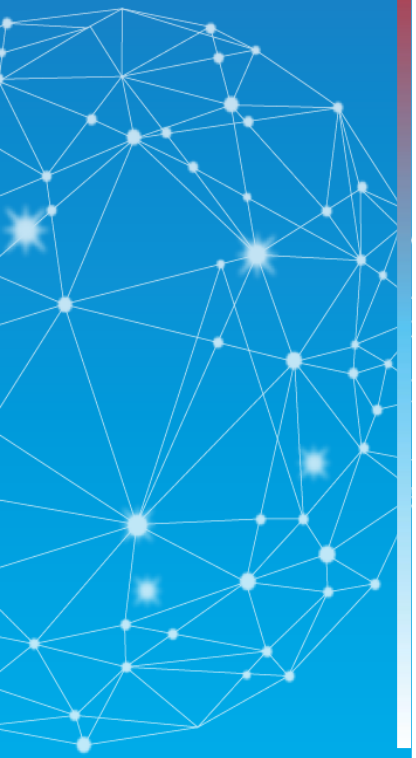
교육부 Ministry of Education

NRF 한국연구재단 National Research Foundation of Korea

5장. 정수계획법

- 참고문헌: 최적 의사결정을 위한 경영과학, 권수태외 5인, 청람, 2018
  알기쉬운 알고리즘, 양성봉, 생능, 2021

Contents



- 정수계획법 개요
- 정수계획법 예제
- 정수계획법의 해법(열거법, 분단탐색법)
- 배낭문제

5. 정수계획법

5-1 정수계획법 개요 및 예제

5-2 정수계획법의 해법(열거법, 분단탐색법)

5-3 배낭문제

5.1 정수계획법 개요

❖ 예제

공유가구㈜에서는 목재, 가죽 , 섬유 등 세 가지 원료를 사용하여 침대와 소파 등 두 가지 가구를 만들고 있다. 앞으로 1달 간 사용하기 위하여 확보한 원료는 목재가 $16m^3$, 가죽은 $12m^2$, 섬유는 $6m^2$ 이다. 침대를 하나 만드는 데는 목재가 $2m^3$, 가죽은 $1m^2$, 섬유는 $1m^2$ 가 필요하고, 소파를 하나 만드는 데는 목재가 $1m^3$, 가죽은 $1m^2$ 필요하고 섬유는 필요하지 않다. 침대는 개당 100만원에 판매하여 원료비 등의 비용을 제외하고 15만원의 이익을 내고 있으며, 소파는 개당 60만원에 판매하여 원료비 등의 비용을 제외하고 10만원의 이익을 내고 있다. 이익을 최대로 만들기 위해서는 1달 간 침대와 소파를 각각 몇 개씩 만들어야 하는가?

5.1 정수계획법 개요

❖ 예제

➢ 의사결정변수의 결정

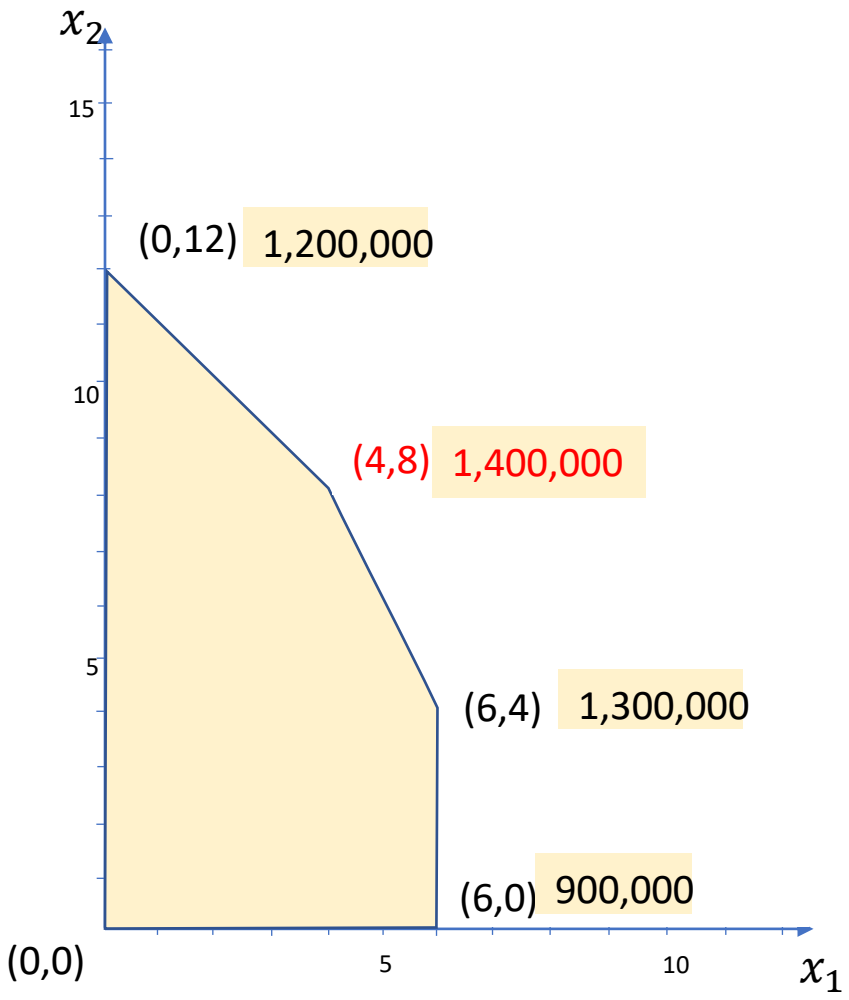
x_1 = 침대생산량

x_2 = 소파생산량

$$Max \ 150{,}000x_1 + 100{,}000x_2$$

$$s.t.$$

$$\begin{aligned} 2x_1 + x_2 &\le 16 \\ x_1 + x_2 &\le 12 \\ x_1 &\le 6 \\ x_1, x_2 &\ge 0 \end{aligned}$$



