10주차 1차시 NP-완전문제 이해

【학습목표】

1. NP-완전문제를 이해할 수 있다.
2. NP-완전성증명을 이해할 수 있다.

학습내용1 : 기본개념

1. 문제의 난이도

① 쉬운(tractable) 문제 : $O(n^k)$와 같이 다항식 시간 알고리즘이 존재하는 문제.

② 어려운(intractable) 문제 : 다항식 시간보다 더 많은 시간을 요구하는 문제.

* 쉬운 문제?, 어려운 문제?
- 다항식 시간 알고리즘이 아직 만들어지지도 않았고, 그렇다고 해서 그러한 알고리즘이 존재하지 않는다고 증명되지도 않은 문제

2. 문제의 종류

⦿ 판정 문제(decision problem) : 문제의 해가 예/아니오 형태로 나오는 문제
⦿ 최적화 문제(optimization problem)

3. NP-완전문제 의 예

0/1 배낭(knapsack)문제

n개의 물체, 배낭의 용량 C
물체의 무게 = $(w_1, w_2, \ldots, w_n)$
물체의 이익 = $(p_1, p_2, \ldots, p_n)$

$\sum_{i=1}^{n} w_i x_i \le C$ 이면서

최적화 문제 $\sum_{i=1}^{n} p_i x_i$를 최대로 하는 x_i를 구하라

판정 문제

$\sum_{i=1}^{n} p_i x_i \ge R$인 x_i를$(1 \le i \le n)$들의 값이 존재?

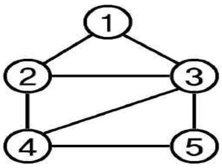
* CNF 만족성 문제

- 정규곱형(CNF)으로 주어진 논리식의 리터럴들에 참 또는 거짓 값을 적절히 지정하여 전체 논리식의 값을 참으로 만들 수가 있는지를 판정하는 문제

$$(x_1 \vee x_3 \vee x_5) \wedge (x_1 \vee \overline{x_2} \vee x_4) \wedge (\overline{x_3} \vee x_4 \vee x_5) \wedge (x_2 \vee x_3 \vee \overline{x_5})$$

4. 해밀토니언 사이클 문제

- 무방향 그래프 G가 주어졌을 때 G가 해밀토니언 사이클을 갖는지를 판정하는 문제
  - G의 한 정점 A에서 출발하여 모든 정점을 정확히 한번 통과한 후 A로 다시 되돌아오는 경로



|V|=n
→ O(n!) → O(n^n)

학습내용2 : NP-완전성과 변환성

1. 비결정론적 알고리즘

✓ **결정론적 알고리즘**

- 결과가 유일하게 정의된 연산만을 써서 만들어진 알고리즘

✓ **비결정론적 알고리즘**

- 연산 결과가 상황에 따라 달라질 수 있는 연산을 써서 만들어진 알고리즘
- 결과가 유일하지 않은 연산을 갖는 것을 허용
  → 지정된 연산 결과의 집합 중 하나를 선택할 수 있도록 허용

✓ **비결정론적 알고리즘에 추가된 연산**

- choice(S)
  - 집합 S의 원소 중 최선의 하나를 임의로 선택
  - 알고리즘을 성공적으로 수행하는 선택이 있을 경우에는 항상 그 선택 중 하나를 택함
  - 선택이 없을 경우에만 실패로 끝남
- failure
  - 알고리즘이 실패로 끝났음을 알림
- success
  - 알고리즘이 성공적으로 끝났음을 알림

2. 비 결정론적 알고리즘 예

A[0..n−1]에서 x를 찾는 비결정론적 알고리즘

```
j = choice (0..n−1);
if (A[j] = x)  { printf( j );  success; }
printf ('−1');  failure;
```

```
비결정론적-정렬(A, n) {
/* 입력:A(배열), n: A의 원소의 수, 출력:B(배열), nro의 양수정렬. */
int B(n), i, j;
B = -1;
for (i=1; i<= n; i++) {
j= choice (1..n);
if (B[j] != -1) failure;
B[j] = A[i];
}
/* 숫자들이 제대로 정렬되었는지 검사한다. */
for (i=1; i<=n; i++) {
if( B[i] > B[i+1] ) failure;
}
printf(B);
success;
}
```

