Chapter 7
NP-완전 문제

차례

7.1 문제의 분류

- 다항식 시간 복잡도를 가진 알고리즘으로 해결되는 P(polynomial) 문제 집합
  - 시간 복잡도가 O(logn), O(n), O(nlogn), $O(n^2)$, $O(n^3)$ 등
  - 이러한 시간 복잡도는 점근적 표기법에 따르면 $O(n^k)$에 포함되기 때문. 단, k는 양의 상수.

문제의 분류

- 다항식 시간보다 큰 복잡도를 가진 알고리즘으로 해결되는 문제의 집합
  - 여러 가지 문제 집합으로 다시 분류
  - 그 중에 가장 중요한 문제 집합은 지수 시간 (exponential time) 시간 복잡도를 가진 알고리즘으로 해결되는 NP-완전 문제 집합이다.

- NP-완전 문제의 특성
  - 어느 하나의 NP-완전 문제에 대해서 다항식 시간의 알고리즘을 찾아내면 (즉, 다항식 시간에 해를 찾을 수 있으면) 모든 다른 NP-완전 문제도 다항식 시간에 해를 찾을 수 있다.

NP 문제 집합

NP 문제 집합

- P 문제 집합과 NP-완전 문제 집합을 둘 다 포함하는 문제의 집합
- NP 문제 집합에 속한 문제를 NP 문제라고 한다.
- NP 문제는 비결정적 다항식 시간(Nondeterministic Polynomial time) 알고리즘을 가진 문제이다.

NP 알고리즘

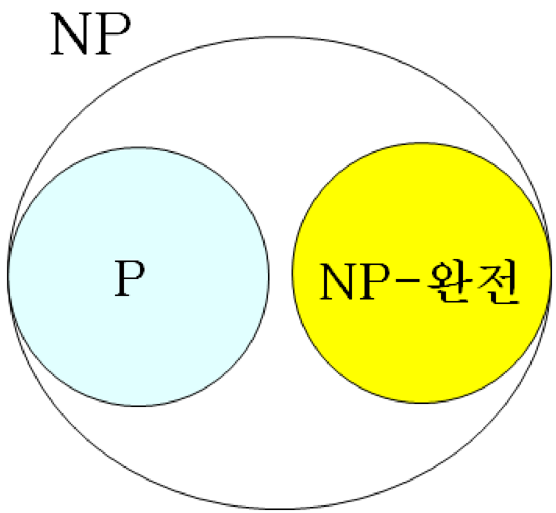
- 비결정적 다항식 시간 알고리즘
- 첫 번째 단계
  - 주어진 입력에 대해서 하나의 해를 추측하고
- 두 번째 단계
  - 그 해를 다항식 시간에 확인한 후에
  - 그 해가 '맞다/아니다'라고 답한다.

NP 알고리즘

- NP 알고리즘
  - 해를 찾는 알고리즘이 아니라,
  - 해를 다항식 시간에 확인하는 알고리즘이다.

문제의 관계

- P 문제, NP-완전 문제, NP 문제 집합 사이의 관계



[주의] 위 그림의 문제 포함 관계는 아직 증명되지 못했으나 대부분의 학자들이 맞을 것이라고 생각한다.

P 문제 집합이 NP 문제 집합에 속하는 이유

- P 문제를 해결하는데 다항식 시간이 걸리므로 이를 NP 알고리즘이 문제의 해를 다항식 시간에 확인하는 것과 대응시킬 수 있기 때문
- P 문제를 위한 NP 알고리즘은 해를 추측하는 단계를 생략하고, 해를 확인하는 단계 대신에 해를 직접 다항식 시간에 구하고 확인 결과를 '맞다'라고 답한다.

문제의 변형

- NP 알고리즘은 추측한 해를 확인하여 '맞다/아니다'라고 답하므로, 문제의 해가 'yes' 또는 'no'가 되도록 주어진 문제를 변형시켜야 한다.
- 이러한 유형의 문제를 결정(decision) 문제라고 한다.
- 여행자 문제 (TSP: Traveling Salesperson Problem)
  - 각 도시를 한 번씩만 방문하고 시작 도시로 돌아오는 최단 경로의 거리를 찾는 문제
  - 각 도시를 1번씩만 방문하고 시작 도시로 돌아오는 경로의 거리가 K보다 짧은 경로가 있는가?

NP 알고리즘의 예

- 8개 도시 (A B C D E F G H)에 대한 여행자 문제의 NP 알고리즘은 다음과 같다. 단, A는 시작 도시이다.
- 8개 도시 (A B C D E F G H)의 여행자 문제의 하나의 해를 추측한다. 예를 들어, A G D H F E B C를 추측했다면
- 추측한 해, 즉, 경로의 거리를 다음과 같이 계산한다.

경로의 거리 = (A와 G 사이의 거리)
+ (G와 D 사이의 거리)
+ (D와 H 사이의 거리)
⋮
+ (B와 C 사이의 거리)
+ (C와 A 사이의 거리)

  – 그리고 경로의 거리가 K보다 작으면 'yes'라고 답한다.

NP 알고리즘의 예

- 두 번째 단계에서 계산에 소요되는 시간은 선형 시간
  - 왜냐하면 입력으로 8개 도시가 주어질 때, 8개의 거리를 합하는데 걸리는 시간은 8번의 덧셈 연산
  - 계산된 경로의 거리와 K를 1번 비교하는 것이기 때문

➢ 다른 NP-완전 문제에 대해서도 위와 같이 상수를 사용하여, 각각의 문제를 결정 문제로 바꿀 수 있다.