5.1 정수계획법 개요

❖ 예제

공유가구㈜에서는 목재, 가죽 , 섬유 등 세 가지 원료를 사용하여 침대와 소파 등 두 가지 가구를 만들고 있다. 앞으로 1달 간 사용하기 위하여 확보한 원료는 목재가 16m^3 , 가죽은 18m^2, 섬유는 6m^2 이다. 침대를 하나 만드는 데는 목재가 2m^3, 가죽은 1m^2, 섬유는 1m^2 가 필요하고, 소파를 하나 만드는 데는 목재가 1m^3, 가죽은 2m^2 필요하고 섬유는 필요하지 않다. 침대는 개당 100만원에 판매하여 원료비 등의 비용을 제외하고 15만원의 이익을 내고 있으며, 소파는 개당 60만원에 판매하여 원료비 등의 비용을 제외하고 10만원의 이익을 내고 있다. 이익을 최대로 만들기 위해서는 1달 간 침대와 소파를 각각 몇 개씩 만들어야 하는가?

5.1 정수계획법 개요

❖ 예제 (실수해)

➢ 의사결정변수의 결정

x_1 = 침대생산량

x_2 = 소파생산량

$$Max \ \ 150{,}000x_1 + 100{,}000x_2$$

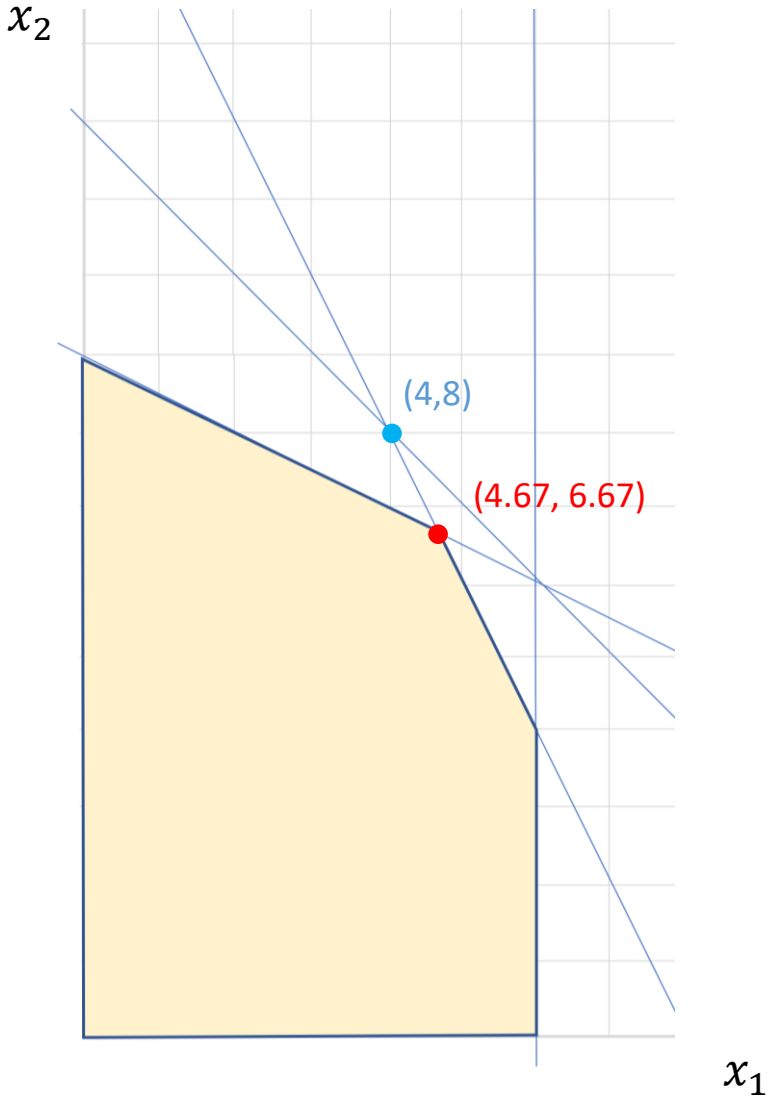
$$s.t.$$

$$2x_1 + x_2 \leq 16$$

$$x_1 + 2x_2 \leq 18$$

$$x_1 \leq 6$$

$$x_1, x_2 \geq 0$$



5.1 정수계획법 개요

❖ 예제 (정수해)