비 결정론적 알고리즘 예 (CNF 만족성)

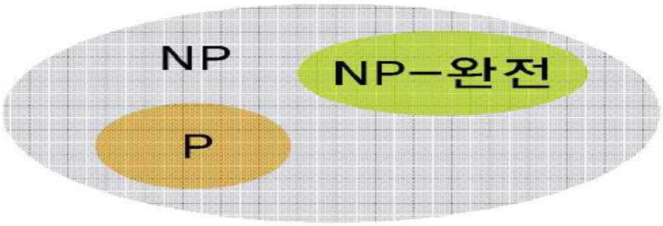
```
비결정론적-CNF 만족성 (E, n)
/* 입력: E : 논리식,  n : 서로 다른 논리 변수의 수
   출력: 논리식이 참이 될 수 있는 경우 성공
              아니면 실패
          주어진 논리식 E가 참이 될 수 있는지 결정 */
{
   boolean x(n)
   for (i = 1; i ≤ n; i++)  {
       x[i]= choice (true, false)
   }
   if (E(x[1],  ····  , x[n]) = true) success;
   else  failure ;
}
```

3. P와 NP의 정의

① 부류 P의 정의 : 결정론적 알고리즘에 의해 다항식 시간에 풀 수 있는 모든 판정 문제 집합

② 부류 NP의 정의 : 비결정론적 알고리즘에 의해 다항식 시간에 풀 수 있는 모든 판정 문제집합
- 비결정론적의 의미 : 여러 가지 중에서 하나를 택해야 할 때 스스로 최선의 선택을 할 수 있는 능력
- NP문제 : 여러 경우에서 선택을 해야 하는 경우 스스로 최선의 선택을 하여 다항식 시간에 풀릴 수 있는 것.

③ P⊆NP, P≠NP(?)

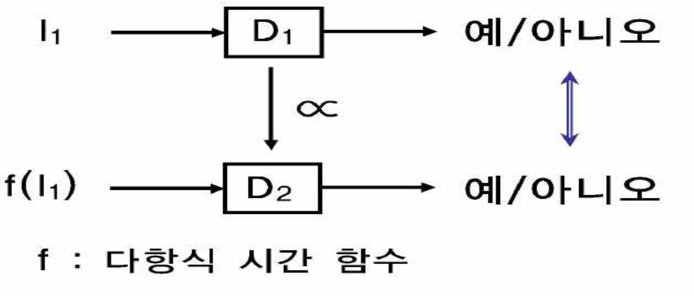


P≠NP일 때 문제의 관계

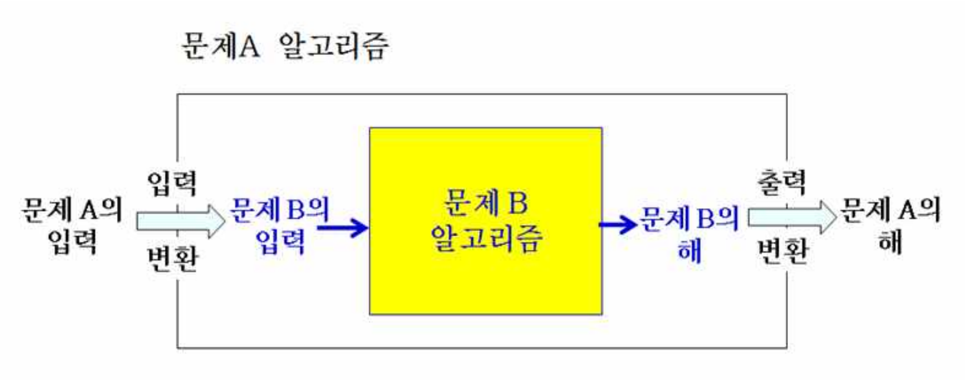
4. 변환

- ✓ **문제 B에 대한 알고리즘으로 문제 A를 풀 수 있을 때, 문제 A를 문제 B로 변환(transformation, reduction)할 수 있다고 표현**
  - 다항식 시간 변환 : A ∝ B
    - 변환에 다항식 시간이 소요되는 경우
    - '∝'의 기호로 표시
- ✓ **추이적** : $(D_1 \propto D_2) \wedge (D_2 \propto D_3) \Rightarrow D_1 \propto D_3$

