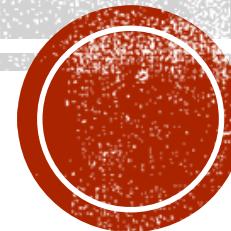


Lect 11. 계산복잡도와 다루기 힘든 정도 NP 이론의 소개





3

문제의 분류

Algorithm

2020-06-09

다항식시간 알고리즘(Polynomial-time Algorithm)

- 입력의 크기가 n 일 때, 최악의 경우 수행시간이 $W(n) \in O(p(n))$ 인 알고리즘.
 - 여기서 $p(n)$ 은 n 의 다항식 함수(polynomial function)이다.
- 보기
 - 시간 복잡도가 $2n, 3n^3 + 4n, 5n + n^{10}, n \lg n$ 인 알고리즘은 다항식시간 알고리즘이다.
 - 시간복잡도가 $2^n, 2^{0.01n}, 2^{\sqrt{n}}, n!$ 인 알고리즘은 다항식시간 알고리즘이 아니다.

다루기 힘든 정도(Intractability)

- 다항식으로 시간복잡도가 표시되는 알고리즘을 찾을 수 없는 문제를 “다루기 힘들다(intractable)”라고 한다.
- 반례: 연쇄행렬곱셈문제
 - 무작정알고리즘: $\Theta(2^n)$
 - 동적계획법 알고리즘: $\Theta(n^3)$
 - 따라서 이 문제는 다루기 힘들지 않다 (not intractable)

문제의 일반적인 분류

- 다차시간 알고리즘을 찾은 문제
- 다루기 힘들다고 증명된 문제
- 다루기 힘들다고 증명되지 않았고, 다차시간 알고리즘도 찾지 못한 문제

I. 다차시간 알고리즘을 찾은 문제

- 모든 알고리즘이 다차시간 알고리즘

- 정렬 문제: $\Theta(n \lg n)$
 - 정렬된 배열 검색 문제: $\Theta(\lg n)$
 - 행렬곱셈 문제: $\Theta(n^{2.38})$

- 다차시간이 아닌 알고리즘도 있으나, 다차시간 알고리즘을 찾은 경우

- 연쇄행렬곱셈 문제
 - 최단경로 문제
 - 최소비용신장트리(Minimum spanning tree) 문제

2. 다루기 힘들다고 증명된 문제

- 비다항식(**nonpolynomial**) 크기의 결과를 요구하는 비현실적인 문제
 - 보기: 모든 해밀토니안 회로를 결정하는 문제
 - 만일 모든 정점들 간에 에지가 있다면, $(n-1)!$ 개의 답을 얻어야 한다.
 - 이러한 문제는 하나의 해밀토니안 회로를 구하는 문제에 비해서 필요이상으로 많은 답을 요구 하므로 사실상 비현실적이고, 다루기 힘든 문제로 분류된다.
- 요구가 현실적이나, 다차시간에 풀 수 없다고 증명된 문제
 - 놀랍게도 이런 부류에 속하는 문제는 상대적으로 별로 없다.
 - 결정불가능한 문제(**Undecidable Problem**)
 - 그 문제를 풀 수 있는 알고리즘이 존재할 수 없다고 증명이 된 문제
 - 예: 종료 문제(**Halting Problem**) 등
 - 프레스버거 산술(**Presburger Arithmetic**) 문제
 - Fischer와 Rabin에 의하여 증명됨(1974)

종료 문제 (Halting Problem)

- 어떤 프로그램 P가 정상적으로 수행되어서 종료하는지를 결정하는 문제
 - 1936년 Alan Turing에 의해서 증명됨
- 정리:** 종료 문제는 결정불가능하다.
- 증명:** 이 문제를 풀 수 있는 알고리즘이 존재한다고 가정하자. 문자열 i를 입력하여 이를 처리하는 a라는 프로그램이 있다고 가정하자. a를 실행하여 이 프로그램의 실행이 끝나면 true, 끝나지 않고 영원히 반복되면 false를 반환하는 halt(a, i)라는 알고리즘이 있다고 주장한다. 아래의 문제를 halt(a, i)가 해결할 수 있는가?

```
function trouble(string s)
    if halt(s,s) == false
        return true
    else
        loop forever
```

종료 문제 (Halting Problem)

1. 만약 trouble(t)가 계산을 끝낸다고 하면, 그건 분명히 halt(t, t)가 반환값으로 **false**를 내놓기 때문이다. 그러나, 그것은 다시 trouble(t)가 멈추지 않고 영원히 지속된다는 말이다.
 2. 만약 trouble(t)이 무한히 돋다면, 그것은 halt가 영원히 계산을 끝내지 않거나, halt가 **true**를 반환하기 때문이다. 이것은 다시 halt가 임의의 문자열과 프로그램에 대해 판정할 수 없거나, trouble(t)가 계산 끝내고 멈춘다는 결론을 이끌어 낸다.
- 어떠한 경우에도 halt는 옳은 답을 내놓지 못하여, 처음의 주장과 모순을 일으킨다

```
function trouble(string s)
    if halt(s,s) == false
        return true
    else
        loop forever
```

halt(a,i) : a를 실행하여 이 프로그램의 실행이 끝나면 true, 끝나지 않고 영원히 반복되면 false를 반환하는 알고리즘

3. 다른기 힘들다고 증명되지 않았고, 다차시간알고리즘도 찾지 못한 문제

- 많은 문제들이 이 카테고리에 속한다.
 - 0-1 배낭채우기 문제
 - 외판원 문제
 - m -색칠하기 문제($m \geq 3$)
 - 해밀토니안 회로 문제 등등
- 이러한 문제들은 NP(Non-deterministic Polynomial) 문제에 속한다.

최적화 문제 vs. 결정 문제

- 최적화 문제(optimization problem) - 최적의 해를 찾아야 한다.
- 결정 문제(decision problem) - 대답이 “예” 또는 “아니오”로 이루어지는 문제 ⇒ 이론을 전개하고 이해하기 쉬움
- 최적화 문제에 대한 해답을 가지고 결정 문제에 대한 해답은 저절로 얻는다.
- 어떤 최적화 문제에 대해서 다차시간 알고리즘이 있다면, 그 알고리즘으로부터 그 문제에 해당하는 결정 문제에 대한 다항식 시간 알고리즘을 쉽게 유추해 낼 수 있다.
- NP와 P를 다룰 때 결정 문제만 고려해도 충분하다.

Decision Versus Search Problems

Decision Problem

YES/NO answers

Does G have a
Hamilton cycle?

Can G be
3-colored ?

Search Problem

Find a Hamilton cycle
in G if one exists,
else return NO

Find a 3-coloring of
G if one exists, else
return NO



What is an efficient algorithm?

Is an $O(n)$ algorithm efficient?

How about $O(n \log n)$?

$O(n^2)$?

$O(n^{10})$?

$O(n^{\log n})$?

$O(2^n)$?