➢ 의사결정변수의 결정

x_1 = 침대생산량

x_2 = 소파생산량

$$Max \ \ 150{,}000x_1 + 100{,}000x_2$$

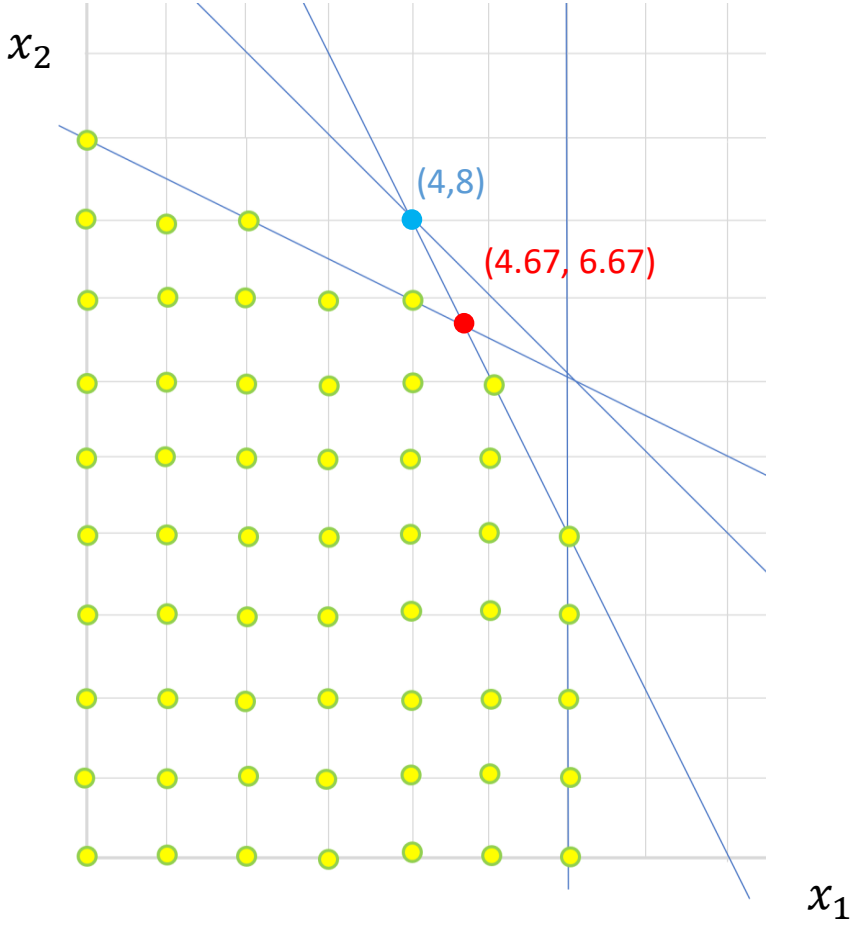
$$s.t.$$

$$2x_1 + \ \ x_2 \ \leq 16$$

$$x_1 + 2x_2 \ \leq 18$$

$$x_1 \ \ \leq \ \ 6$$

$$x_1, x_2 \geq 0$$



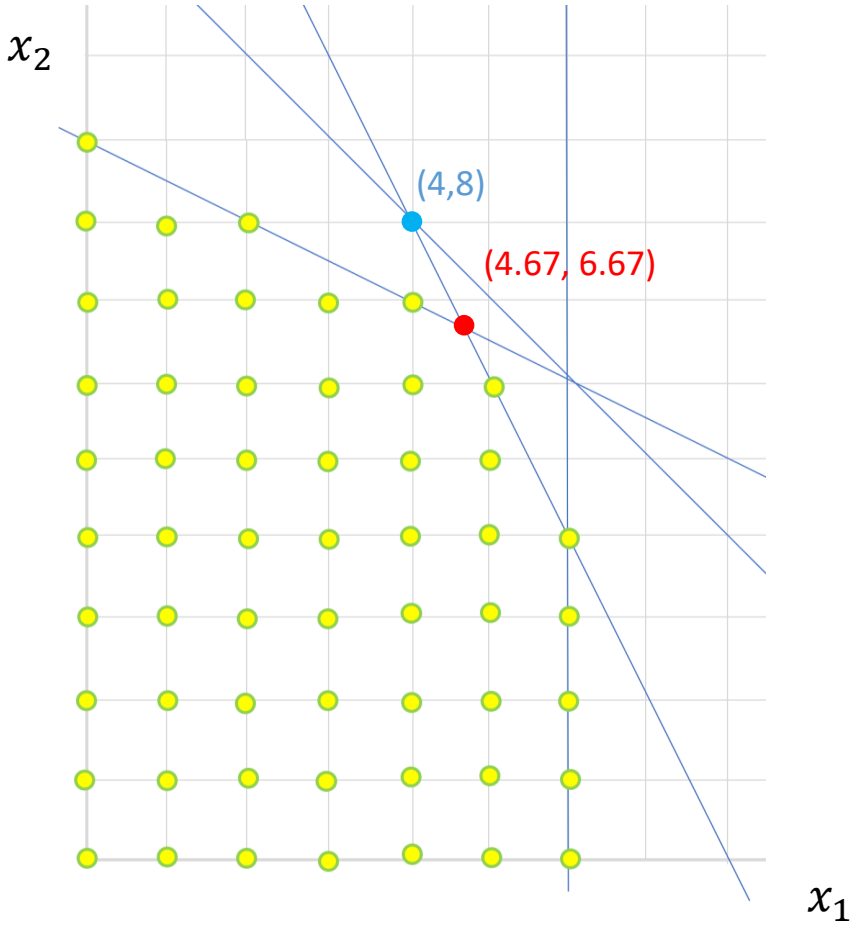
5.1 정수계획법 개요

❖ 예제

➢ 의사결정변수는 정수

✓ 반올림 -> (5,7)

✓ 내림 -> (4,6) 최적해?



만약 선형계획법에 의한 최적해가
정수해로 나오게 된다면 다행,
선형계획법에 의한 최적해를
올림 혹은 내림 처리하는 과정이
변수가 많아질수록 매우 복잡해지는
문제가 존재,
구해진 정수가능해가 최적해인지 여부를
판단하기 어려움

따라서, 정수해 조건이 붙는 경우에는 선형계획법과 다른 정수계획모형을 위한 해법이 필요

5.1 정수계획법 개요

❖ 정수계획모형

➢ 제약식은 선형계획모형과 동일하고 변수만 정수해 조건이 붙는 경우

$$
\begin{aligned}
&Max \;\; 150{,}000x_1 + 100{,}000x_2 \\
&s.t. \\
&\quad 2x_1 + \;\; x_2 \;\leq 16 \\
&\quad\; x_1 + 2x_2 \;\leq 18 \\
&\quad\; x_1 \qquad\;\; \leq \;\; 6 \\
&\quad\; x_1, x_2 \in \{1, 2, \cdots\}
\end{aligned}
$$

➢ 종류

✓ 순수정수계획모형 (Pure Integer Programming Model) : 모든 변수가 정수

✓ 혼합정수계획모형 (Mixed Integer Programming Model) : 일부 변수가 정수

✓ 이진정수계획모형 (Binary Integer Programming Model) : 모든 변수가 정수 중 0,1

5.2 정수계획법 예제

❖ 배낭문제 (Knapsack Problem)