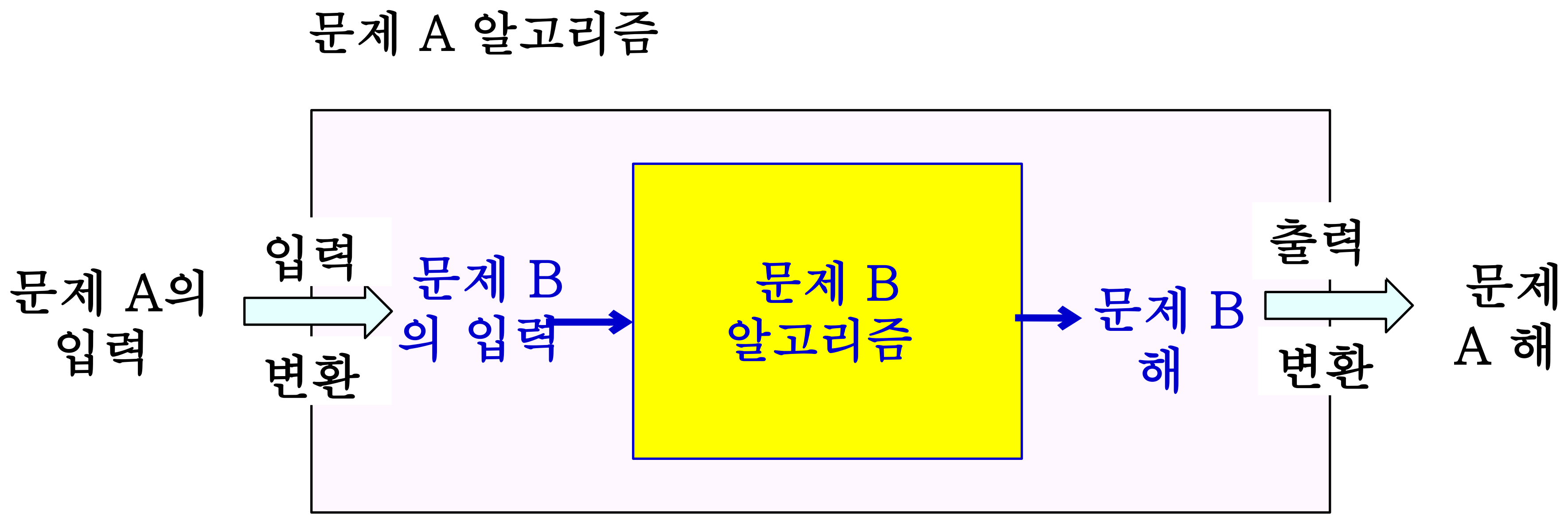
7.2 NP-완전 문제의 특성

- NP-완전 문제의 특성을 알기 위해 어떤 문제를 다른 문제로 변환 (reduction)하는 과정을 이해하여야 한다.

- 문제의 변환
  - 문제 A를 해결하기 위해서 문제 B를 해결하는 알고리즘을 이용하는 것을 의미

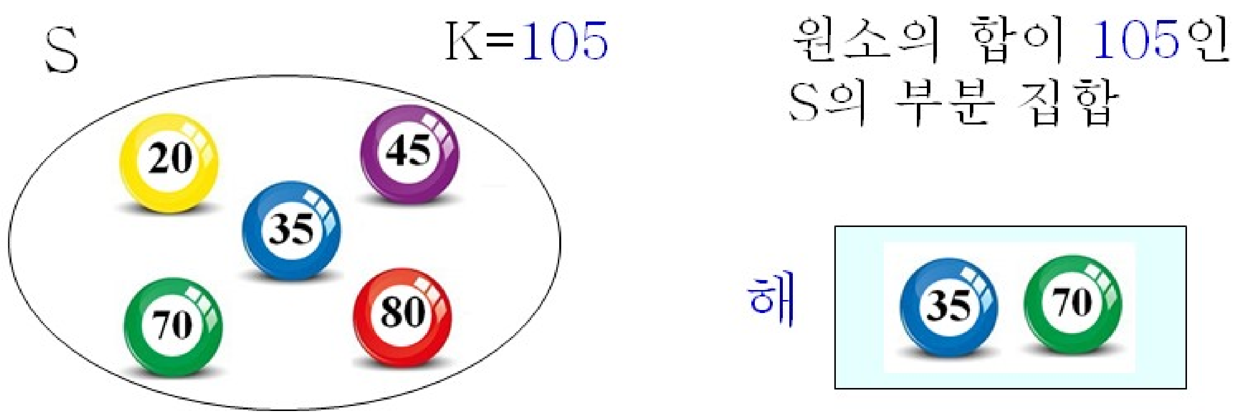
- 문제의 변환 과정
  - 먼저 문제 A의 입력을 문제 B의 입력 형태 (format)로 변환시키고,
  - 변환된 입력으로 문제 B를 해결하는 알고리즘을 수행
  - 마지막으로 수행 결과인 해를 문제 A의 해로 변환

문제의 변환 과정



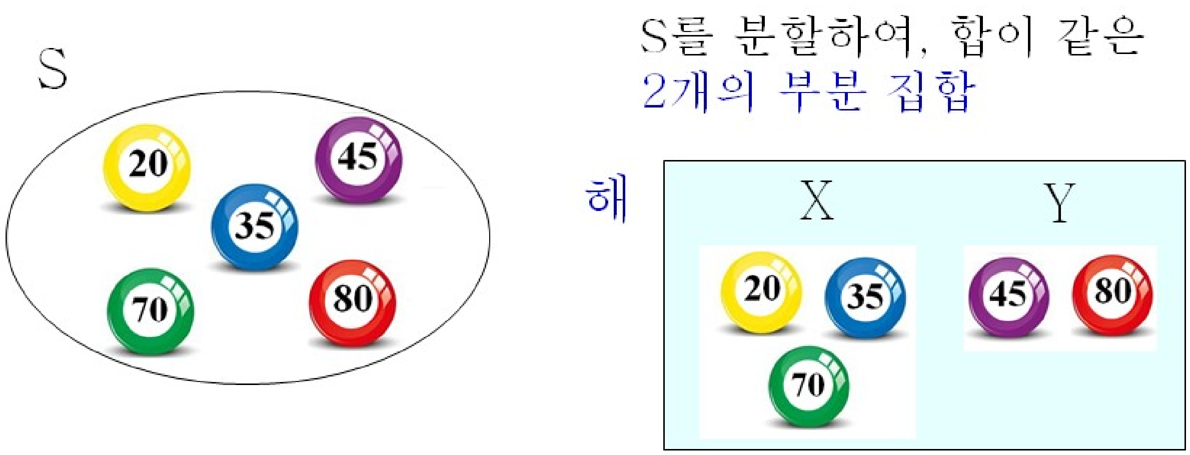
문제 변환

- 문제 A = 부분 집합의 합(Subset Sum) 문제
  - 정수의 집합 S에 대해, 부분 집합의 합 문제는 S의 부분 집합들 중에서 원소의 합이 K가 되는 부분 집합을 찾는 문제
  - 예를 들어, S={20, 35, 45, 70, 80}이고, K=105이라면, {35, 70}의 원소의 합이 105가 되므로, 문제의 해는 {35, 70}



