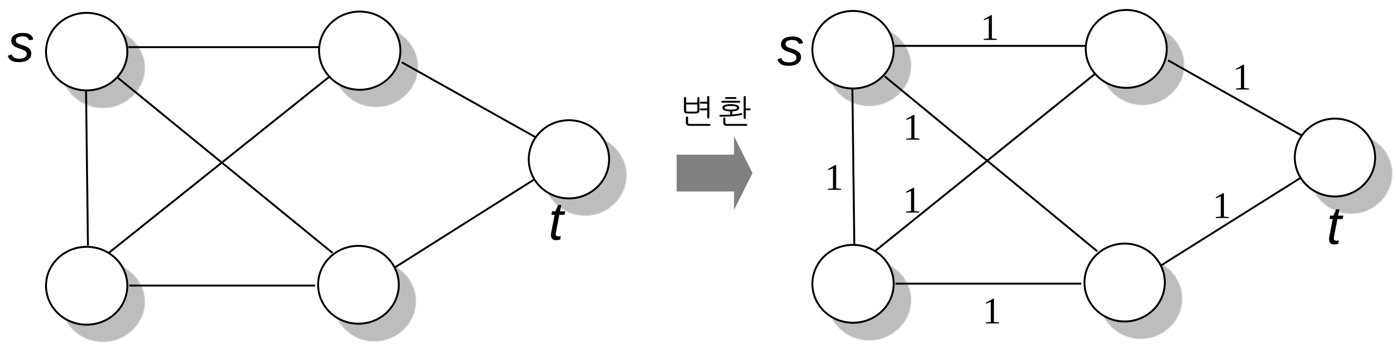
- Shortest path
  - 그래프의 정점 s에서 t로 가는 shortest path는 간단히 구할 수 있다
- Longest path
  - 그래프의 정점 s에서 t로 가는 longest path는 간단히 구할 수 없다
  - NP-Hard

✓ 얼핏 비슷해 보이지만 위 두 문제의 난이도는 천지차이다! (지금까지의 연구 결과로는)

- Longest Path 문제
  - (최적화 버전) 주어진 그래프에서 vertex s에서 t로 가는 가장 긴 simple path의 길이는 얼마인가?
  - (Yes/No 버전) 주어진 그래프에서 vertex s에서 t로 가는 길이 k 이상인 simple path가 존재하는가?

- 두 점 사이 해밀토니안 경로 문제
  - 주어진 그래프에서 정점 s에서 t에 이르는 해밀토니안 경로가 존재하는가?
  - NP-Complete라고 알려져 있음

- 두 점 사이 해밀토니안 경로 문제의 instance α로부터 Longest Path 문제의 instance β로 다항식 시간 변환



HAM−PATH−2−POINTS 문제의 instance α

LONGEST−PATH 문제의 instance β

Instance α가 두 점 s와 t사이에 해밀토니안 경로를 갖는다

⇔ Instance β가 두 점 s와 t 사이에 길이 n-1 이상인 (사실은 정확히 n-1) simple path를 갖는다 (위 예에서 n-1은 4)

➢ 그러므로 Longest Path 문제는 NP-Hard이다. ■

핵심에 집중하기 위해 성질 1(Longest Path는 NP)은 일부러 누락.
추가로 성질 1을 증명해서 NP-Complete임을 보이는 것은 매우 간단하다.