$O(n!)$?

polynomial time

$O(n^c)$ for some
constant c

non-polynomial
time



NP문제의 예

▪ 외판원 문제

- 최적화 문제: 가중치가 있는 방향 그래프에서, 한 정점에서 출발하여 다른 모든 정점을 정확히 한 번씩 만 방문하면서 출발점으로 돌아오는 일주여행경로 중에서 총 여행거리가 최소가 되는 경로를 구하시오.
- 결정 문제: 어떤 양수 d 가 주어지고, 총 일주여행거리가 d 를 넘지 않는 경로가 있는지 없는지를 결정하시오.

▪ 0-1 배낭채우기 문제

- 최적화 문제: 배낭에 넣을 아이템의 무게와 가치를 알고 있을 때, 용량이 W 가 되는 배낭에 아이템을 총 이익이 최대가 되도록 채우시오.
- 결정 문제: 용량이 W 가 되는 배낭에 아이템을 총 이익이 최소한 P 가 되도록 채울 수 있는지를 결정하시오

검증(verification)

- 정의: P는 다차 시간 알고리즘으로 풀 수 있는 결정 문제들의 집합이다.
P에 속해 있는 문제들 - 정렬문제, 검색문제, 행렬곱셈문제 등.
P에 속해 있지 않은 문제 - Presburger Arithmetic
- 검증(Verification): 결정 문제의 답이 "예"인지를 검증하는 절차. 예를 들어 외판원 결정 문제의 답이 "예"라면, 한 일주여행경로가 주어졌을 때, 그 경로가 과연 그런지를 확인한다.

```
function verify(G: weighted-digraph;
               d: number;
               S: claimed-tour);
begin
  if S is a tour and the total weight of the edges in S <= d then
    verify := true
  else
    verify := false
end;
```

이 검증 절차는 다차 시간 안에 수행될 수 있다. 즉, d 보다 작은 일주여행경로를 찾는 것이 아니라(이 과정은 다차 시간으로 해결하지 못할 수도 있다), 주어진 일주여행경로가 d 보다 작게 걸리는 것인지를 알아보는 것이다.

Nondeterministic Algorithm(비결정적 알고리즘)

1. 추측단계(Guessing state: 비결정적임):

문제의 입력이 주어지면, 단순히 해답을 추측한다.(말도 안 되는 답이어도 상관없음)

2. 검증단계(Verification state: 결정적임):

입력: 입력과 추측한 해답

출력:

- “맞음”이라는 답을 주고 멈춘다
- “틀림”이라는 답을 주고 멈춘다
- 무한 루프

실제 상황에서 이 비결정적 알고리즘으로 문제를 푸는 것은 불가능하다.

그러나 다음과 같은 경우에는 비결정적 알고리즘이 결정 문제를 “푼다”고 한다:

- “예” 답을 줄 입력에 대해서 검증단계가 “맞음” 답을 줄 추측한 해답이 있다.
- “아니오” 답을 줄 입력에 대해서 검증단계가 “맞음” 답을 줄 추측한 해답이 없다.

NP의 정의

- 정의: 다차시간 비결정적 알고리즘(Polynomial-time nondeterministic algorithm)이란 검증단계가 다차시간에 수행되는 비결정적 알고리즘을 말한다.
- 정의: NP(Nondeterministic Polynomial)는 다차 시간 비결정적 알고리즘에 의해 풀 수 있는 모든 결정 문제의 집합이다.
- 어떤 결정 문제에 대해서 검증을 다차 시간에 하는 알고리즘이 있다면, 그 결정 문제는 NP에 속한다.
- 어떤 결정 문제를 풀 수 있는 다차 시간 알고리즘을 찾을 수 없을 때, 다차 시간 비결정적 알고리즘을 찾으면 그 문제는 NP에 속한다.
- 사색: 그러면 어떤 결정 문제가 주어졌을 때, 그 문제가 NP임을 어떻게 증명할까?

NP의 특징

- P에 속하는 모든 문제는 당연히 NP에도 속한다.
- NP에 속하지 않는 문제는 어떤 것이 있는가?
 - 다루기 힘들다(intractable)고 증명이 된 문제, 즉 Halting 문제, Presburger Arithmetic 문제 등이다.
- (중요한 사실) NP에 속한 문제 중에서 P에 속하지 않는다고 증명이 된 문제는 하나도 없다. 따라서 아마도 $NP - P = 0$ 일지도 모른다. 그럼 $P = NP$? 이것이 사실이라면 거의 모든 결정 문제가 NP에 속하기 때문에 우리가 아는 거의 모든 문제가 다른 시간 알고리즘이 있다는 얘기가 된다. 사실 많은 사람들이 $P \neq NP$ 일 것이라고 생각하고 있기는 하지만 아무도 이것을 증명하지 못하고 있는 것이다

NP-complete(완전)

- NP-완전 문제에 속하는 문제 중에서 어떤 하나 만이라도 P에 속한다는 것이 밝혀지면, 다른 모든 문제도 P에 속해야 한다.
- CNF(Conjunctive Normal Form): x 를 논리변수(logical variable)라고 하면, x 가 참이라는 말은 \bar{x} 는 거짓이라는 말과 동일하다. x 나 \bar{x} 는 리터럴(literal)이라 하고, \vee 연산자로 리터럴을 결합하면 절(clause)이라 한다. \wedge 로 절을 연결하면 CNF가 된다. 예를 들면,
 $(\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_4}) \wedge (\overline{x_2} \vee x_3 \vee x_4)$ 는 CNF이다.
- CNF-Satisfiability 결정 문제: CNF가 참이 될 수 있도록 논리값(참 또는 거짓)을 지정할 수 있는지의 여부를 결정하는 문제.

보기:

- $(x_1 \vee x_2) \wedge (\overline{x_2} \vee \overline{x_3}) \wedge \overline{x_2}$ - yes
- $(x_1 \vee x_2) \wedge x_1 \wedge x_2$ - no

이 결정 문제는 NP에 속한다.

NP-complete(cont)

- **변환(transformation) 알고리즘:** 풀고 싶은 어떤 문제를 A라고 하고, 이미 알고리즘을 알고 있는 어떤 문제를 B라고 하자. A에 대해서 “예”의 답을 해줄 모든 입력을 B에 대해서도 “예”의 답을 해줄 입력으로 변환하는 알고리즘. 그러면, 변환 알고리즘과 B에 대한 알고리즘을 합하면, A에 대한 알고리즘이 나온다.
- **정의:** A에서 B로 다항식시간에 변환하는 알고리즘이 있다면, A는 “P-시간 (polynomial-time) 다대일 축소가능(many-one reducible)”하다고 하고, $A \in P$ 로 쓴다. 따라서 만일 B문제에 대해서 P-시간 알고리즘이 있고, A에서 B로의 변환 알고리즘도 P-시간이라면, 그 두 알고리즘을 합함으로서 A에 대한 P-시간 알고리즘을 얻게 된다.
- **정리 1:** 결정 문제 B가 P에 속하고 $A \in P$ 이면 결정 문제 A는 P에 속한다.