문제 변환

- 문제 B = 분할 (Partition) 문제
  - 정수의 집합 S에 대해, S를 분할하여 원소들의 합이 같은 2개의 부분 집합을 찾는 문제
  - 예를 들어, S={20, 35, 45, 70, 80}이 주어지면, X = {20, 35, 70}과 Y = {45, 80}이 해
  - 왜냐하면 X의 원소의 합이 20+35+70 = 125이고, Y의 원소의 합도 45+80 = 125이기 때문



변환 아이디어

- 부분 집합의 합 문제의 입력인 집합 S를 분할 문제의 입력으로 변환할 때 t를 집합 S에 추가

  t = s - 2K

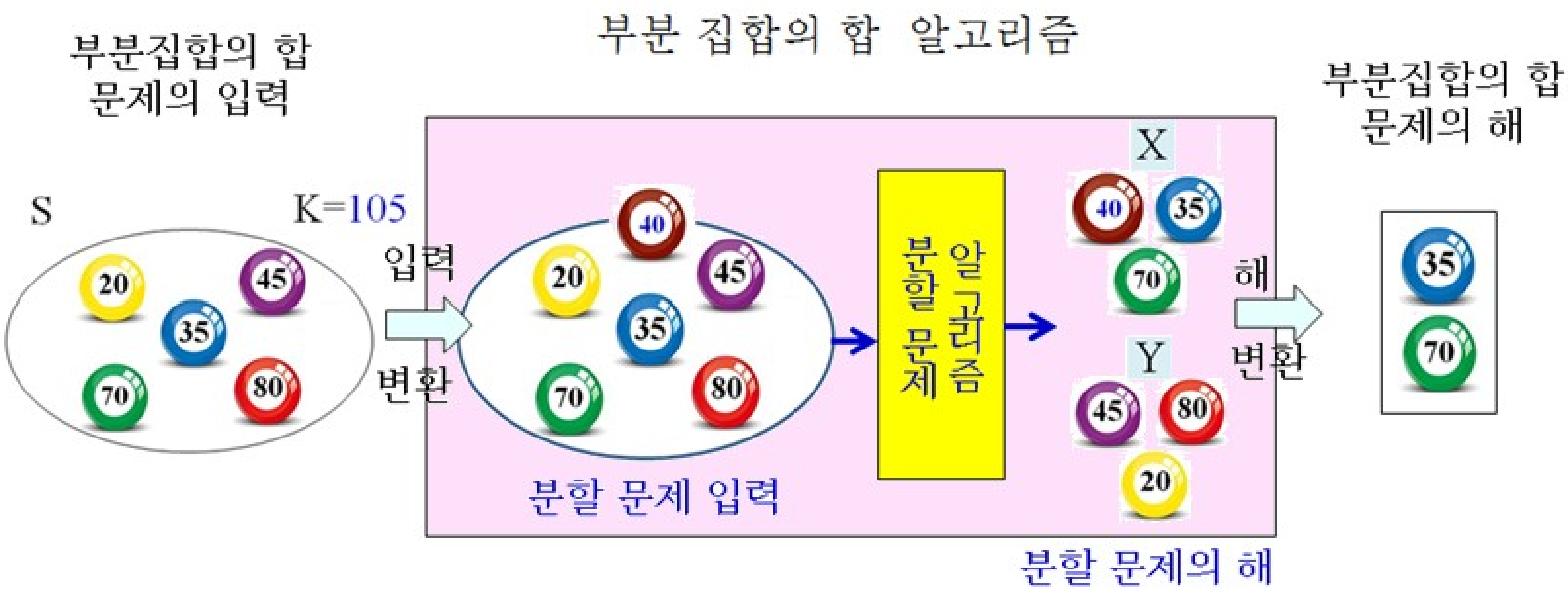
- 단, s는 집합 S의 모든 원소의 합
- 부분 집합의 합 문제를 해결하기 위해서, 집합 S' = S∪{t}를 입력으로 하는 분할 문제를 위한 알고리즘을 이용
- 분할 문제 알고리즘의 해인 2개의 집합 X와 Y에 대해, X에 속한 원소의 합과 Y에 속한 원소의 합이 같으므로, 각각의 합은 (s−K)
- 왜냐하면 새 집합 S'의 모든 원소의 합이 s+t = s+(s-2K) = 2s-2K이고, (2s−2K)의 1/2이면 (s−K)이므로

변환 아이디어

- 따라서 분할 문제의 해인 X와 Y 중에서 t를 가진 집합에서 t를 제거한 집합이 부분 집합의 합 문제의 해가 된다.
- 왜냐하면 만일 X에 t가 속해 있었다면, X에서 t를 제외한 원소의 합이 (s-K)-t = (s-K)-(s-2K) = s-K-s+ 2K = K가 되기 때문
- 그러므로 부분 집합의 합 문제의 해는 바로 X-t이다.

변환 예제

– 부분 집합의 합 문제를 분할 문제로 변환하여 해결하는 예제로 s, K, t 값은 다음과 같다.
– s = 20+ 35+ 45+ 70+ 80 = 250
– K = 105
– t = s−2K = 250−(2x105) = 250−210 = 40