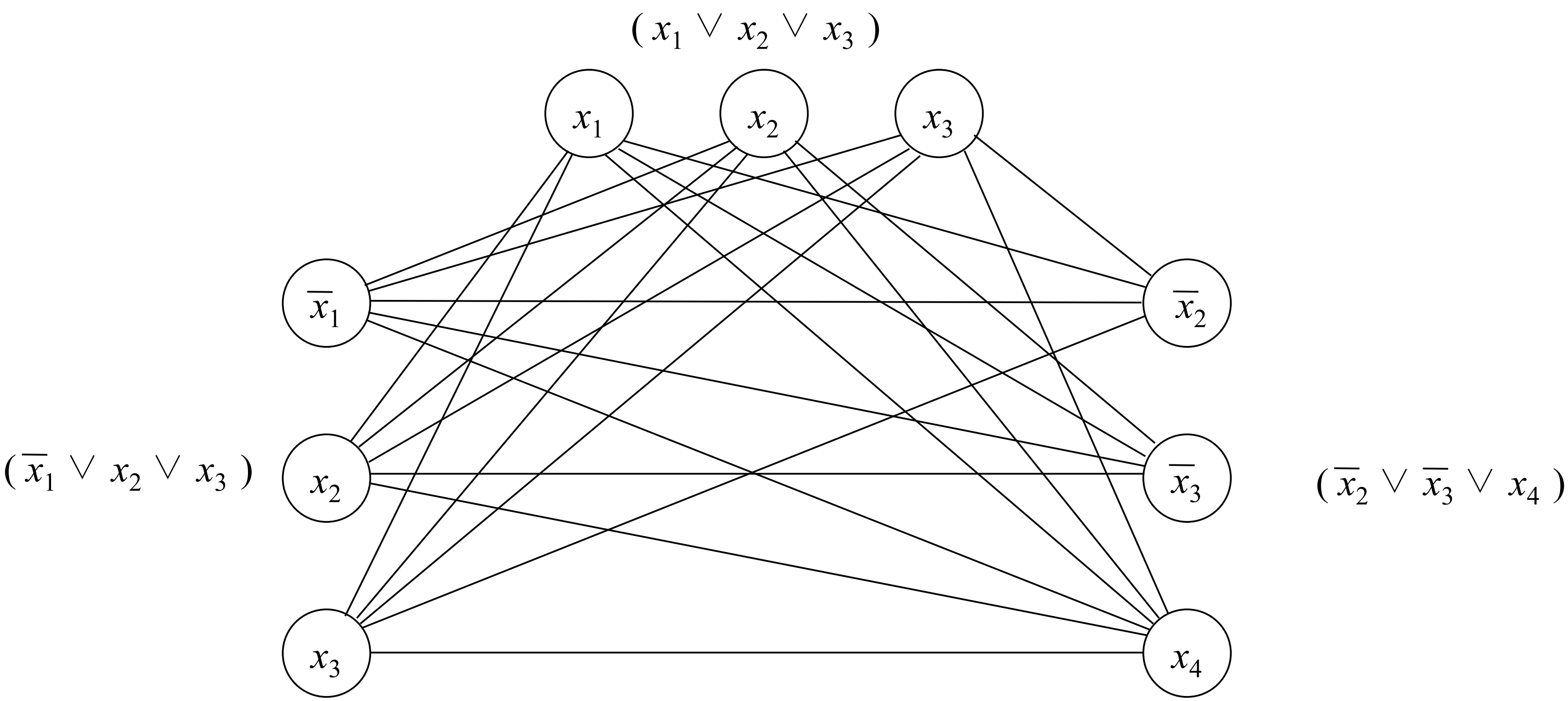
CLIQUE

- 입력
  - 그래프 $G = (V, E)$, 양의 정수 K
- 질문
  - 그래프 G에 크기 K인 complete subgraph가 존재하는가?

- CLIQUE은 NP-Complete이다

3SAT $\leq_p$ CLIQUE임을 증명하기 위한 그림

예: Boolean formula $(x_1 \vee x_2 \vee x_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4)$이 true가 되는 x_i들 값 할당이 존재하는가?