NP-complete(cont)

- 정의: 만일 (1) 문제 B가 NP에 속하고, (2) NP에 속해 있는 모든 다른 문제 A에 대해서 $A \leq B$ 이면 B는 NP-완전이라고 불리운다.
- 어떤 문제가 NP-완전인지를 위의 정의에 근거해서 증명하는 일은 매우 어렵다. 왜냐하면 NP에 속한 모든 문제가 그 문제로 축소가능(reducible)하다는 것을 보여야 하기 때문이다. 그러나 다행스럽게도, 1971년 Cook이 다음의 2 정리를 증명했다.
- 정리 2: (Cook's Theorem) CNF-Satisfiability 문제는 NP-완전이다.
- 정리 3: 만일 (1) 문제 C가 NP에 속하고, (2) 어떤 NP-완전 문제 B에 대해서 $B \leq C$ 이면, C는 NP-완전이다.
- Cook에 의해서 CNF-Satisfiability 문제가 NP-완전임을 알았기 때문에, 주어진 문제가 NP-완전인지 아닌지는 위의 정리에 의해서 비교적 쉽게 증명할 수 있다.

다시 정리하면 ... 문제 분류

- 다항식 시간 (polynomial time)의 알고리즘이 아직 발견되지 않은 NP-완전 (Nondeterministic Polynomial-Complete) 문제를 소개한다.
- 다항식 시간 알고리즘을 가진 문제의 집합인 **P** (Polynomial) 문제
- 비결정적 다항식 시간 (Nondeterministic Polynomial time) 알고리즘을 가진 **NP** 문제
- NP 문제만큼 어려운 **NP-하드** (hard) 문제

문제의 분류

1. 다항식 시간복잡도를 가진 알고리즘으로 해결되는 P (polynomial) 문제 집합:
 - 이전까지 살펴본 대부분의 문제들 (집합 커버 문제 알고리즘, 배낭문제 알고리즘 제외)
 - 시간복잡도가 $O(\log n)$, $O(n)$, $O(n \log n)$, $O(n^2)$, $O(n^3)$ 등이고, 이러한 시간복잡도는 점근적 표기법에 따르면 $O(n^k)$ 에 포함되기 때문이다. 단, k 는 양의 상수이다.
2. 다항식보다 큰 시간복잡도를 가진 알고리즘으로 해결되는 문제 집합

문제의 분류(cont)

2. 다항식보다 큰 시간복잡도를 가진 알고리즘으로 해결되는 문제 집합

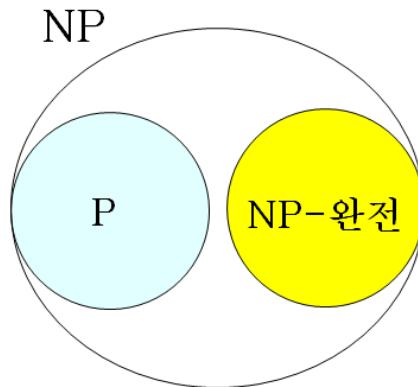
- 여러 가지 문제 집합으로 다시 분류된다.
- 그 중에 가장 중요한 문제 집합은 지수 시간 (exponential time) 시간복잡도를 가진 알고리즘으로 해결되는 **NP-완전** 문제 집합이다.
- **NP-완전 문제의 특성**: 어느 하나의 **NP-완전** 문제에 대해서 다항식 시간의 알고리즘을 찾아내면 (즉, 다항식 시간에 해를 구할 수 있으면) 모든 다른 **NP-완전** 문제도 다항식 시간에 해를 구할 수 있다.(앞의 슬라이드 참조)

문제의 분류(cont)

- 또한 P 문제 집합과 NP-완전 문제 집합을 둘 다 포함하는 문제의 집합인 **NP 문제 집합**이 있다.
- NP 문제 집합에 속한 문제를 **NP 문제**라고 한다.
- NP 문제는 비결정적 다항식 시간 (**Nondeterministic Polynomial time**) 알고리즘을 가진 문제이다.
- 비결정적 다항식 시간 알고리즘을 **NP 알고리즘**이라고 한다. **NP 알고리즘**은
 - 첫 번째 단계: 주어진 입력에 대해서 하나의 해를 '추측하고,'
 - 두 번째 단계: 그 해를 다항식 시간에 확인(verification)한 후에, 그 해가 '맞다'라고 답한다.

문제의 분류(cont)

- NP 알고리즘은 해를 찾는 알고리즘이 아니라, **해를 다항식 시간에 확인하는 알고리즘**이다.
- P 문제, NP-완전 문제, NP 문제 집합 사이의 관계를 보여준다.



문제의 분류(cont)

- 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 때문이다.
 - 즉, P 문제를 위한 NP 알고리즘은 해를 추측하는 단계를 생략하고, 해를 확인하는 단계 대신에 **해를 직접 다항식 시간에 구하고** 확인 결과를 '맞다'라고 답한다.

문제의 분류(cont)

- NP 알고리즘의 예를 살펴보자.
- NP 알고리즘은 추측한 해를 확인하여 '맞다'라고 답하므로, 문제의 해가 'yes' 또는 'no'가 되도록 주어진 문제를 변형시켜야 한다. 이러한 유형의 문제를 **결정 (decision) 문제**라고 한다.
- 예를 들어, 각 도시를 한 번씩만 방문하고 시작 도시로 돌아오는 최단 경로의 거리를 찾는 문제인 여행자 문제 (Traveling Salesman Problem)는 상수 **K**를 사용하여 다음과 같이 결정 문제로 변형된다.

각 도시를 1번씩만 방문하고 시작 도시로 돌아오는 경로의 거리가 **K**보다 짧은 경로가 있는가?

- 7개 도시 ($A B C D E F G H$)에 대한 여행자 문제의 NP 알고리즘은 다음과 같다. 단, A는 시작 도시이다.
- 7개 도시 ($A B C D E F G H$)의 여행자 문제의 하나의 해를 추측한다. 예를 들어, $A G D H$ $E B C$ 를 추측했다고 가정하자.
- 추측한 해의 값을 다음과 같이 계산한다.

해의 값 = (A와 G 사이의 거리)
+ (G와 D 사이의 거리)
+ (D와 H 사이의 거리)
...
+ (B와 C 사이의 거리)
+ (C와 A 사이의 거리)
- 그리고 해의 값이 K보다 작으면 'yes'라고 답한다.