- 전통적 정수계획모형으로서 주어진 아이템들을 하나의 배낭에 최대한 할당하는 문제를 의미
- 각각의 아이템은 크기와 수익 정보를 가지고 있으며 배낭 역시 최대 할당가능 크기 정보를 가지고 있음
- 따라서, 아이템을 배낭에 할당하는 조합에 따라 전체 수익이 달라지게 됨

5.2 정수계획법 예제

❖ 배낭문제 (Knapsack Problem)

➢ 기호정의

✓ K : 아이템(제품)의 집합

✓ k : 각각의 아이템, $k \in K$

✓ x_k : 의사결정 변수 (배낭에 할당되는지 여부를 나타냄)

✓ C : 배낭의 크기

✓ c_k : 아이템의 크기 정보

✓ p_k : 아이템의 수익 정보

$$Max \quad \sum_{k \in K} p_k x_k$$

$$s.t. \quad \sum_{k \in K} c_k x_k \leq C$$

$$x_k \in \{0, 1\}$$

5.2 정수계획법 예제

❖ 복수배낭문제 (Multiple Knapsack Problem)

➢ 기호정의

- ✓ K : 아이템(제품)의 집합 (k : 각각의 아이템, $k \in K$)
- ✓ I : 배낭의 집합
- ✓ x_{ik} : 의사결정 변수 (아이템이 어떤 배낭에 할당되는지 여부를 나타냄)
- ✓ C_i : 배낭 i 의 크기
- ✓ c_k : 아이템의 크기 정보
- ✓ p_k : 아이템의 수익 정보

$$Max \quad \sum_{i \in I} \sum_{k \in K} p_k x_{ik}$$

$$s.t. \quad \sum_{k \in K} c_k x_{ik} \leq C_i$$

$$x_{ik} \in \{0,1\}, \quad \forall i \in I, k \in K$$

5.2 정수계획법 예제

❖ 할당 문제 (Assignment Problem)

- 수송 및 할당에서 다룬 할당문제는 하나의 집합에 속한 아이템을 다른 집합의 아이템에 할당하는 문제를 의미한다. 예를 들어 수행해야 하는 일이 n개 있고, 이를 수행할 수 있는 인력이 m명 있다고 하자.
- 수행해야 하는 일의 개수가 인력보다 적고 (n<m), 각각의 인력은 최대 하나의 일만 수행할 수 있다고 가정하자.
- 각각의 인력은 경험 및 숙련도에 의해 일을 수행하는데 있어 효율성이 달라질 수 있어 인력 $i \in I$ 가 일 $j \in J$ 를 수행하는데 드는 비용 C_{ik} 가 정의되어 있다.
- 전체 비용을 최소화하면서 인력들을 일에 할당하는 문제

5.2 정수계획법 예제

❖ 할당 문제 (Assignment Problem)

➢ n < m

$$Max \quad \sum_{i \in I} \sum_{j \in J} C_{ij} x_{ij}$$

$$s.t.$$

$$\sum_{i \in I} x_{ij} = 1$$

$$\sum_{j \in J} x_{ij} \leq 1$$

$$x_{ij} \in \{0, 1\}, \quad \forall i \in I, j \in J$$

5.2 정수계획법 예제

❖ 할당 문제 (Assignment Problem)

➢ n = m (이진 매칭문제)

$$Max \quad \sum_{i \in I} \sum_{j \in J} C_{ij} x_{ij}$$

$$s.t.$$

$$\sum_{i \in I} x_{ij} = 1$$

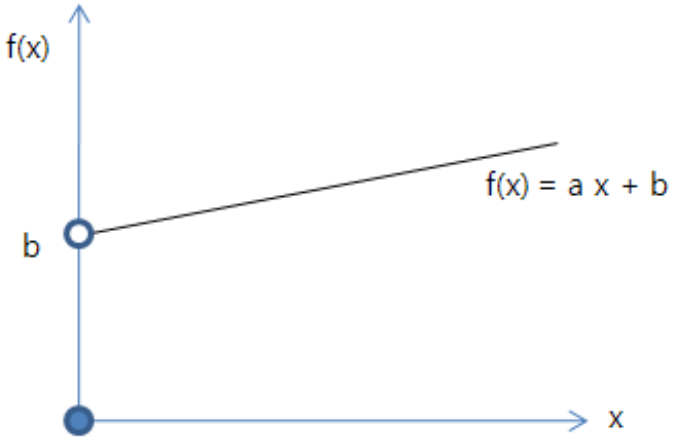
$$\sum_{j \in J} x_{ij} = 1$$

$$x_{ij} \in \{0,1\}, \quad \forall i \in I, j \in J$$

5.2 정수계획법 예제

❖ 고정비용 문제 (Fixed Charge Problem)

➢ 기업경영에 있어 비용 발생을 살펴보면 고정비와 변동비로 구분되는 경우가 많다. 고정비는 일을 수행하면 고정적으로 발생하는 비용으로 일을 수행하지 않으면 발생하지 않는다. 변동비용은 일을 수행할 경우 수반되는 자원에 비례하여 증가하는 비용



$$f(x) = \begin{cases} ax + b, & if\ x > 0 \\ 0 \quad\quad , & if\ x = 0 \end{cases}$$

5.2 정수계획법 예제

❖ 고정비용 문제 (Fixed Charge Problem)

- 고정비와 변동비를 하나의 함수로 표현할 경우, 이는 선형계획모형의 기본가정인 선형성과 연속성에 위배
- 이러한 경우 이진 정수 변수를 도입하여 정수계획모형으로 모형화

$$Min\ f(x) = ax + by$$

$$s.t. \qquad x \leq My$$

$$y = 0 \quad or \quad 1$$

$$f(x) = \begin{cases} ax + b, & if\ x > 0 \\ 0 \quad , & if\ x = 0 \end{cases}$$