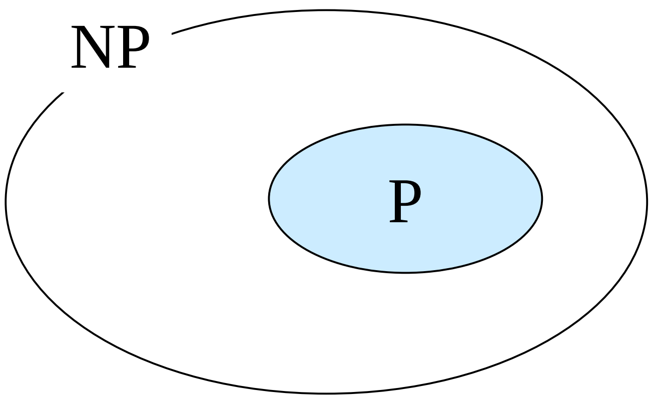


NP 이론의 유용성

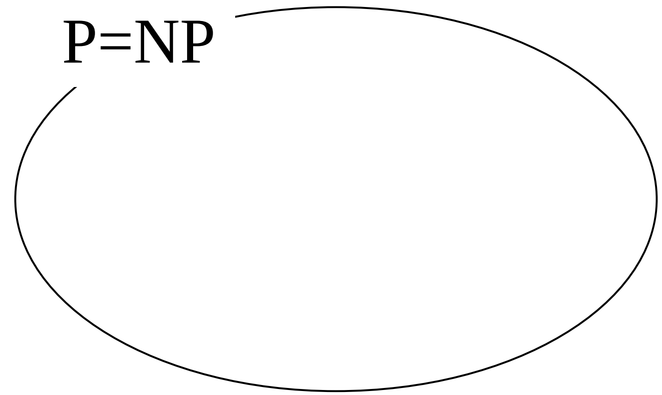
- 어떤 문제가 NP-Complete/Hard임이 확인되면
  - ⇨ 쉬운 알고리즘을 찾으려는 헛된 노력은 일단 중지한다
  - ⇨ 주어진 시간 예산 내에서 최대한 좋은 해를 찾는 알고리즘 (heuristic) 개발에 집중한다

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

P와 NP의 포함 관계



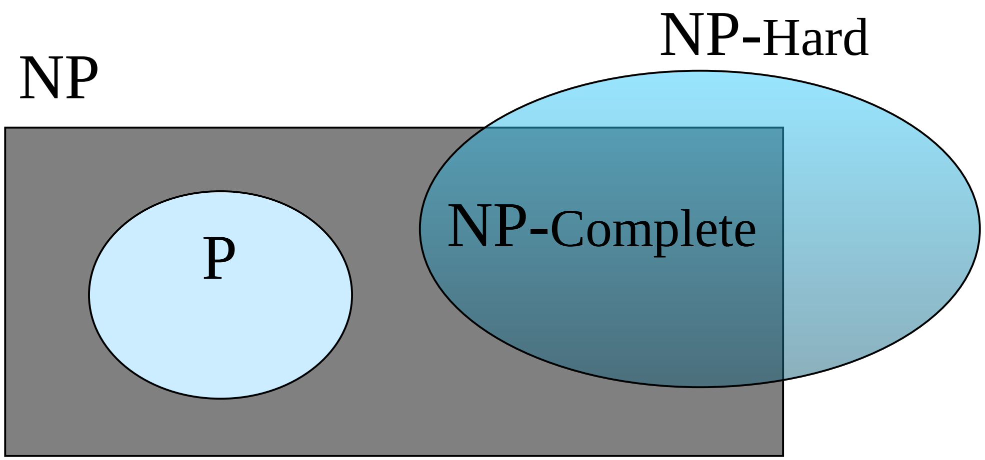
(a)



(b)

✓ 위 (a)인지 (b)인지는 아직 밝혀지지 않음.
백만불의 상금이 걸려 있다.

NP와 NP-Complete, NP-Hard의 관계



✓ P 부분은 추정

NP-Hard를 최적화 문제로 확장하기

- 정의상 NP-Complete은 decision problem(yes/no problem)에 국한된다
- NP-Hard는 그렇지 않다

- [Reminder] 모든 NP 문제가 L로 다항식 시간 변환가능하면 L은 NP-Hard다

다항식 시간 변환의 재정의

[def.] (Yes/No 버전에 맞는 정의) 문제 A의 사례 α를 문제 B의 사례 β로 바꾸되 다음 성질을 만족하면 다항식 시간 변환이라 한다

1. 변환은 다항식 시간에 이루어진다(별 심각하지 않은 조건)
2. 두 사례의 답은 일치한다

[def. 2] (확장) 문제 A의 사례 α를 문제 B의 사례 β로 바꾸되 다음 성질을 만족하면 다항식 시간 변환이라 한다

1. 변환은 다항식 시간에 이루어진다(별 심각하지 않은 조건)
2. β의 답을 이용해서 α의 답을 구할 수 있다

TSP의 최적화 버전

입력: 양의 가중치를 갖는 undirected complete graph $G = (V, E)$

질문: G에서 길이가 가장 짧은 Hamiltonian cycle의 길이는?