* 다항식의 시간 변환



* 문제의 변환 과정

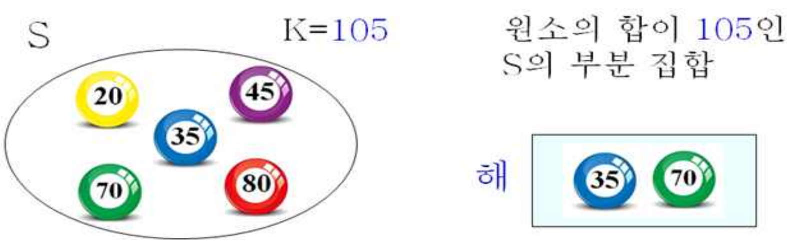


예)

● 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 문제이다.

● 예를 들어, S={20, 35, 45, 70, 80}이고, K=105이라면, {35, 70}의 원소의 합이 105가 되므로, 문제의 해는 {35, 70}이다.

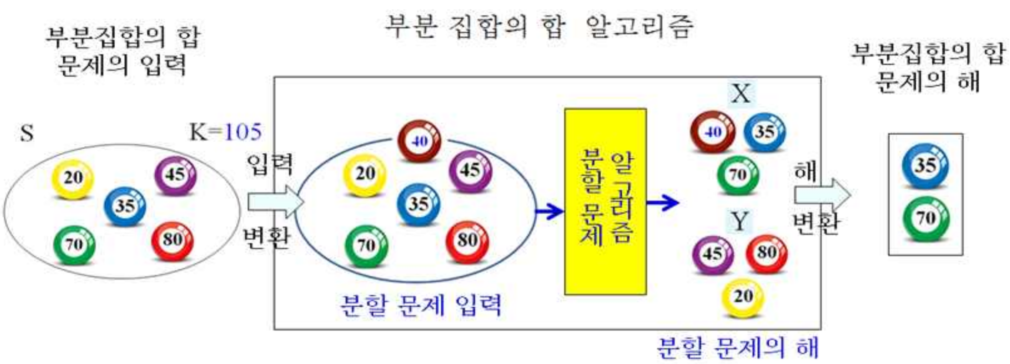


● 문제 B = 분할 문제: 정수의 집합 S에 대해, S를 분할하여 원소들의 합이 같은 2개의 부분 집합을 찾는 문제이다.
● 예를 들어, S={20, 35, 45, 70, 80}이 주어지면, X = {20, 35, 70}과 Y = {45, 80}이 해이다. 왜냐하면 X의 원소의 합이 20+35+70 = 125이고, Y의 원소의 합도 45+80 = 125이기 때문이다

● 부분 집합의 합 문제의 입력인 집합 S를 분할 문제의 입력으로 변환할 때 <u>t를 집합 S에 추가</u>한다.
t = s - 2K
● 단, s는 집합 S의 모든 원소의 합이다.
● 부분 집합의 합 문제를 해결하기 위해서, 집합 S' = S∪{t }를 입력으로 하는 분할 문제를 위한 알고리즘을 이용한다.
● 분할 문제 알고리즘의 해인 2개의 집합 X와 Y에 대해, X에 속한 원소의 합과 Y에 속한 원소의 합이 같으므로, 각각의 합은 (s-K)이다.
● 왜냐하면 새 집합 S'의 모든 원소의 합이 s+t = s+(s-2K) = 2s-2K이고, (2s-2K)의 1/2이면 (s-K)이기 때문이다