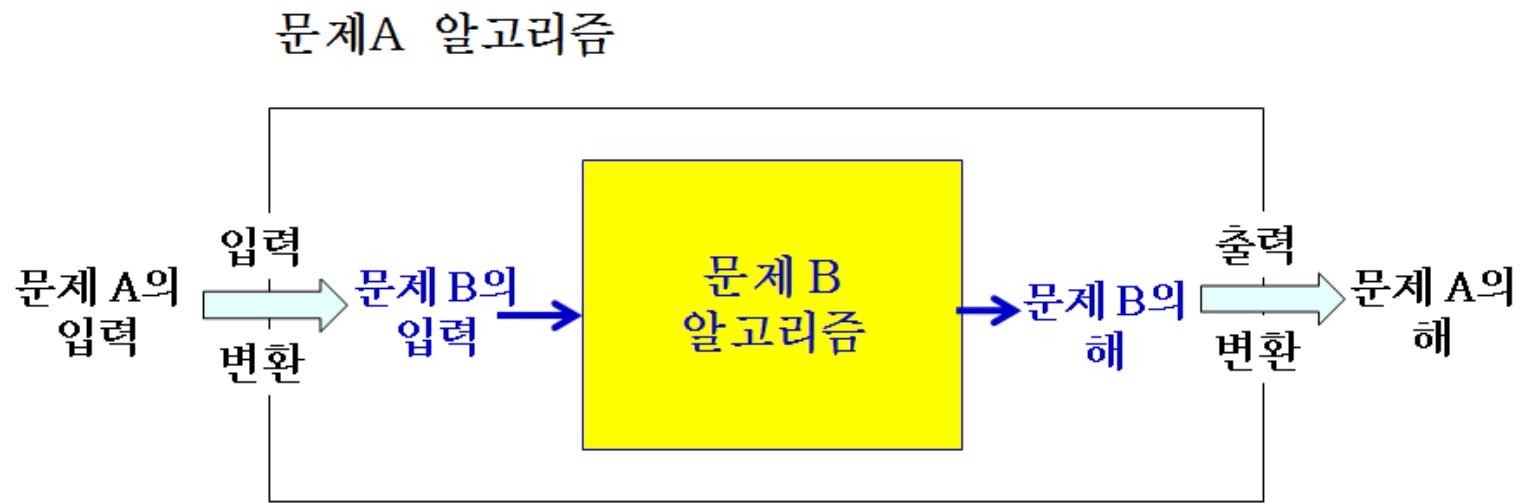
- 두 번째 단계에서 계산에 소요되는 시간은 선형 시간임을 쉽게 알 수 있다. 왜냐하면 입력으로 7개 도시가 주어질 때, 7개의 거리를 합하는데 걸리는 시간은 6번의 덧셈 연산이면 되고, 계산된 해의 값과 K 를 1번 비교하는 것이기 때문이다.
- 다른 NP-완전 문제에 대해서도 위와 같이 상수를 사용하여, 각각의 문제를 결정 문제로 바꿀 수 있다.

NP-완전문제(NP-complete problem)의 특성

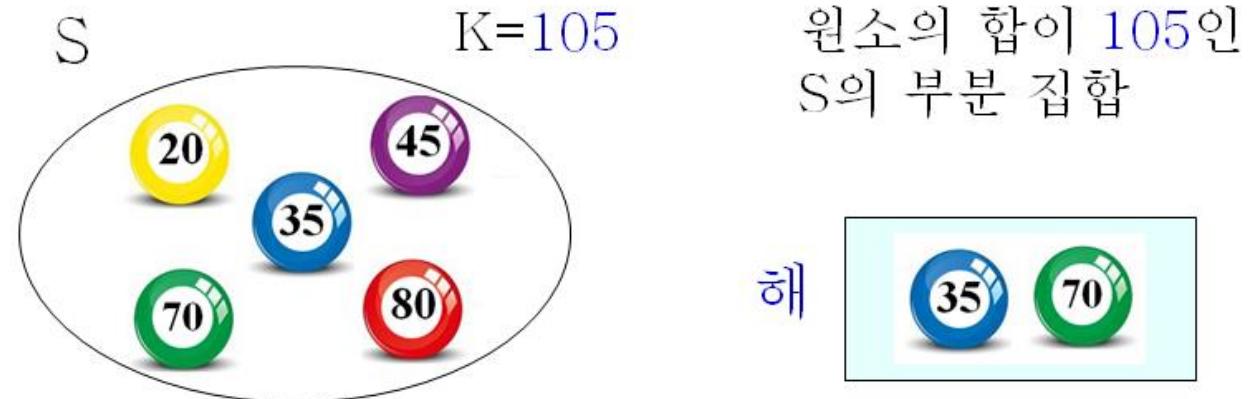
NP-완전 문제의 특성

- NP-완전 문제의 특성을 상세히 알기 위해서는 먼저 어떤 문제를 다른 문제로 **변환** 또는 **환원 (reduction)**하는 과정을 이해하여야 한다.
- 문제의 **변환이란** 문제 A를 해결하기 위해서 문제 B를 해결하는 알고리즘을 이용하는 것을 의미한다.
- 이를 위해 먼저 문제 A의 입력을 문제 B의 입력 형태 (**format**)로 **변환**시키고, 변환된 입력으로 문제 B를 해결하는 알고리즘을 수행한다. 마지막으로 수행 결과인 해를 문제 A의 해로 **변환**시킨다.

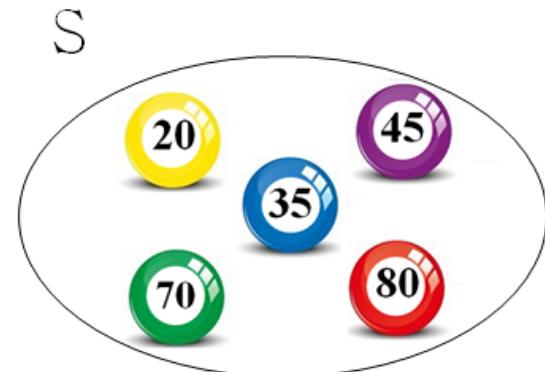
- 문제의 변환 과정을 보이고 있다.



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

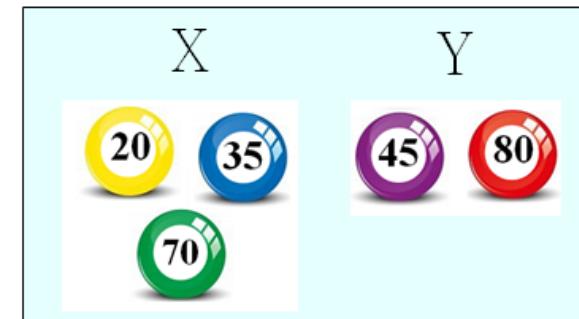


- 문제 B = 분할 문제: 정수의 집합 S에 대해, S를 분할하여 원소들의 합이 같은 2개의 부분 집합을 찾는 문제이다.
- 예를 들어, $S=\{20, 35, 45, 70, 80\}$ 이 주어지면, $X = \{20, 35, 70\}$ 과 $Y = \{45, 80\}$ 이 해이다.
- 왜냐하면 X의 원소의 합이 $20+35+70 = 125$ 이고, Y의 원소의 합도 $45+80 = 125$ 이기 때문이다.



S를 분할하여, 합이 같은
2개의 부분 집합

해



- 부분 집합의 합 문제의 입력인 집합 S 를 분할 문제의 입력으로 변환할 때 t 를 집합 S 에 추가한다.

$$t = s - 2K$$

- s 는 집합 S 의 모든 원소의 합이다.
- 부분 집합의 합 문제를 해결하기 위해서, 집합 $S' = S \cup \{t\}$ 를 입력으로 하는 분할 문제를 위한 알고리즘을 이용한다.
- 분할 문제 알고리즘의 해인 2개의 집합 X 와 Y 에 대해, X 에 속한 원소의 합과 Y 에 속한 원소의 합이 같으므로, 각각의 합은 $(s-K)$ 이다.
- 왜냐하면 새 집합 S' 의 모든 원소의 합이 $s+t = s+(s-2K) = 2s-2K$ 이고, $(2s-2K)$ 의 $1/2$ 이면 $(s-K)$ 이기 때문이다.