[Reminder] TSP (Yes/No 버전)

입력: 양의 가중치를 갖는 undirected complete graph $G = (V, E)$, 양의 실수 K

질문: G에서 길이가 K 이하인 Hamiltonian cycle이 존재하는가?

최적화 TSP는 NP-Hard다

알려진 NP-Hard 문제 하나로부터 최적화 TSP로 변환가능하면 최적화 TSP는 NP-Hard.
알려진 NP-Hard 문제인 Yes/No 버전 TSP로부터 변환.

Yes/No 버전 TSP의 사례를 그대로 최적화 TSP의 사례로 사용
(변환 시간 제로)

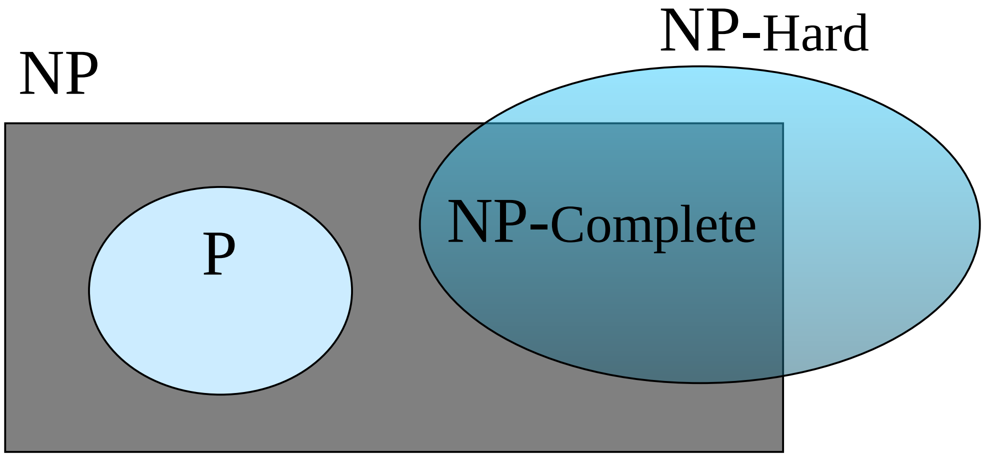
질문 "*G*에서 길이가 *K* 이하인 Hamiltonian cycle이 존재하는가?"는 최적화 TSP를 해결하고 나면 저절로 풀린다.

  도출된 최적해의 길이가 *K* 이하 → 대답은 Yes
  Otherwise, 대답은 No

최적화 TSP의 답을 이용해서 Yes/No 버전 TSP의 답을 구할 수 있다.
∴최적화 TSP는 NP-Hard

✓ 대체로 이렇다. Yes/No 문제와 최적화 문제는 동전의 앞뒷면 같다.

NP가 아니면서 NP-Hard일 수 있다



✓ P 부분은 추정

Approximation:
with TSP as an Example

Approximation

- 어떤 문제가 NP-Complete/Hard임이 확인되면
  - ⇨ 쉬운 알고리즘을 찾으려는 헛된 노력은 일단 중지한다
  - ⇨ 주어진 시간 예산 내에서 최대한 좋은 해를 찾는 알고리즘 (heuristic) 개발에 집중한다
  - ⇨ approximation
- 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

TSP w/ Triangle Inequality (Metric TSP)