5.2 정수계획법 예제

❖ 고정비용 문제 (Fixed Charge Problem)

- 공장 위치 선정 문제
  - 후보위치의 집합 $J = \{1, \cdots, n\}$와 수요지 집합 $I = \{1, \cdots, m\}$ 가 주어져 있을 때 전체 수요지에 물건을 공급하면서 비용을 최소화하는 후보지를 결정하는 문제
  - 공장 후보지 $j \in J$ 에 공장을 건설할 경우 고정비용 f_j가 소요되고, 이후 공장후보지 j 에서 수요지 i 에 물건을 공급할 때 변동비용 c_{ij} 가 비용으로 지출된다고 할 때 최적의 후보지를 결정하기 위해서는 의사결정 변수를 먼저 정의해야 한다.
  - 공장 후보지 j 에 공장을 건설할 경우 $z_j = 1$ 이라고 하고, 공장 후보지 j 에서 수요지 i 에 물건을 공급한다면 $x_{ij} = 1$ 이 된다고 하자. 모든 변수는 이진 정수 변수이다. 그렇다면 전체 문제는 정수계획모형으로 모형화된다.

5.2 정수계획법 예제

❖ 고정비용 문제 (Fixed Charge Problem)

➢ 공장 위치 선정 문제

$$Min \quad \sum_{j \in J} f_j \, z_j + \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij}$$

$$s.t.$$

$$\sum_{j \in J} x_{ij} = 1$$

$$\sum_{i \in I} x_{ij} \leq M z_j$$

$$z_j, \; x_{ij} \in \{0, 1\}, \quad \forall i \in I, j \in J$$

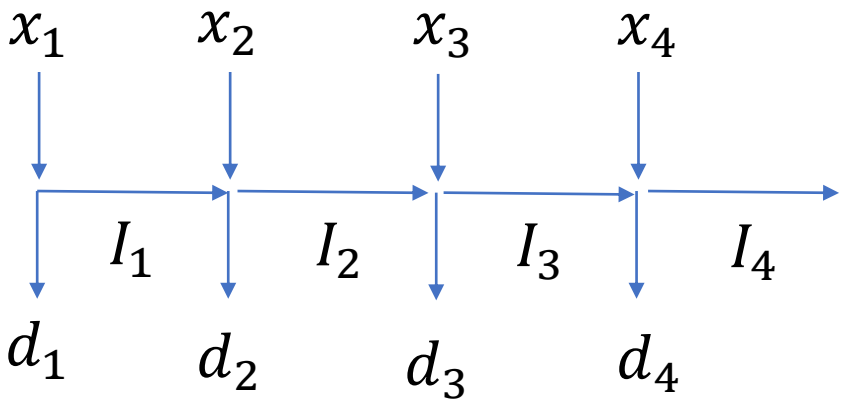
5.2 정수계획법 예제

❖ 로트크기 결정문제 (Lot Sizing Problem)

- 고정비용 문제를 생산계획 문제에 확장 적용하면 생산계획 시 중요한 문제로 부각되고 있는 로트크기 결정문제를 해결할 수 있음
- 로트크기 결정문제는 대상 생산품에 대해 계획 주기 내 단위기간별 생산량을 결정하는 문제
- 단위기간 $t \in \{1, \cdots, n\}$에서의 생산고정비용 f_t, 생산에 따른 단위생산비용 c_t, 재고유지비용 h_t, 수요량 d_t 이 주어져 있다고 할 때, 전체 비용을 최소화하는 매 기간별 생산량 x_t 와 기간말 재고량 I_t 를 구하는 것이 로트크기 결정문제의 목표
- 고정비용 항목 모형화를 위하여 추가 변수로서 z_t 를 도입하면, 기간 t에 생산이 이루어질 때 $z_t = 1$ 이 되고, 생산이 이루어지지 않으면 $z_t = 0$ 이 되는 이진 정수 변수가 됨

5.2 정수계획법 예제

❖ 로트크기 결정문제 (Lot Sizing Problem)



$$I_{t-1} + x_t = d_t + I_t$$

$$\begin{aligned} Min \quad & \textstyle\sum_{t\in T} f_t\, z_t + \sum_{t\in T} c_t\, x_t + \sum_{t\in T} h_t\, I_t \\ s.t. \quad & \\ & x_t \le M z_t \\ & I_{t-1} + x_t = d_t + I_t \\ & z_t,\ x_t \in \{0,1\},\ \forall t \in T \end{aligned}$$

5.3 정수계획법의 해법들

❖ 열거법(Full Enumeration Method)