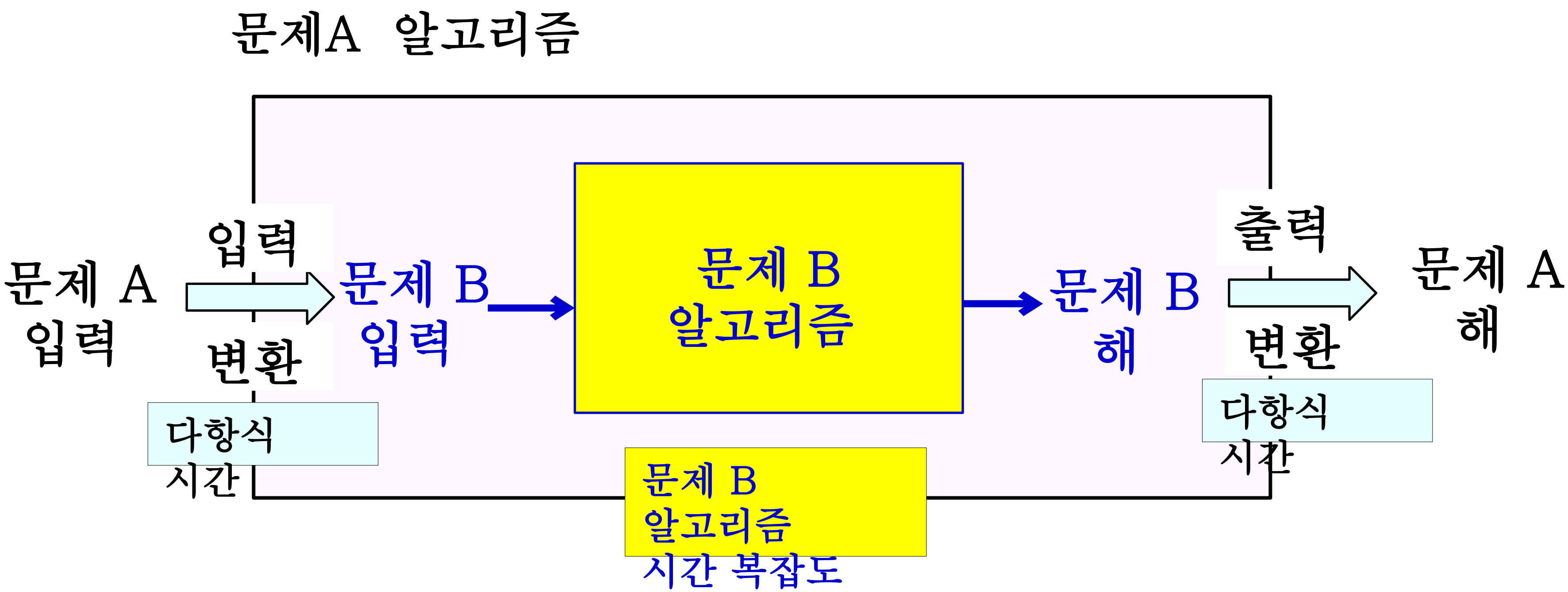
문제 변환과 시간 복잡도

- 문제를 변환하는 전 과정의 시간 복잡도는 다음의 3 단계의 시간 복잡도의 합
  - 문제 A의 입력을 문제 B의 입력으로 변환하는 시간
  - 문제 B를 위한 알고리즘이 수행되는 시간
  - 문제 B의 해를 문제 A의 해로 변환하는 시간

  - 첫 단계와 세 번째 단계
    - 단순한 입출력 변환이므로 다항식 시간에 수행

  - 두 번째 단계
    - 문제 변환의 시간 복잡도는 두 번째 단계의 시간 복잡도에 따라 결정된다.
    - 두 번째 단계가 다항식 시간이 걸리면, 문제 A도 다항식 시간에 해결된다.

문제 변환과 시간 복잡도


문제A 알고리즘
문제 A 입력
입력 변환
다항식 시간
문제 B 입력
문제 B 알고리즘
문제 B 해
출력 변환
다항식 시간
문제 A 해
문제 B 알고리즘 시간 복잡도

추이 (transitive) 관계

- 문제 A와 문제 B 사이에 다항식 시간 변환 관계가 성립하면, 문제 A가 문제 B로 다항식 시간에 변환 (polynomial time reduction)이 가능하다고 한다.
- 동시에 문제 B가 문제 C로 다항식 시간에 변환 가능하면, 결국 문제 A가 문제 C로 다항식 시간에 변환 가능하다.
- 이러한 추이 관계로 NP-완전 문제들이 서로 얽혀 있어서, NP-완전 문제들 중에서 어느 한 문제만 다항식 시간에 해결되면, 모든 다른 NP-완전 문제들이 다항식 시간에 해결된다.

NP-하드(Hard) 문제 집합

- 문제의 변환을 통해 또 다른 문제 집합인 NP-하드 (hard) 문제 집합을 다음과 같이 정의

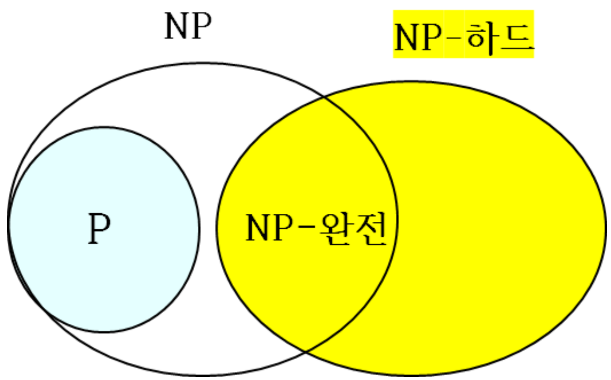
> 어느 문제 A에 대해서, 만일 모든 NP 문제가 문제 A로 다항식 시간에 변환이 가능하다면, 문제 A는 NP-하드 문제이다.

- '하드'란 적어도 어떤 NP 문제보다 해결하기 어렵다는 뜻

- 모든 NP 문제가 NP-하드 문제로 다항식 시간에 변환 가능하여야 함에도 불구하고, NP-하드 문제는 반드시 NP 문제일 필요는 없다.

문제 집합들 사이의 관계

- NP-완전 문제는 NP-하드 문제이면서 동시에 NP 문제이다.



[주의] 문제 포함 관계는 아직 증명되지 못했으나 대부분의 학자들이 맞을 것이라고 생각한다.

- NP-완전 문제의 정의
  - 문제 A가 NP-완전 문제가 되려면,
    - 문제 A는 NP 문제이고, 동시에
    - 문제 A는 NP-하드 문제이다.

7.3 NP-완전 문제의 소개

- NP-완전 문제 집합에는 컴퓨터 분야 뿐만 아니라 과학, 공학, 의학, 약학, 경영학, 정치학, 금융 심지어는 문화 분야 등에까지 광범위한 분야에서 실제로 제기되는 문제들이 포함되어 있다.

- NP-완전 문제는 다항식 시간에 하나의 문제에서 다른 문제로 변환 가능

- 이러한 문제 변환은 부분 집합의 합 문제를 분할 문제로 변환하는 것같이 간단한 경우도 있고, 반면에 매우 복잡한 경우도 있다.