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

1. NN (Nearest Neighbor Algorithm)
   - Start ar 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 (이론적 흥미. 실용성 낮음.)

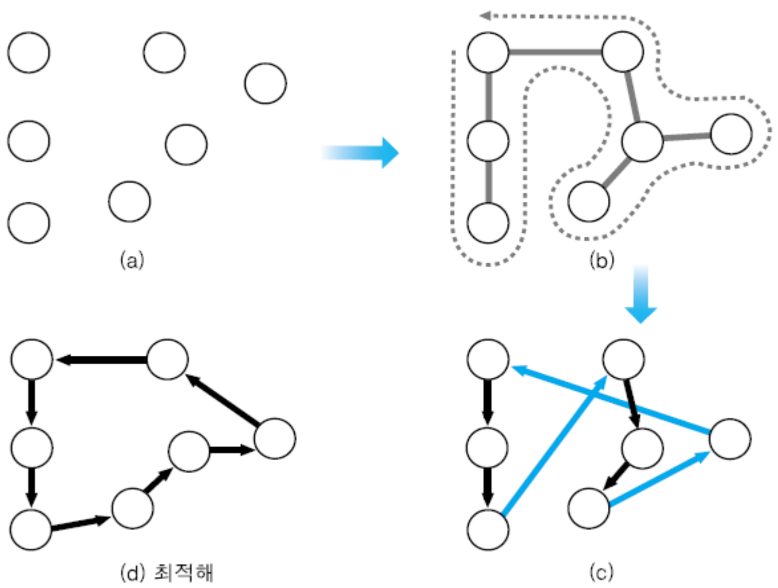
2. MST (Minimum Spanning Tree Algorithm)

- 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

$$\rho(n) = 2$$

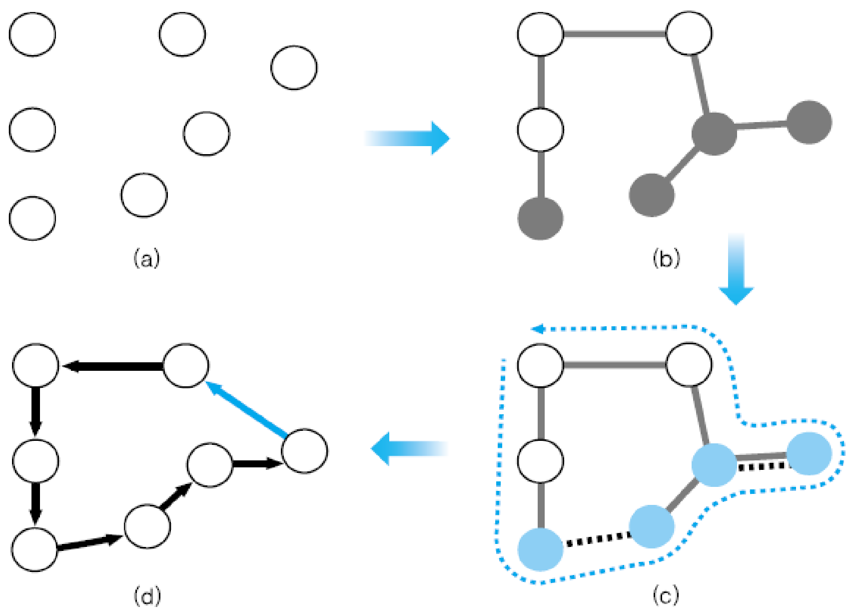
<Proof>

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



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 graph 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: $\text{cost}(T) \le C^*$

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

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

<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.

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

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

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

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

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

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

TSP w/o Triangle Inequality

[Thm] P≠NP라면, 어떠한 상수 ρ에 대해서도 최적해의 ρ배를 넘지 않는 해를 보장하는 poly-time 근사 알고리즘은 존재하지 않는다

<Proof>

Show that if there is a poly-time approximation algorithm *A* with a ratio bound ρ, then 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:

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

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 ∃ a Ham cycle in *G*,
otherwise ∃ no Ham cycle in *G*.

Thus HAM can be solved in poly time. Therefore NP=P.

Conclusion: If P≠NP, there is no such poly-time approximation algorithm.

Co-NP

an instance α

HAM

NP: $x \in L$이면 **nondeterministic poly-time**에 **Yes**라고 대답할 수 있는 문제 L들의 집합

Co-NP: $\overline{L} \in$ **NP**인 문제 L들의 집합

$\overline{\text{HAM}}$

P: $x \in L$ 인지 **deterministic poly-time**에 **Yes** 또는 **No**라고 대답할 수 있는 문제 L들의 집합

Co-P: $\overline{L} \in$ **P**인 문제 L들의 집합

→ ∴ **P=Co-P** (trivially)

NP≠Co-NP이면 P≠NP이다

Equally, P=NP이면 NP=Co-NP이다

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

Hence, NP = Co-NP.