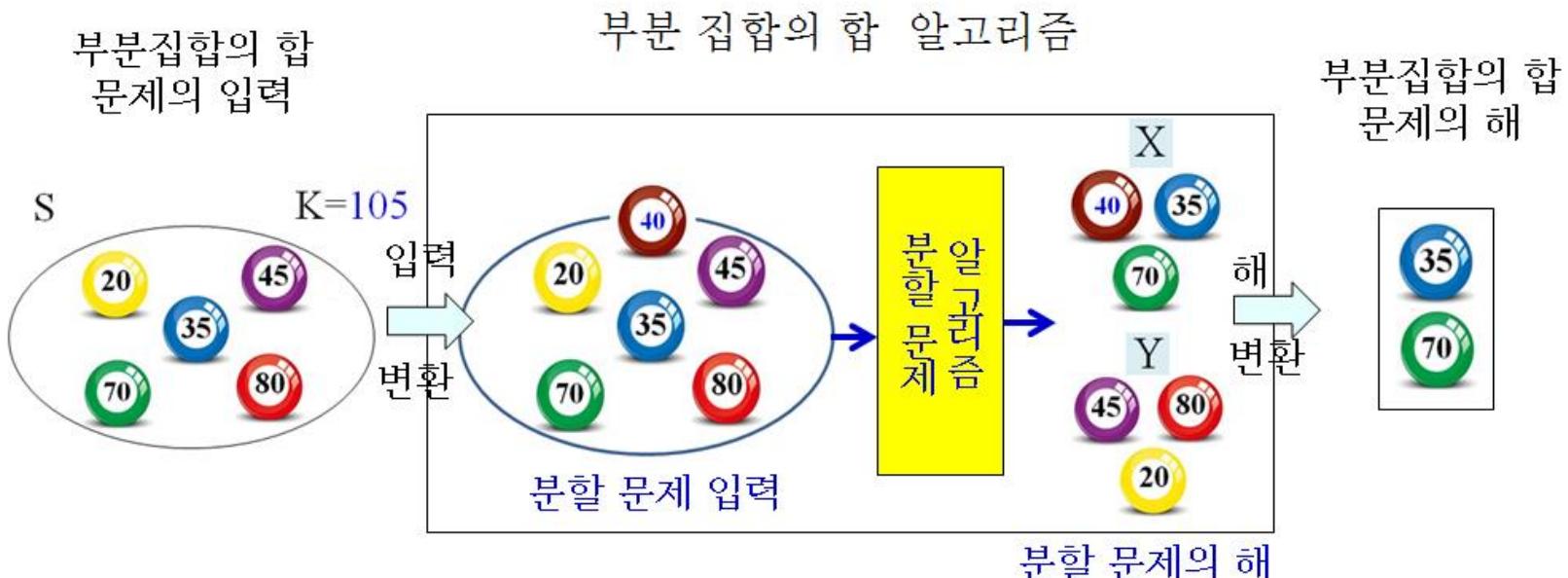
- 따라서 분할 문제의 해인 X 와 Y 중에서 t 를 가진 집합에서 t 를 제거한 집합이 부분 집합의 합 문제의 해가 된다.
- 왜냐하면 만일 X 에 t 가 속해 있었다면, X 에서 t 를 제외한 원소의 합이 $(s-K)-t = (s-K)-(s-2K) = s-K-s+2K = K$ 가 되기 때문이다.
- 그러므로 부분 집합의 합 문제의 해는 바로 $X-t$ 이다.

- 다음의 그림은 부분 집합의 합 문제를 분할 문제로 변환하여 해결하는 것을 위의 예제를 통해서 보이고 있다. 여기서 s , K , t 값은 다음과 같다. $t = s - 2K$

$$s = 20+35+45+70+80 = 250$$

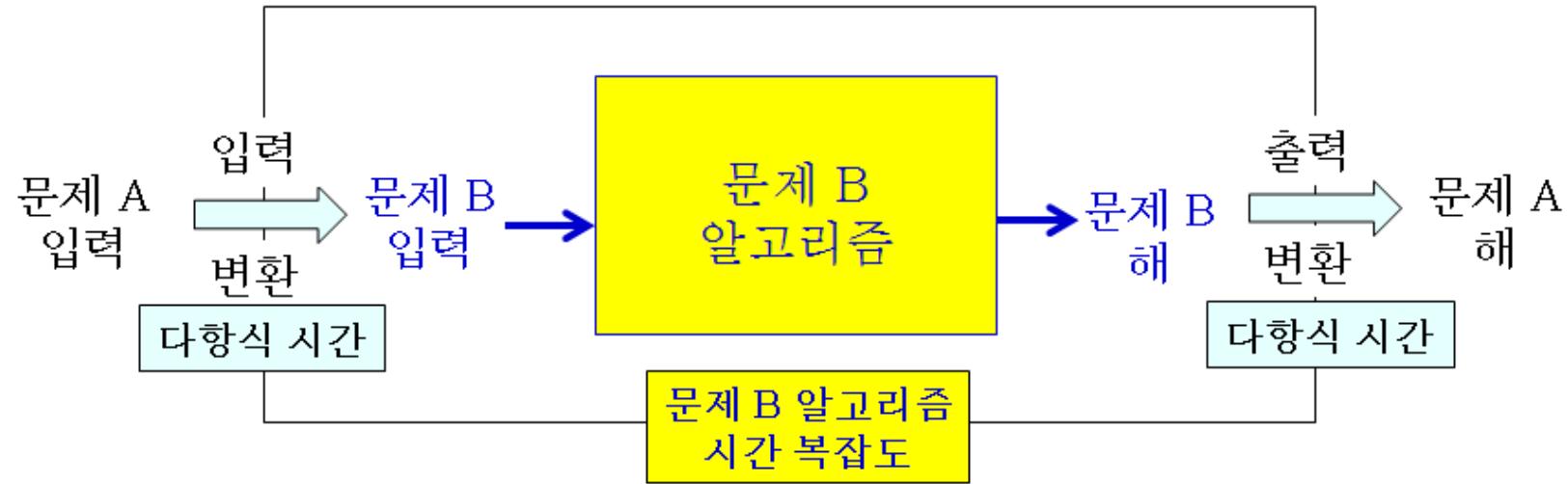
$$K = 105$$

$$t = s-2K = 250-(2 \times 105) = 250-210 = 40$$



- 문제를 변환하는 전 과정의 시간복잡도는 다음의 3단계의 시간복잡도의 합이다.
 - 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 단계는 단순한 입출력 변환이므로 다항식 시간에 수행된다.
- 따라서 문제 변환의 시간복잡도는 두 번째 단계의 시간복잡도에 따라 결정된다고 할 수 있다.
- 두 번째 단계가 다항식 시간이 걸리면, 문제 A도 다항식 시간에 해결된다.