SAT (Satisfiability)

- 부울 변수 (Boolean variable)들이 ∨(OR)로 표현된 논리식이 여러 개 주어질 때, 이 논리식들을 모두 만족시키는 각 부울 변수의 값을 찾는 문제

[예제] 부울 변수 w, x, y, z에 대하여,

$(w \vee y), (\bar{w} \vee x \vee z), (\bar{x} \vee \bar{y} \vee \bar{z})$

– 해: w=true, x=true, y=false, z=true or false

$(w \vee \bar{x}), (x \vee \bar{y}), (y \vee \bar{w}), (w \vee x \vee y), (\bar{w} \vee \bar{x} \vee \bar{y})$

– 해: 없음

부분 집합의 합 (Subset Sum)

- 주어진 정수의 집합 S의 원소의 합이 K가 되는 S의 부분 집합을 찾는 문제

[예제]

- S = {20, 30, 40, 80, 90}이고, 합이 200이 되는 부분 집합을 찾고자 할 때,

- [해] {30, 80, 90}의 원소 합이 200

분할 (Partition)

- 주어진 정수의 집합 S를 분할하여 원소의 합이 같은 2개의 부분 집합을 찾는 문제

[예제]

- S = {20, 30, 40, 80, 90}일 때, S를 2개의 합이 동일한 부분 집합으로 분할하면,

- [해] X = {20, 30, 80}, Y = {40, 90}; 각각의 부분 집합의 합이 130

0-1 배낭 (Knapsack)

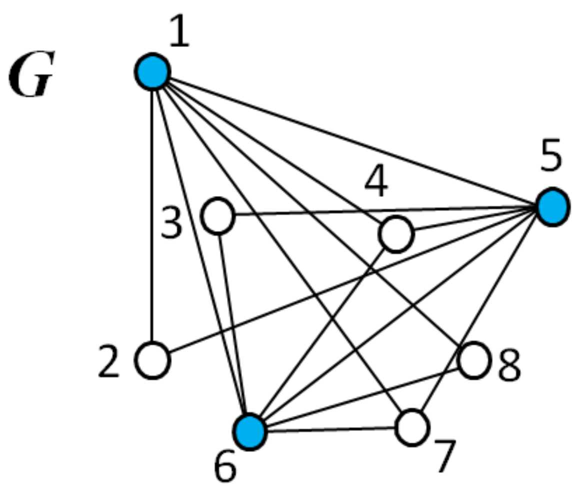
- 배낭의 용량이 C이고, 물건 각각의 무게와 가치가 w_i와 v_i 일 때, 단, i = 1, 2, …, n, 배낭에 담을 수 있는 물건의 최대 가치를 찾는 문제
  - 단, 담을 물건의 무게의 합이 배낭의 용량을 초과하지 말아야 한다.

[예제]

- C = 20kg, w_1 = 12kg, w_2 = 8kg, w_3 = 6kg, w_4 = 5kg이고, v_1 = 20, v_2 = 10, v_3 = 15, v_4 = 25
- [해] 물건 2, 3, 4를 배낭에 담으면, 그 무게의 합은 8+6+5 = 19kg, 그 가치의 합은 10+15+25 = 50으로 최대

정점 커버 (Vertex Cover)

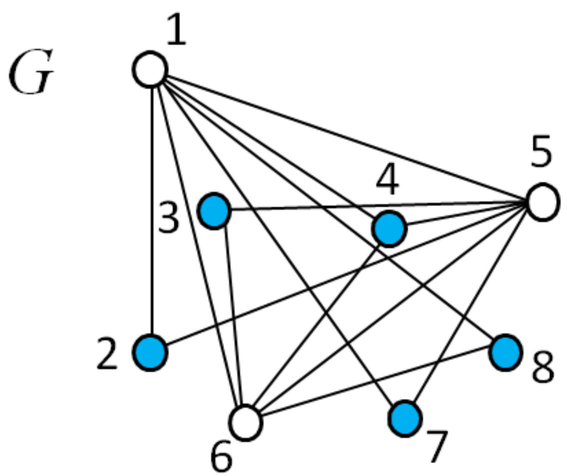
- 정점 커버란 주어진 그래프 G=(V,E)에서 각 간선의 양 끝점들 중에서 적어도 1개의 점을 포함하는 집합이다.
- 정점 커버 문제는 최소 크기의 정점 커버를 찾는 문제



- [해] {1, 5, 6}: 그래프의 각 간선의 양 끝점들 중에서 적어도 1개의 끝점이 점 1, 5, 6 중에 하나이다. 그리고 이는 최소 크기의 커버

독립 집합 (Independence Set)

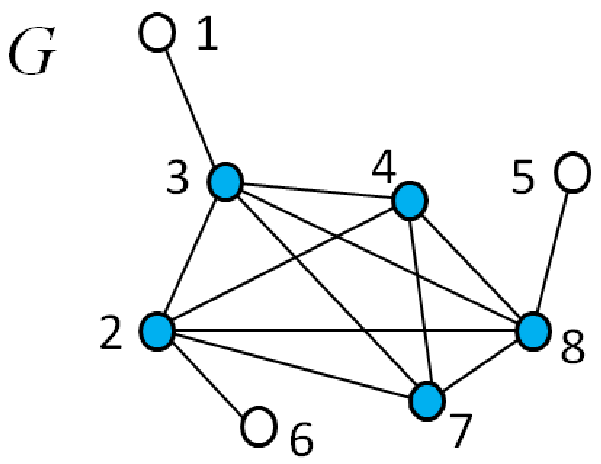
- 독립 집합이란 주어진 그래프 G=(V,E)에서 연결하는 간선이 없는 점들의 집합이다.
- 독립 집합 문제는 최대 크기의 독립 집합을 찾는 문제



- [해] {2, 3, 4, 7, 8}은 서로 간선으로 연결 안 된 최대 크기의 독립 집합

클리크 (Clique)

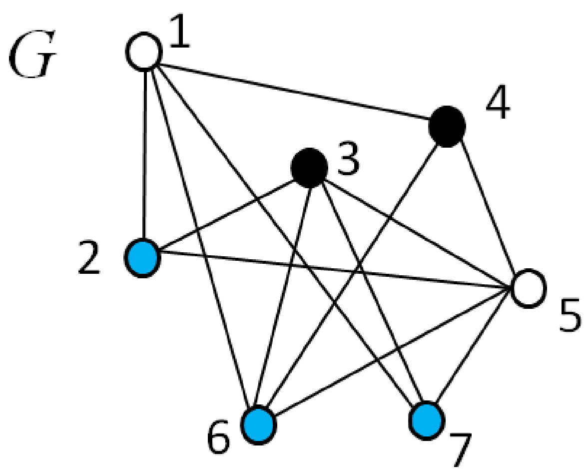
- 클리크란 주어진 그래프 G=(V,E)에서 모든 점들 사이를 연결하는 간선이 있는 부분 그래프이다.
- 클리크 문제는 최대 크기의 클리크를 찾는 문제