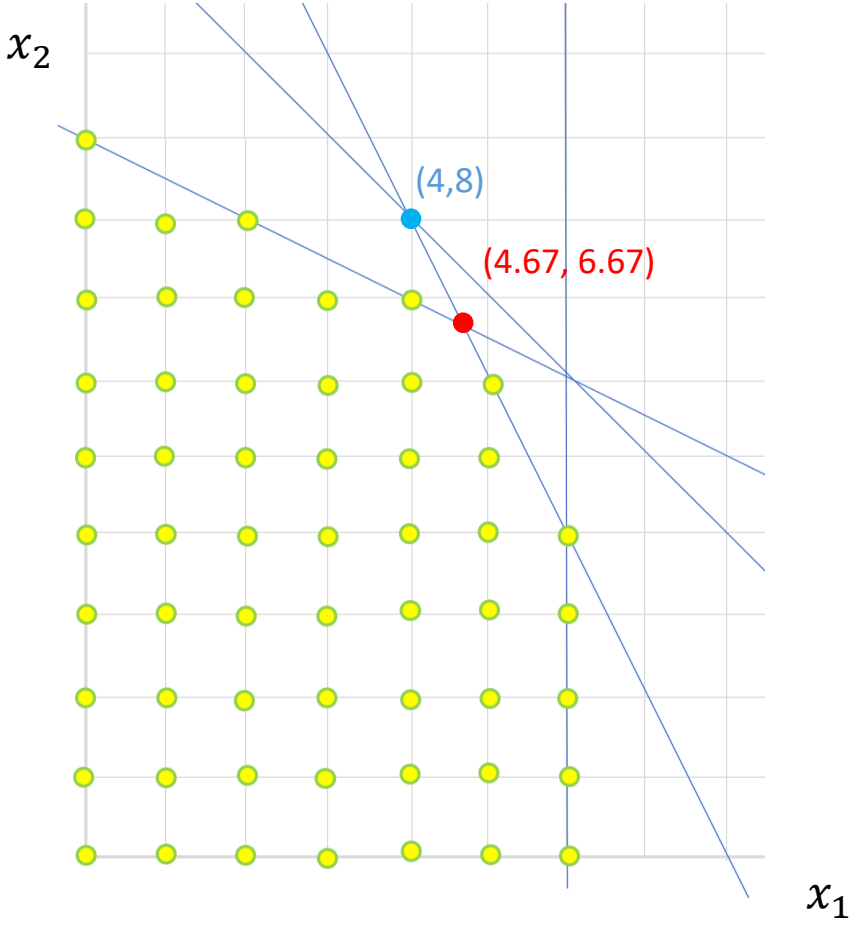
- 실행가능해(Feasible Solution)의 각 의사결정변수를 특정 정수에 고정하여 구할 수 있음
- 최적해(Optimal Solution)는 전체 정수해 조합 중 목적함수를 최대화 혹은 최소화하는 해를 선택하여 도출
- 즉, 모든 가능한 정수해 조합을 하나씩 비교하여 해를 도출하는 기법
- 그러나, 의사결정변수가 증가할수록 가능한 정수해 조합은 기하급수적으로 증가하게 되므로 문제의 규모가 작은 경우를 제외하고 열거법으로 최적해를 도출하는 것은 매우 비효율적

5.3 정수계획법의 해법들

❖ 열거법(Full Enumeration Method)

x_1 = 침대생산량
x_2 = 소파생산량

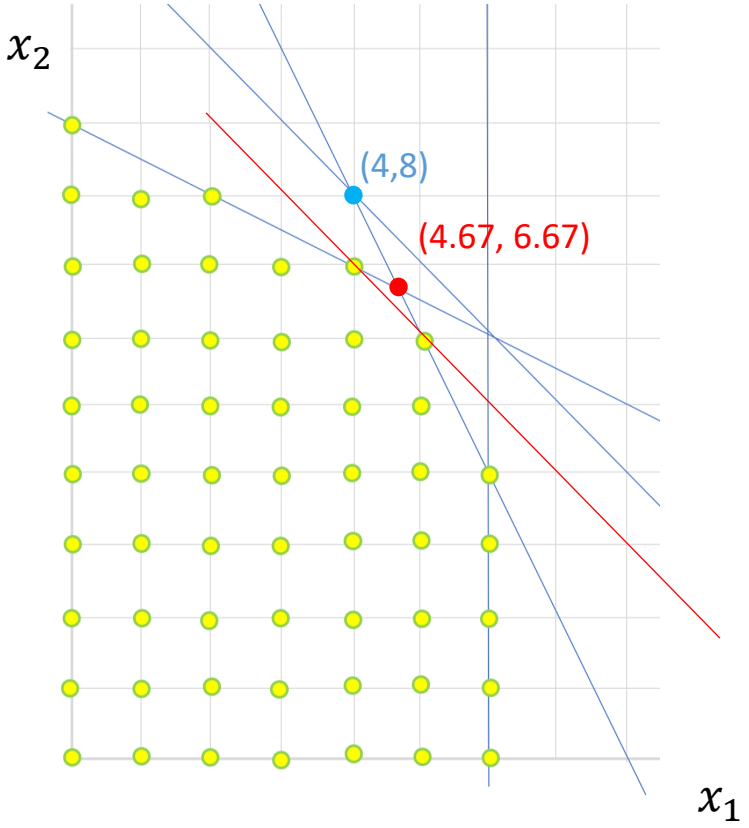
$$Max \ 150{,}000x_1 + 100{,}000x_2$$
$$s.t.$$
$$2x_1 + x_2 \leq 16$$
$$x_1 + 2x_2 \leq 18$$
$$x_1 \leq 6$$
$$x_1, x_2 \geq 0$$



5.3 정수계획법의 해법들

❖ 절단평면법(Cutting Plane Method)

➢ 기본 선형계획모형에 제약식을 추가하여 전체 해영역의 정점이 정수해 조건을 만족하도록 만들어 가는 기법



문제점?