문제A 알고리즘



- 문제 A와 문제 B 사이에 위와 같은 관계가 성립하면, 문제 A가 문제 B로 **다항식 시간에 변환** (polynomial time reduction) 가능하다고 한다.
- 그리고 만일 문제 B가 문제 C로 다항식 시간에 변환 가능하면, 결국 문제 A가 문제 C로 다항식 시간에 변환 가능하다.
- 이러한 추이 (transitive) 관계로 NP-완전 문제들이 서로 얹혀 있어서, **NP-완전 문제들 중에서 어느 한 문제만 다항식 시간에 해결되면, 모든 다른 NP-완전 문제들이 다항식 시간에 해결된다.**

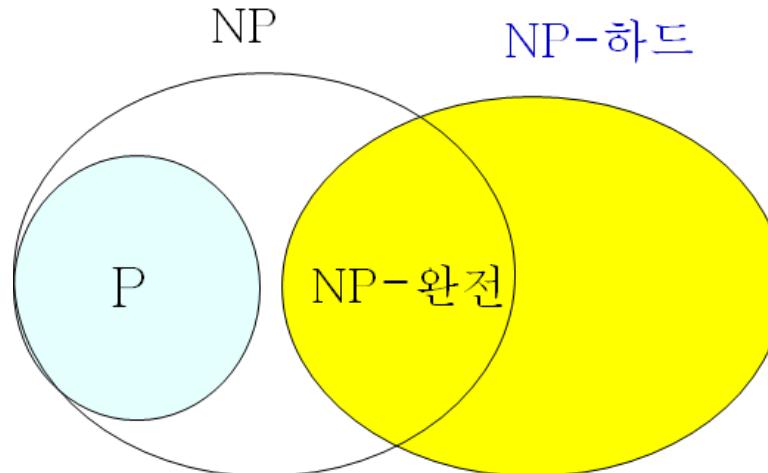
NP-hard Problem

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

어느 문제 A에 대해서, 만일 모든 NP 문제가 문제 A로 다향식 시간에 변환이 가능하다면, 문제 A는 **NP-하드 문제**이다

- 'hard'란:
 - 적어도 어떤 NP 문제보다 해결하기 어렵다는 뜻
 - 모든 NP 문제가 NP-하드 문제로 다향식 시간에 변환 가능하여야 함에도 불구하고, **NP-하드 문제는 반드시 NP 문제일 필요는 없다.**

- 따라서 다음과 같은 문제 집합들 사이의 관계가 이루어진다.



- NP-완전 문제는 NP-하드 문제이면서 동시에 NP 문제이다.

- NP-완전 문제를 정확하게 다음과 같이 다시 정의하면.
- 문제 A가 **NP-완전 문제**가 되려면,
 - 1) 문제 A는 NP 문제이고, 동시에
 - 2) 문제 A는 NP-하드 문제이다.

46

NP-완전 문제의 소개

Algorithm

2020-06-09

NP-완전 문제의 소개

- NP-완전 문제 집합에는 컴퓨터 분야뿐만 아니라 과학, 공학, 의학, 약학, 경영학, 정치학, 금융 심지어는 문화 분야 등에까지 광범위한 분야에서 실제로 제기되는 문제들이 포함되어 있다.
- 이러한 문제들 중에서 대표적인 NP-완전 문제들을 살펴본다.

SAT (Satisfiability)

- 부울 변수 (Boolean variable)들이 \vee (OR)로 표현된 논리식이 여러 개 주어질 때, 이 논리식들을 모두 만족시키는 각 부울 변수의 값을 찾는 문제이다.
- [예제] 부울 변수 w, x, y, z 에 대하여,

$$1) (w \vee y), (\neg w \vee x \vee z), (\neg x \vee \neg y \vee \neg z)$$

해: $w=true, x=true, y=false, z=true \text{ or } false$

$$2) (w \vee \neg x), (x \vee \neg y), (y \vee \neg w), (w \vee x \vee y), (\neg w \vee \neg x \vee \neg y)$$

해: 없음

부분 집합의 합 (Subset Sum)

- 주어진 정수의 집합 S 의 원소의 합이 K 가 되는 S 의 부분 집합을 찾는 문제이다.
- [예제] $S = \{20, 30, 40, 80, 90\}$ 이고, 합이 200이 되는 부분 집합을 찾고자 할 때,
- [해] $\{30, 80, 90\}$ 의 원소 합이 200이다.

분할 (Partition)

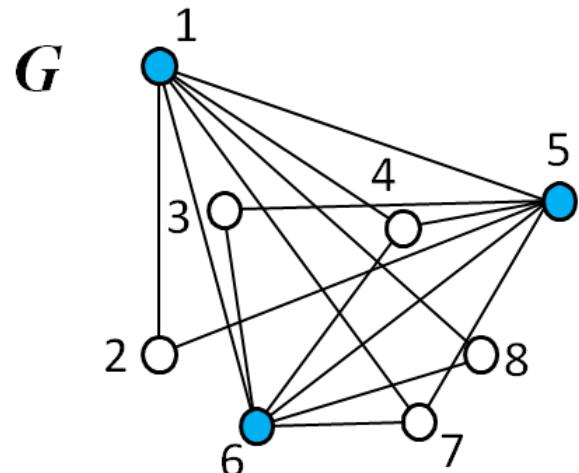
- 주어진 정수의 집합 S 를 분할하여 원소의 합이 같은 2개의 부분 집합을 찾는 문제이다.
- [예제] $S = \{20, 30, 40, 80, 90\}$ 일 때, S 를 2개의 합이 동일한 부분 집합으로 분할하면,
- [해] $X = \{20, 30, 80\}, Y = \{40, 90\}$; 각각의 부분 집합의 합이 130이다.

0-1 배낭 (Knapsack)

- 배낭의 용량이 C 이고, n 개의 물건의 각각의 무게와 가치가 w_i 와 v_i 일 때, 단, $i = 1, 2, \dots, n$, 배낭에 담을 수 있는 물건의 최대 가치를 찾는 문제이다. 단, 담을 물건의 무게의 합이 배낭의 용량을 초과하지 말아야 한다.
- [예제] $C = 20\text{kg}$, $w_1 = 12\text{kg}$, $w_2 = 8\text{kg}$, $w_3 = 6\text{kg}$, $w_4 = 5\text{kg}$ 이고, $v_1 = 20$, $v_2 = 10$, $v_3 = 15$, $v_4 = 25$ 라면,
- [해] 물건 2, 3, 4를 배낭에 담으면, 그 무게의 합은 $8+6+5 = 19\text{kg}$, 그 가치의 합은 $10+15+25 = 50$ 으로 최대가 된다.

정점 커버 (Vertex Cover)

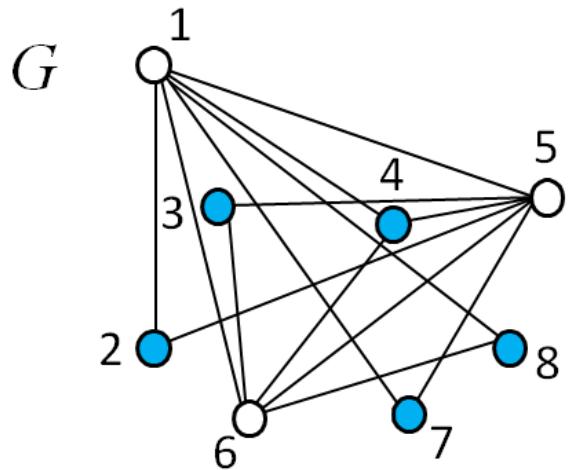
- 정점 커버란 주어진 그래프 $G=(V,E)$ 에서 각 선분의 양 끝점들 중에서 적어도 1개의 점을 포함하는 집합이다. 정점 커버 문제는 최소 크기의 정점 커버를 찾는 문제이다.