- [해] {2, 3, 4, 7, 8}은 서로 간선으로 모두 연결된 최대 크기의 클리크

그래프 색칠하기 (Graph Coloring)

- 그래프 색칠하기란 주어진 그래프 G=(V,E)에서 인접한 점들을 서로 다른 색으로 색칠하는 것이다.
- 그래프 색칠하기 문제는 가장 적은 수의 색을 사용하여 그래프를 색칠하는 문제



- [해] {1, 5}는 흰색, {3, 4}는 검은 색, {2, 6, 7}은 파란색으로 칠한다. 3가지 색보다 적은 수의 색으로 이 그래프를 칠할 수는 없다.

집합 커버 (Set Cover)

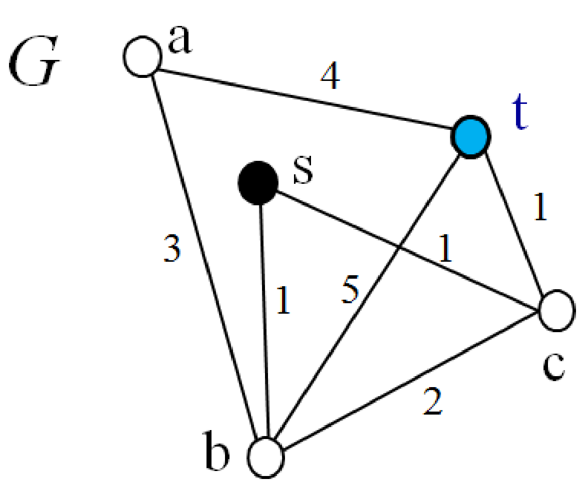
- 주어진 집합 S = {1, 2, 3, ⋯, n}에 대해서 S의 부분 집합들이 주어질 때, 이 부분 집합들 중에서 합집합하여 S와 같게 되는 부분 집합들을 집합 커버라고 한다.
- 집합 커버 문제는 가장 적은 수의 부분 집합으로 이루어진 집합 커버를 찾는 문제

[예제]

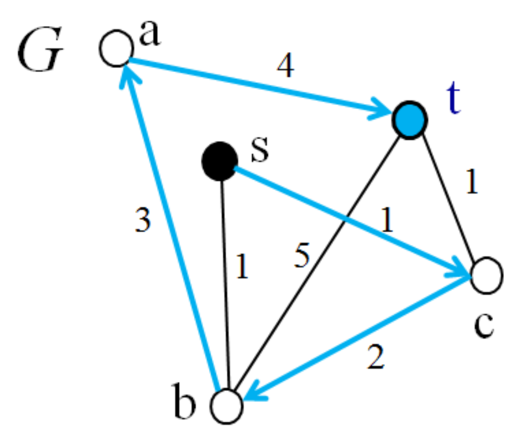
- S = {1, 2, 3, 4, 5}, 부분 집합: {1, 2, 3}, {2, 3, 4}, {3, 5}, {3, 4, 5} 라면,
- [해] {1, 2, 3}과 {3, 4, 5}를 합집합하면 S가 되고, 부분 집합 수가 최소이다.

최장 경로 (Longest Path)

- 주어진 가중치 그래프 G=(V, E)에서 시작점 s에서 도착점 t까지의 가장 긴 경로를 찾는 문제
  - 단, 간선의 가중치는 양수이고, 찾는 경로에는 반복되는 점이 없어야 한다.

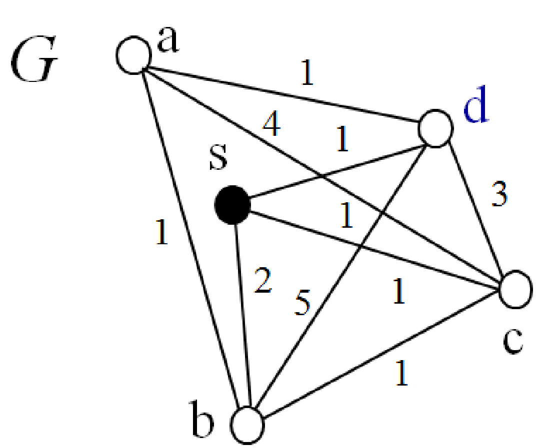


[해] s→c→b→a→t가 최장 경로로 그 길이는 10이다.

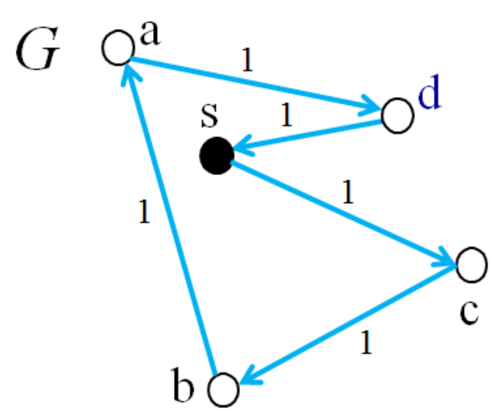


여행자 (Traveling Salesman) 문제

- 주어진 가중치 그래프 G=(V,E)에서, 임의의 한 점에서 출발하여, 다른 모든 점들을 1번씩만 방문하고, 다시 시작점으로 돌아오는 경로 중에서 최단 경로를 찾는 문제

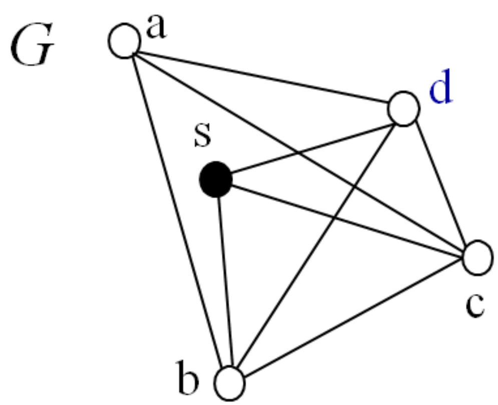


[해]