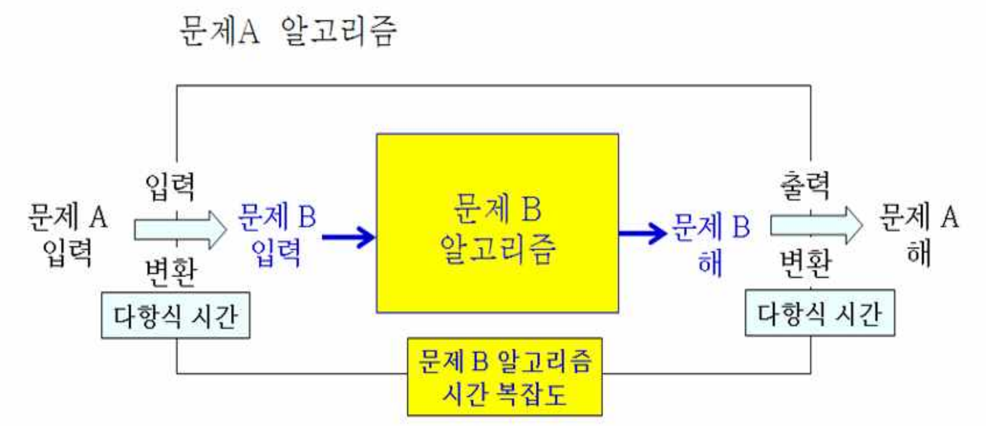
● 따라서 분할 문제의 해인 X와 Y 중에서 t를 가진 집합에서 t를 제거한 집합이 부분 집합의 합 문제의 해가 된다.
● 왜냐하면 만일 X에 t가 속해 있었다면, X에서 t를 제외한 원소의 합이 (s-K)-t = (s-K)-(s-2K) = s-K-s+2K = K가 되기 때문이다.
● 그러므로 부분 집합의 합 문제의 해는 바로 X-t이다.

● 다음의 그림은 부분 집합의 합 문제를 분할 문제로 변환하여 해결하는 것을 위의 예제를 통해서 보이고 있다. 여기서 s, K, t 값은 다음과 같다.
● s = 20+35+45+70+80 = 250
● K = 105
● t = s-2K = 250-(2x105) = 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.



● 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 얽혀 있어서, NP-완전 문제들 중에서 어느 한 문제만 다항식 시간에 해결되면, 모든 다른 NP-완전 문제들이 다항식 시간에 해결된다.

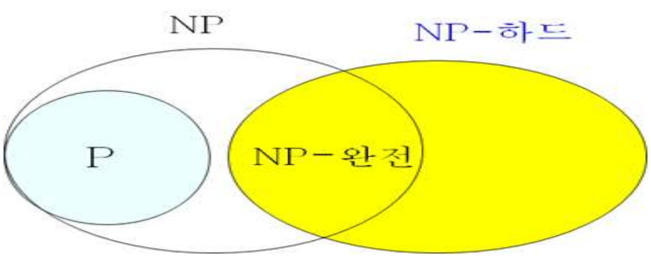
5. NP-하드 (hard) 문제

● 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 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.

따라서 다음과 같은 문제 집합들 사이의 관계가 이루어진다.



NP-완전 문제는 NP-하드 문제이면서 동시에 NP 문제이다.

6. N-P하드와 N-P완전

✓ **NP-하드 문제**

- NP의 모든 문제가 어떤 문제 A로 다항식 시간 변환되는 경우

✓ **NP-완전 문제**

- NP의 모든 문제가 어떤 문제 A로 다항식 시간 변환되고, A가 NP에 속하는 경우

학습내용3 : NP-완전성 증명

● 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.

1. 부분 집합의 합 (Subset Sum)

● 주어진 정수의 집합 S의 원소의 합이 K가 되는 S의 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.

2. 분할 (Partition)

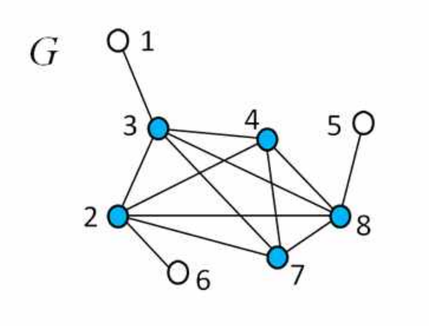
● 주어진 정수의 집합 S를 분할하여 원소의 합이 같은 2개의 부분 집합을 찾는 문제이다.
● [예제] S = {20, 30, 40, 80, 90 }일 때, S를 2개의 합이 동일한 부분 집합으로 분할하면,
● [해] X = {20, 30, 80 }, Y = {40, 90 }; 각각의 부분 집합의 합이 130이다.