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

독립 집합 (Independence Set)

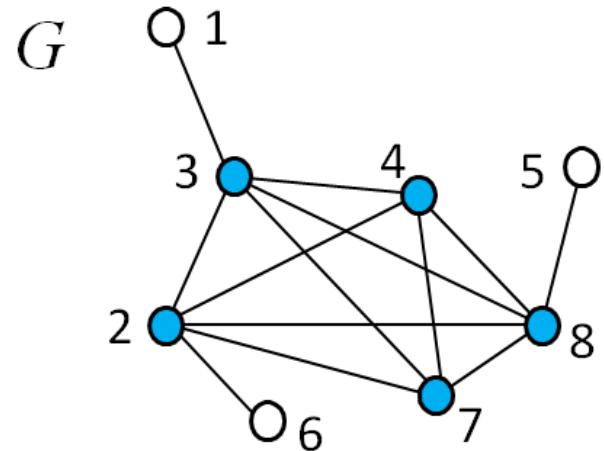
- 독립 집합이란 주어진 그래프 $G=(V,E)$ 에서 연결하는 선분이 없는 점들의 집합이다. 독립 집합 문제는 최대 크기의 독립 집합을 찾는 문제이다.



- [해] $\{2, 3, 4, 7, 8\}$ 은 서로 선분으로 연결 안 된 최대 크기의 독립 집합이다.

클리크 (Clique)

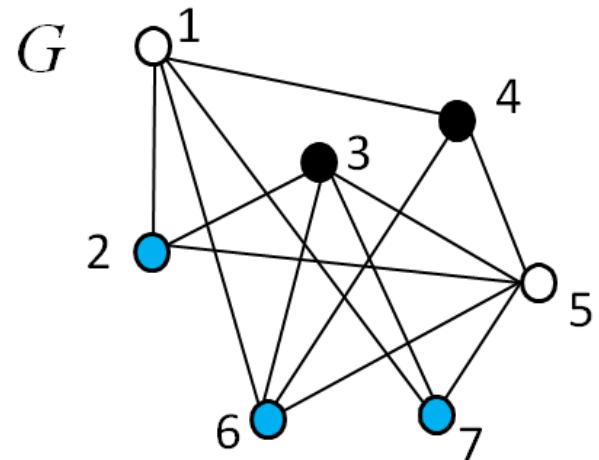
- 클리크란 주어진 그래프 $G=(V,E)$ 에서 모든 점들 사이를 연결하는 선분이 있는 부분 그래프이다. 클리크 문제는 최대 크기의 클리크를 찾는 문제이다.



- [해] $\{2, 3, 4, 7, 8\}$ 은 서로 선분으로 모두 연결된 최대 크기의 클리크이다.

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 문제이다.



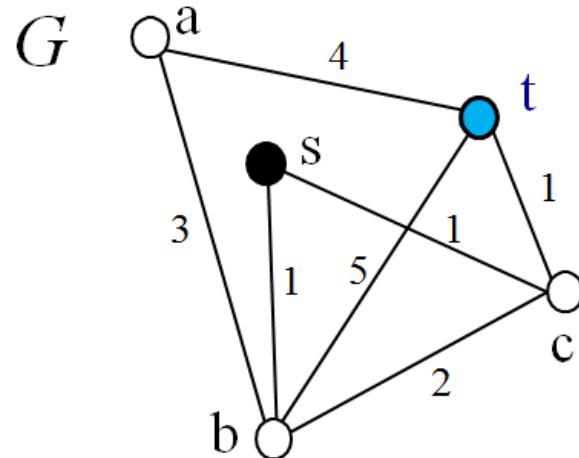
- [해] $\{1, 5\}$ 는 흰색, $\{3, 4\}$ 는 검은색, $\{2, 6, 7\}$ 은 파란색으로 칠한다. 3가지 색보다 적은 수의 색으로 이 그래프를 칠할 수는 없다.

집합 커버 (Set Cover)

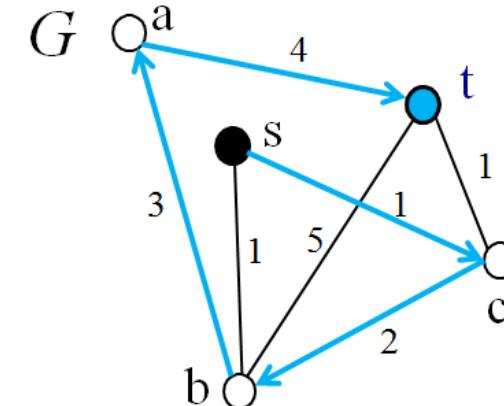
- 주어진 집합 $S = \{1, 2, 3, \dots, n\}$ 에 대해서 S 의 부분 집합들이 주어질 때, 이 부분 집합들 중에서 합집합하여 S 와 같게 되는 부분 집합들을 집합 커버라고 한다. 집합 커버 문제는 가장 적은 수의 부분 집합으로 이루어진 집합 커버를 찾는 문제이다.
- [예제] $S = \{1, 2, 3, 4, 5\}$, 부분 집합: $\{1, 2, 3\}, \{2, 3, 4\}, \{3, 5\}, \{3, 4, 5\}$ 라면,
- [해] $\{1, 2, 3\}$ 과 $\{3, 4, 5\}$ 를 합집합하면 S 가 되고, 부분 집합 수가 최소이다.

최장 경로 (Longest Path)

- 주어진 가중치 그래프 $G=(V,E)$ 에서 시작점 s 에서 도착점 t 까지의 가장 긴 경로를 찾는 문제이다. 단, 선분의 가중치는 양수이고, 찾는 경로에는 반복되는 점이 없어야 한다.

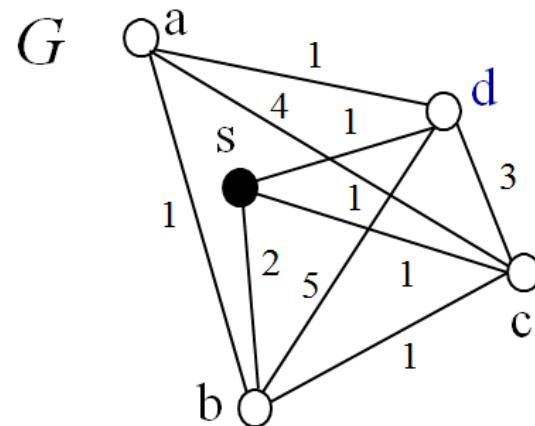


[해] $s \rightarrow c \rightarrow b \rightarrow a \rightarrow t$ 가 최장 경로
로서 그 길이는 10이다.