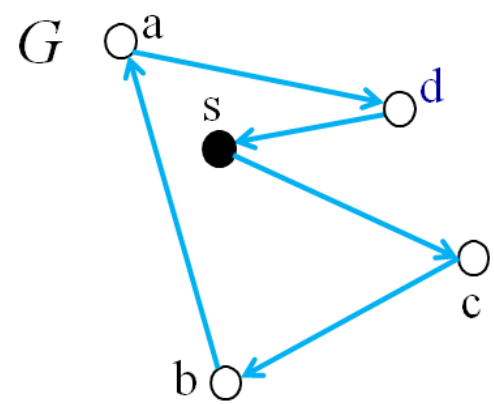


헤밀토니안 사이클 (Hamiltonian Cycle)

- 주어진 그래프 G=(V, E)에서, 임의의 한 점에서 출발하여 모든 다른 점들을 1번씩만 방문하고, 다시 시작점으로 돌아오는 경로를 찾는 문제
  - 간선의 가중치를 모두 동일하게 하여 여행자 문제의 해를 찾았을 때, 그 해가 헤밀토니안 사이클 문제의 해가 된다.



[해]

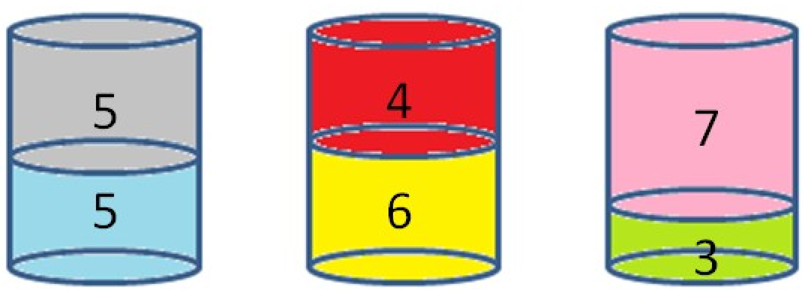


통 채우기 (Bin Packing)

- n개의 물건이 주어지고, 통 (bin)의 용량이 C일 때, 가장 적은 수의 통을 사용하여 모든 물건을 통에 채우는 문제
  - 단, 각 물건의 크기는 C보다 크지 않다.

[예제]

- 통의 용량 C=10이고, n=6개의 물건의 크기가 각각 5, 6, 3, 7, 5, 4
- [해] 3 울 수 있다.

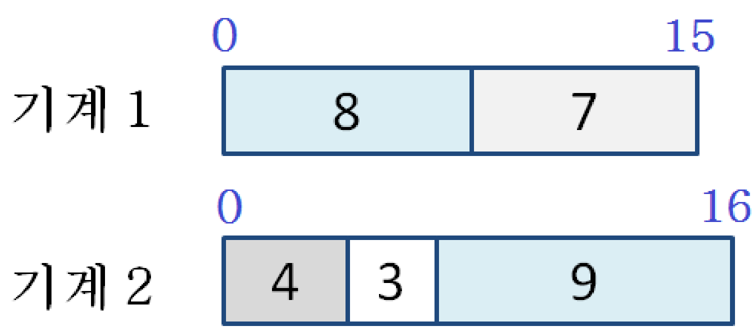


작업 스케줄링 (Job Scheduling)

- 각 작업의 수행 시간 t_i, 단, $i = 1, 2, 3, \cdots, n$, 그리고 m개의 동일한 성능의 기계가 주어질 때, 모든 작업이 가장 빨리 종료되도록 작업을 기계에 배정하는 문제

[예제]

- n = 5개의 작업이 주어지고, 각각의 수행 시간이 8, 4, 3, 7, 9이며, m = 2
- [해] 아래와 같이 작업을 배정하면 가장 빨리 모든 작업 종료



7.4 NP-완전 문제들의 활용

- 지금까지 살펴본 문제들은 각각 지수 시간 알고리즘을 가지고 있다.

- 각각의 문제는 문제 그 자체로서도 중요한 문제이지만, 실세계에서 해결해야 할 매우 광범위한 응용문제들과 직접적으로 연관되어 있다.

- 각각의 NP-완전 문제가 직접 또는 간접적으로 활용되는 사례는 다음과 같다.

SAT (Satisfiability)

- 반도체 칩 (Chip)을 디자인하는 전자 디자인 자동화 (Electronic Design Automation)
- 소프트웨어에 핵심적인 부분인 형식 동치 관계 검사 (Formal Equivalence Checking)
- 모델 검사 (Model Checking)
- 형식 검증 (Formal Verification)
- 자동 테스트 패턴 생성 (Automatic Test Pattern Generation)
- 인공 지능에서의 계획 (Planning)과 명제 모델을 컴파일하는 지식 컴파일 (Knowledge Compilation)
- 생물 정보 공학 분야에서 염색체로부터 질병 인자를 추출 또는 염색체의 진화를 연구하는데 사용되는 단상형 추론 (Haplotype Inference) 연구
- 소프트웨어 검증 (Software Verification)
- 자동 정리 증명 (Automatic Theorem Proving) 등

부분 집합의 합 (Subset Sum)

- 암호 시스템 개발에 사용되는데, 그 이유는 문제 자체는 얼핏 보기에 매우 쉬우나 해결하기는 매우 어렵기 때문이다.
- 실용적인 전자 태그 암호 시스템 (RFID Cryptosystem)
- 격자 기반 (Lattice-based) 암호화 시스템
- 공개 암호 시스템 (Public Key Cryptography)
- 컴퓨터 패스워드 (Password) 검사 및 메시지 검증
- 음악에도 적용하여 스마트폰 앱으로도 만들어진 사례도 있다.