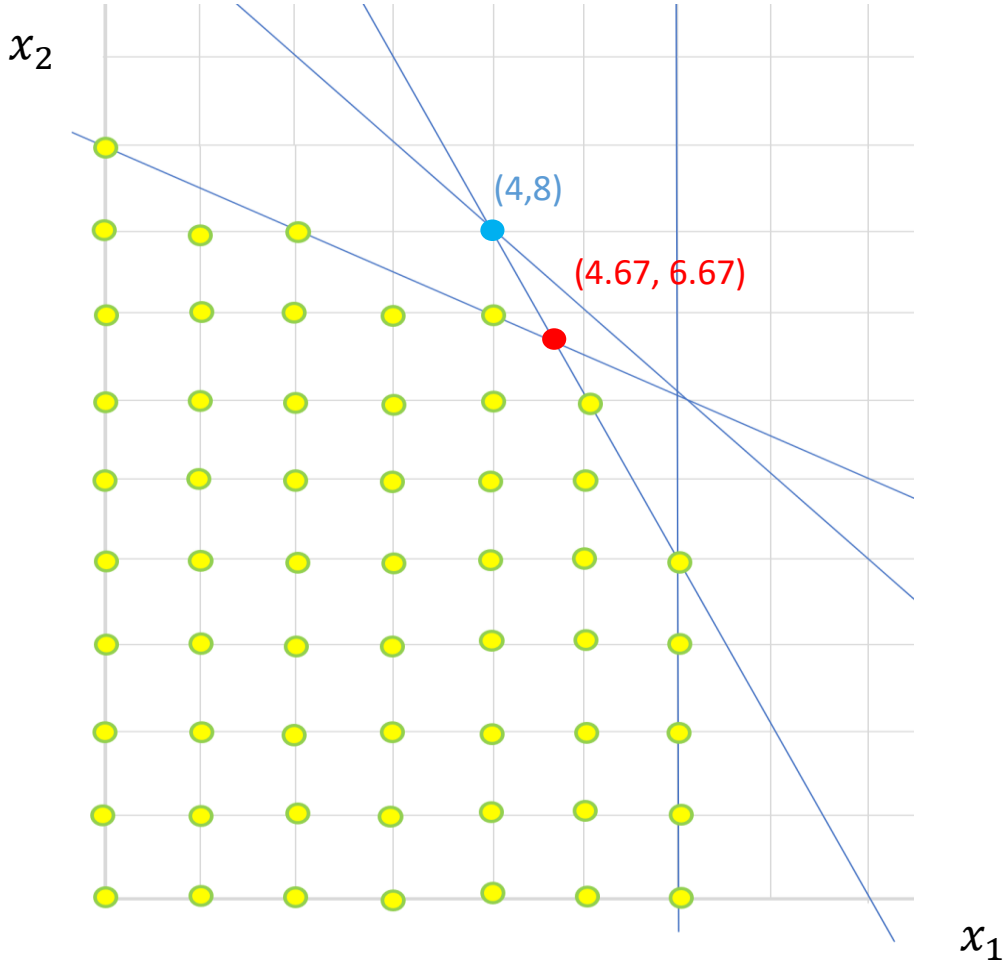
5.3 정수계획법의 해법들

❖ 분단탐색법(Branch-and-Bound Method)

- 해영역을 분리하여 가능성이 높은 영역을 탐색해 가는 방법
- 선형계획법에 의한 최적해가 정수해 조건을 만족시키지 못할 경우에 정수해 조건을 만족시키지 않는 의사결정변수 중 하나를 선택하여, 이 변수값을 기준으로 정수해가 포함되지 않는 해영역을 제거하고, 다시 선형계획법을 적용하는 반복 과정 수행

5.3 정수계획법의 해법들

❖ 분단탐색법(Branch-and-Bound Method)



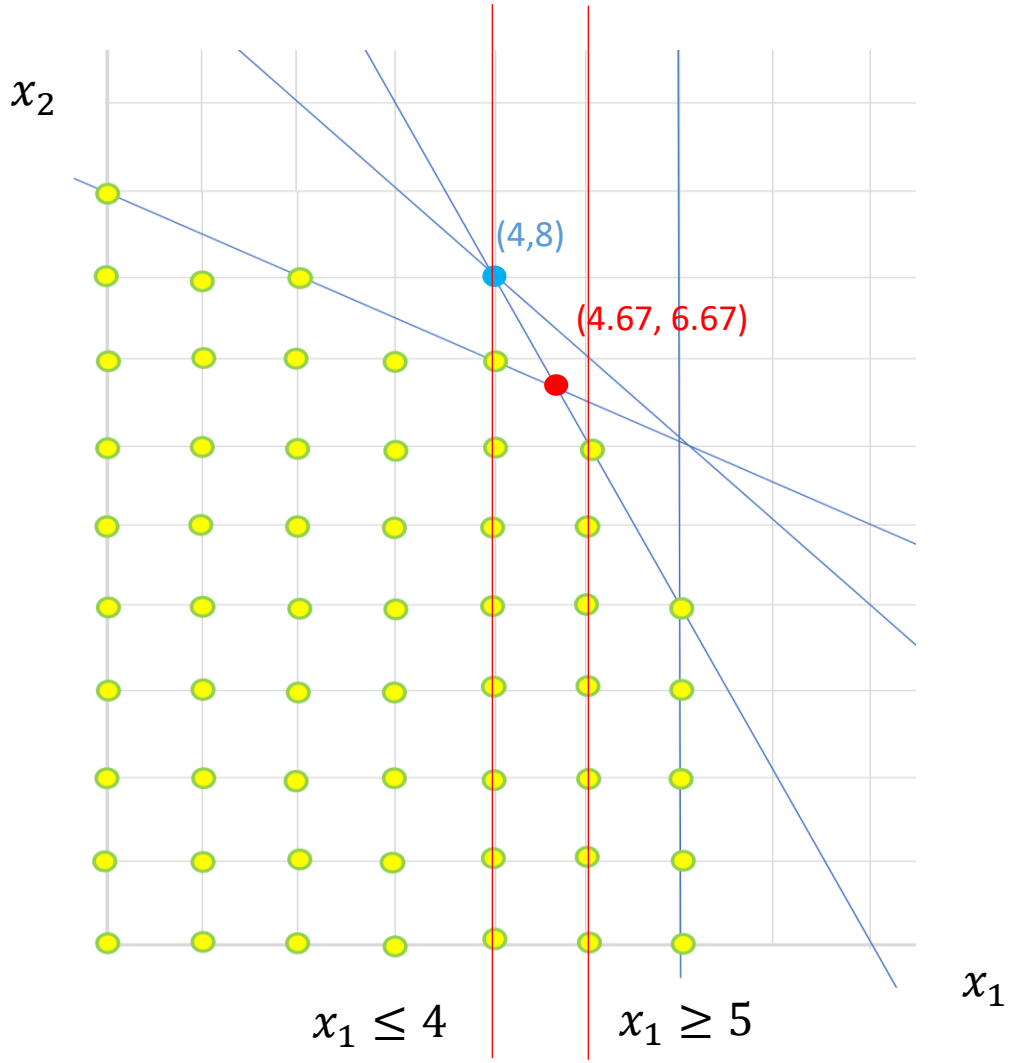
최적해

$(x_1, x_2) = (4.67, 6.67)$

$x_1 \leq 4 \quad or \quad x_1 \geq 5$

5.3 정수계획법의 해법들

❖ 분단탐색법(Branch-and-Bound Method)