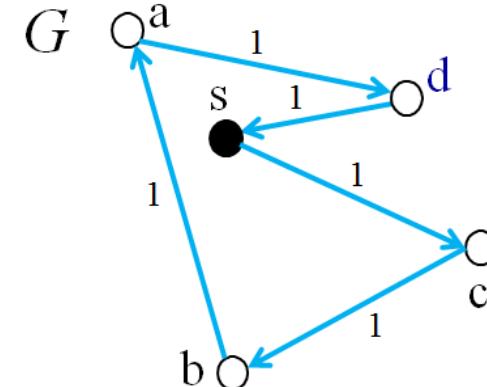


여행자 (Traveling Salesman) 문제

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

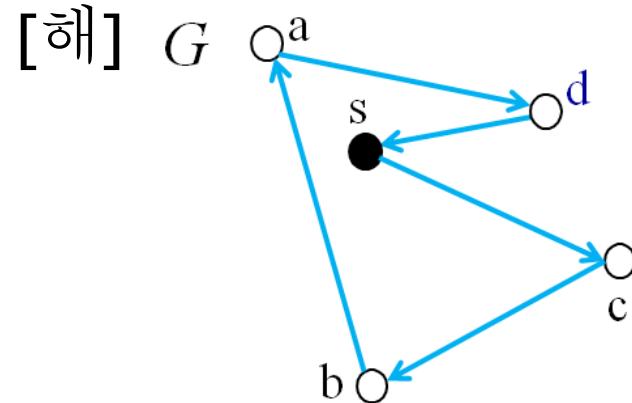
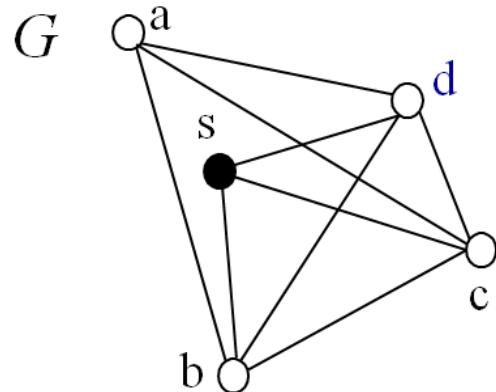


[해]



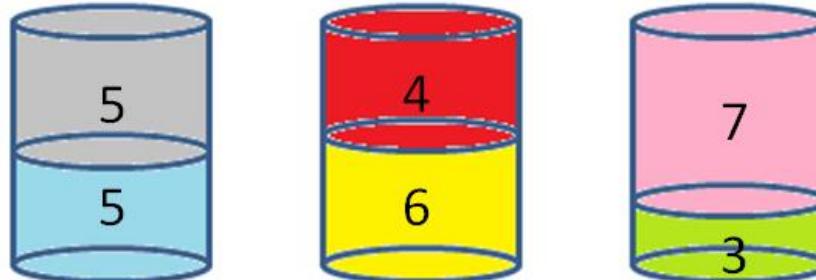
헤밀토니안 사이클 (Hamiltonian Cycle)

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



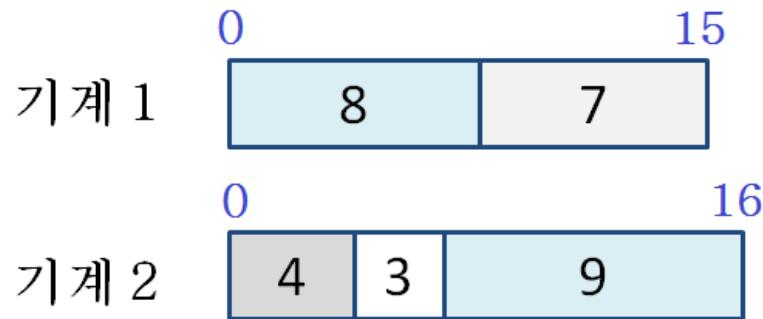
통 채우기 (Bin Packing)

- n 개의 물건이 주어지고, 통 (bin)의 용량이 C 일 때, 가장 적은 수의 통을 사용하여 모든 물건을 통에 채우는 문제이다. 단, 각 물건의 크기는 C 보다 크지 않다.
- [예제] 통의 용량 $C=10$ 이고, $n=6$ 개의 물건의 크기가 각각 5, 6, 3, 7, 5, 4이면,
- [해] 3개의 통을 사용하여 아래와 같이 채울 수 있다.



작업 스케줄링 (Job Scheduling)

- n 개의 작업, 각 작업의 수행 시간 t_i , 단, $i = 1, 2, 3, \dots, n$, 그리고 m 개의 동일한 성능의 기계가 주어질 때, 모든 작업이 가장 빨리 종료되도록 작업을 기계에 배정하는 문제이다.
- [예제] $n = 5$ 개의 작업이 주어지고, 각각의 수행 시간이 8, 4, 3, 7, 9이며, $m = 2$ 라면,
- [해] 아래와 같이 작업을 배정하면 가장 빨리 모든 작업을 종료시킨다.



62

NP-완전 문제들의 활용

Algorithm

2020-06-09

- 지금까지 소개한 NP-완전 문제는 다항식 시간에 하나의 문제에서 다른 문제로 변환 가능하다.
- 이러한 문제 변환은 살펴본 부분 집합의 합 문제를 분할 문제로 변환한 것같이 간단한 경우도 있고, 반면에 매우 복잡한 경우도 있다.

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.
- 다음은 앞서 설명한 각각의 NP-완전 문제가 직접 또는 간접적으로 활용되는 사례를 요약한 것이다.

SAT 문제

- 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)
- 인공지능에서의 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) 문제

- 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) 문제

- 부분 집합의 합 문제의 특별한 경우이다.
- 즉, 부분 집합의 합 문제에서 부분 집합의 합이 전체 원소의 합의 $1/2$ 이라고 하면 분할 문제와 동일하게 된다.
- 분할 문제를 보다 일반화하여 분할할 부분 집합 수를 2개에서 k 개로 확장시키면, 더욱 더 다양한 곳에 응용 가능하다.
- **Switching Network**에서 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) 문제

- 다양한 분야에서 의사 결정 과정에 활용된다.
- 원자재의 버리는 부분을 최소화시키는 분할
- 금융 분야에서 금융 포트폴리오 선택
- 자산 투자의 선택
- 주식 투자
- 다차원 경매 (Combinatorial Auction)
- 암호학 분야에서 암호 생성 (Merkle–Hellman Knapsack)
- 게임 스도쿠 (Sudoku) 등에 활용된다.



		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			

m)

정점 커버 (Vertex Cover) 문제

- 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) 등에 활용된다.

집합 커버 (Set Cover) 문제

- 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 구매업체 선정
- 기업의 경력 직원 고용 등에도 활용된다.



독립 집합 (Independent Set) 문제

- 컴퓨터 비전 (Computer Vision)
- 패턴 인식 (Pattern Recognition)
- 정보/코딩 이론 (Information/Coding Theory)
- 지도 레이블링 (Map Labeling)
- 분자 생물학 (Molecular Biology)
- 스케줄링 (Scheduling)
- 회로 테스트
- CAD 등에 활용된다.



클리크 (Clique) 문제

- 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 활용된다.

그래프 색칠하기 (Coloring) 문제

- 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 색칠하기 와 동일하다.
- 생물학에서 생체 분석
- 고고학 자료 분석에 응용된다.



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

- 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) 등에 활용된다.



통 채우기 (Bin Packing) 문제

- 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 공급망 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) 등에 활용된다.

작업 스케줄링 (Job Scheduling) 문제

- 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 있는 것이다.
- 다항식 시간복잡도를 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.