3. 0-1 배낭 (Knapsack)

● 배낭의 용량이 C이고, n개의 물건의 각각의 무게와 가치가 w_i와 v_i일 때, 단, i = 1, 2, …, n, 배낭에 담을 수 있는 물건의 최대 가치를 찾는 문제이다. 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 = 25라면,
● [해] 물건 2, 3, 4를 배낭에 담으면, 그 무게의 합은 8+6+5 = 19kg, 그 가치의 합은 10+15+25 = 50으로 최대가 된다.

4. 클리크 (Clique)

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



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

5. 기타 문제

- CNF 만족성 문제, 해밀토니언 사이클 문제, 버텍스 커버 문제, 파티션 문제, 3-CNF 만족성 문제, 궤 채우기,문제, TSP (외판원 방문 판매 문제) 등

【학습정리】

1. 기본 개념
- 쉬운(tractable) 문제와 어려운(intractable) 문제: 다항식 시간 알고리즘이 존재하는 문제를 쉬운(tractable) 문제라고 하고, 다항식 시간을 초과하는 문제를 어려운(intractable) 문제라고 한다.
- 판정(decision) 문제와 최적화(optimization) 문제
  → 판정 문제: 문제의 해가 "예"/"아니오" 형태로 나오는 문제
  → 최적화 문제: 최소치나 최대치를 구하는 형태의 문제
- 결정론적(nondeterministic) 알고리즘: 그 결과가 미리 정해져 있지 않고 실행시에 가능한 연산 결과 중 최선의 하나를 스스로 선택할 수 있다는 성질이 허용된 알고리즘
- 부류 P와 부류 NP의 정의
  → 부류 P: 결정론적 알고리즘에 의하여 다항식 시간에 풀릴 수 있는 모든 판정 문제의 집합
  → 부류 NP: 비결정론적 알고리즘에 의하여 다항식 시간에 풀릴 수 있는 모든 판정 문제의 집합

2. NP-완전성과 변환성(reducibility)
- 문제 B에 대한 알고리즘으로 문제 A를 풀 수 있을 때, 문제 A를 문제 B로 변환(reduction)할 수 있다고 하고, 이러한 변환에 다항식 시간이 소요되면 다항식 시간 변환이라 하고 '∝'의 기호로 나타낸다.
- 어떤 부류에 속하는 모든 문제가 그 부류에 속하는 어떤 문제 R로 다항식 시간 변환 가능하다면 문제 R은 그 부류의 완전(complete) 문제이다. 완전인 문제는 그 부류의 가장 어려운 문제인 것으로 생각할 수 있다.
- 어떤 부류의 모든 문제가 문제 R로 다항식 시간 변환될 수 있지만 R이 그 부류에 속한다는 조건이 없다면 R은 그 부류에 대한 하드(hard)인 문제이다. 어떤 부류의 하드인 문제는 그 부류에 속하는 어떤 문제보다도 풀기 어렵거나 또는 비슷한 정도로 풀기 어렵다.
- NP-하드, NP-완전 문제: NP의 모든 문제가 어떤 문제 A에 다항식 시간 변환된다면 문제 A는 NP-하드(hard)한 문제이며 또 A가 NP에 속한다면 그 문제는 NP-완전이다.
  → NP-완전 문제는 쉬운 문제인가 어려운 문제인가? 아직 증명은 되지 않았으나 대부분의 학자들은 어려운 문제일 것으로 믿고 있다.

3. NP-완전성 증명
- 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.
부분 집합의 합, 분할 (Partition), 0/1 배낭 문제, CNF 만족성 문제, 해밀토니언 사이클 문제, 클리크 판정문제, 버텍스 커버 문제, 파티션 문제, 3-CNF 만족성 문제, 궤 채우기 문제, TSP (외판원 방문 판매 문제) 등