분할 (Partition)

- 분할 문제를 보다 일반화하여 분할할 부분 집합 수를 2개에서 k개로 확장시키면, 더욱 더 다양한 곳에 응용 가능
- Switching Network에서 채널 그래프 비교
- 시간과 장소를 고려한 컨테이너의 효율적 배치
- 네트워크 디자인
- 인공 지능 신경망 네트워크 (Artificial Neural Network)의 학습
- 패턴 인식 (Pattern Recognition)
- 로봇 동작 계획 (Robotic Motion Planning)
- 회로 및 VLSI 디자인
- 의학 전문가 시스템 (Medical Expert System)
- 유전자의 군집화 (Gene Clustering) 등

0-1 배낭 (Knapsack)

- 다양한 분야에서 의사 결정 과정에 활용된다.
- 원자재의 버리는 부분을 최소화하는 분할
- 금융 분야에서 금융 포트폴리오 선택
- 자산 투자의 선택
- 주식 투자
- 다차원 경매 (Combinatorial Auction)
- 공개키 암호시스템인Merkle-Hellman Knapsack Cryp-tosystem
- 게임 스도쿠 (Sudoku) 등

		8		1				9
6		1		9		3	2	
	4			3	7			5
	3	5			8	2		
		2	6	5		8		
		4			1	7	5	
5			3	4			8	
	9	7		8		5		6
1				6		9		

정점 커버 (Vertex Cover)

- 집합 커버 문제의 특별한 경우이다.
- 다시 말하면 집합 커버 문제보다 더 일반적인 문제이다.
- 부울 논리 최소화 (Boolean Logic Minimization)
- 센서 (Sensor) 네트워크에서 사용되는 센서 수의 최소화
- 무선 통신 (Wireless Telecommunication)
- 토목 공학 (Civil Engineering)
- 전기 공학 (Electrical Engineering)
- 최적 회로 설계 (Circuit Design)
- 네트워크 플로우 (Network Flow)
- 생물 정보 공학에서의 유전자 배열 연구
- 미술관, 박물관, 기타 철저한 경비가 요구되는 장소의 경비 시스템 – CCTV 카메라의 최적 배치 (Art Gallery 문제) 등

집합 커버 (Set Cover)

- 집합 커버 문제의 응용은 정점 커버 문제의 응용을 포함
- 비행기 조종사 스케줄링 (Flight Crew Scheduling)
- 조립 라인 균형화 (Assembly Line Balancing)
- 정보 검색 (Information Retrieval)
- 도시 계획 (City Planning)에서 공공 기관 배치하기
- 컴퓨터 바이러스 찾기
- 기업의 구매 업체 선정
- 기업의 경력 직원 고용 등

독립 집합 (Independence Set)

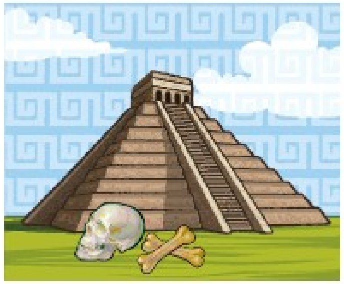
- 컴퓨터 비젼 (Computer Vision)
- 패턴 인식 (Pattern Recognition)
- 정보/코딩 이론 (Information/Coding Theory)
- 지도 레이블링 (Map Labeling)
- 분자 생물학 (Molecular Biology)
- 스케쥴링 (Scheduling)
- 회로 테스트
- CAD 등

클리크 (Clique)

- 생물 정보 공학에서 유전자 표현 데이터 (Gene Expression Data)의 군집화
- 단백질 구조 예측 연구
- 단백질 특성 연구
- 생태학에서 먹이 그물 (Food Web)에 기반한 종 (Species)에 관한 관계 연구
- 진화 계보 유추를 위한 연구
- 전자 공학에서는 통신 네트워크 분석
- 효율적인 집적 회로 설계
- 자동 테스트 패턴 생성 (Automatic Test Pattern Generation)
- 화학 분야에서는 화학 데이터베이스에서 화학 물질의 유사성 연구와 2개의 화학 물질의 결합의 위치를 모델링하는데 활용

그래프 색칠하기 (Graph Coloring)

- 생산 라인, 시간표 등의 스케줄링
- 무선 네트워크에서 주파수 할당 (Bandwidth Allocation)
- 컴파일러의 프로그램 최적화
- 패턴 인식
- 데이터 압축 (Data Compression)
- 스도쿠 (Sudoku) 게임: 81개의 점이 있는 그래프에서 9개의 색으로 점을 색칠하기와 동일하다
- 생물학에서 생체 분석
- 고고학 자료 분석에 응용



최장 경로 (Longest Path), 여행자 (Traveling Salesman) 문제, 헤밀토니안 사이클 (Hamiltonian Cycle)

- 운송 및 택배 사업에서의 차량 운행 (Vehicle Routing)
- 가전 수리 및 케이블 회사에서의 서비스 콜의 스케줄링
- 회로 기판에 구멍을 뚫기 위한 기계의 스케줄링
- 회로 기판에서의 배선 (Wiring)
- 논리 회로 테스트
- 건축 시공에서의 배관 및 전선 배치,
- 데이터의 군집화 (Clustering) 등

통 채우기 (Bin Packing)

- 다중 처리 장치 (Multiprocessor) 스케줄링
- 멀티미디어 저장 장치 시스템
- Video-on-Demand 서버의 비디오 데이터 배치 등의 자원 할당 (Resource Allocation)
- 생산 조립 라인에서의 최적화
- 산업 공학, 경영 공학의 주요 분야인 공급 망 경영 (Supply Chain Management)
- 트럭, 컨테이너에 화물 채우기
- 재료 절단 (Cutting Stock) 문제
- 작업의 부하 균등화 (Load Balancing)
- 스케줄링 (Scheduling)
- 프로젝트 경영 (Project Management)
- 재무 예산 집행 계획 (Financial Budgeting) 등

작업 스케줄링 (Job Scheduling)

- 컴퓨터 운영 체제의 작업 스케줄링
- 다중 프로세서 (Multiprocessor) 스케줄링
- 웹 서버 (Web Server)에서 사용자 질의 처리
- 주파수 대역 스케줄링 (Bandwidth Scheduling)
- 기타 산업 및 경영 공학에서의 공정 스케줄링
- 시간표 작성 (Timetable Design)
- 항공 산업에서 공항 게이트 (Gate) 스케줄링
- 조종사 스케줄링
- 정비사 스케줄링

요약

- NP-완전 문제의 특성은 어느 하나의 NP-완전 문제에 대해서 다항식 시간의 알고리즘을 찾아내면, 모든 다른 NP-완전 문제도 다항식 시간에 해를 구할 수 있다.
- 다항식 시간 복잡도를 가진 알고리즘으로 해결되는 문제의 집합을 P (Polynomial) 문제 집합이라고 한다.
- 어느 문제 A에 대해서, 만일 모든 NP 문제가 문제 A로 다항식 시간에 변환이 가능하다면, 문제 A는 NP-하드 문제이다.
- 문제 A가 NP-완전 문제가 되려면, 문제 A는 NP 문제이고 동시에 NP-하드 문제여야 한다.