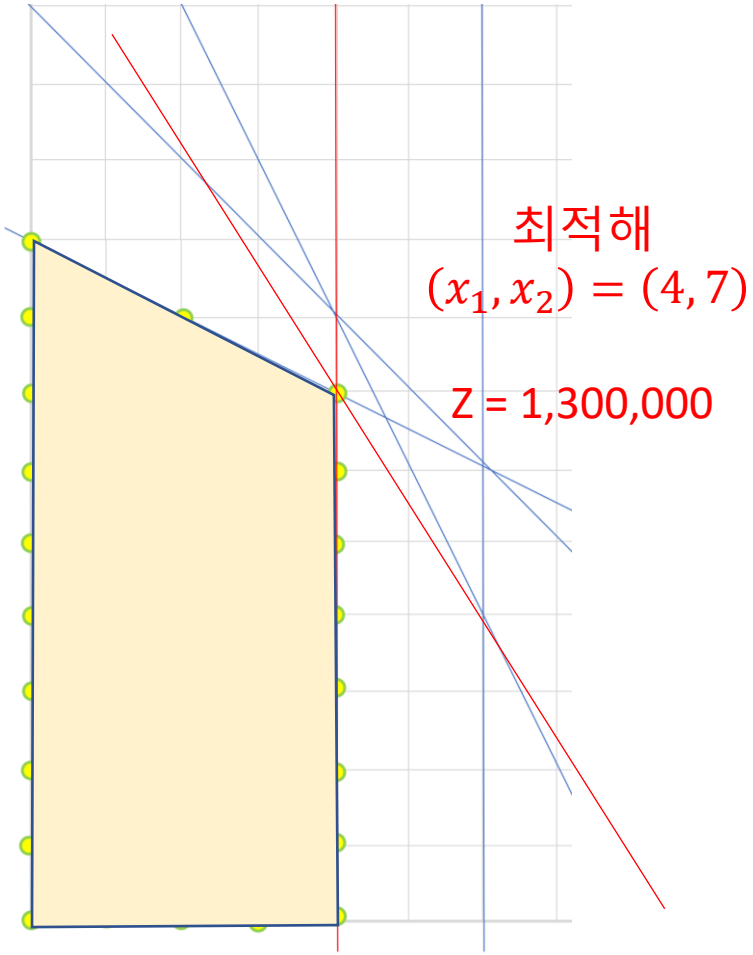
최적해
$(x_1, x_2) = (4.67, 6.67)$

$x_1 \leq 4 \quad or \quad x_1 \geq 5$

5.3 정수계획법의 해법들

❖ 분단탐색법(Branch-and-Bound Method)

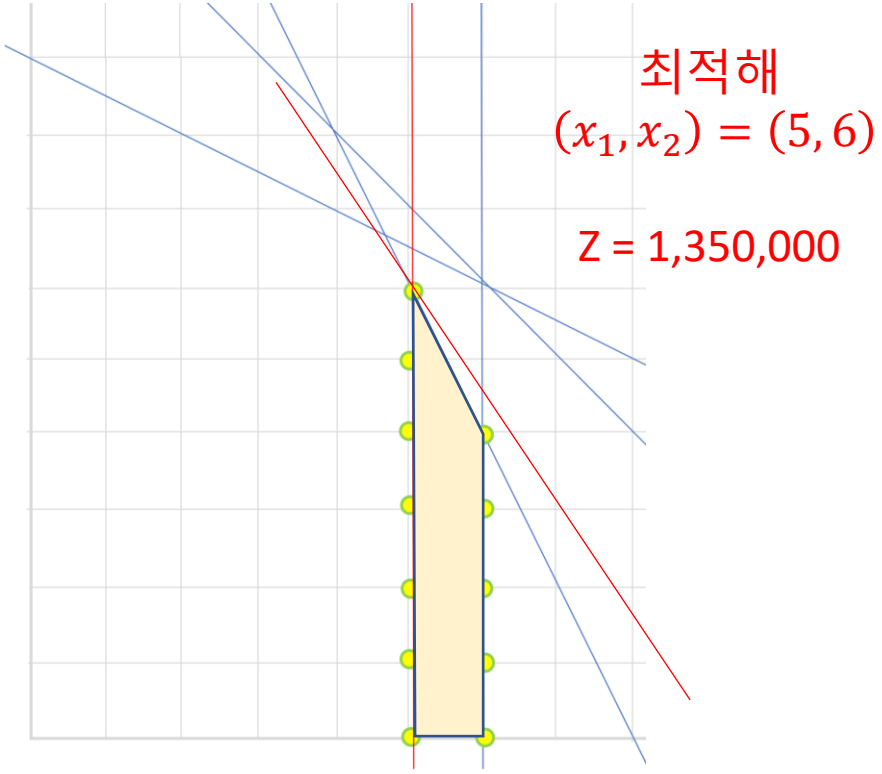
$$Max \ \ 150{,}000x_1 + 100{,}000x_2$$
$$s.t.$$
$$2x_1 + x_2 \leq 16$$
$$x_1 + 2x_2 \leq 18$$
$$x_1 \leq 6$$
$$x_1 \leq 4$$
$$x_1, x_2 \geq 0$$



5.3 정수계획법의 해법들

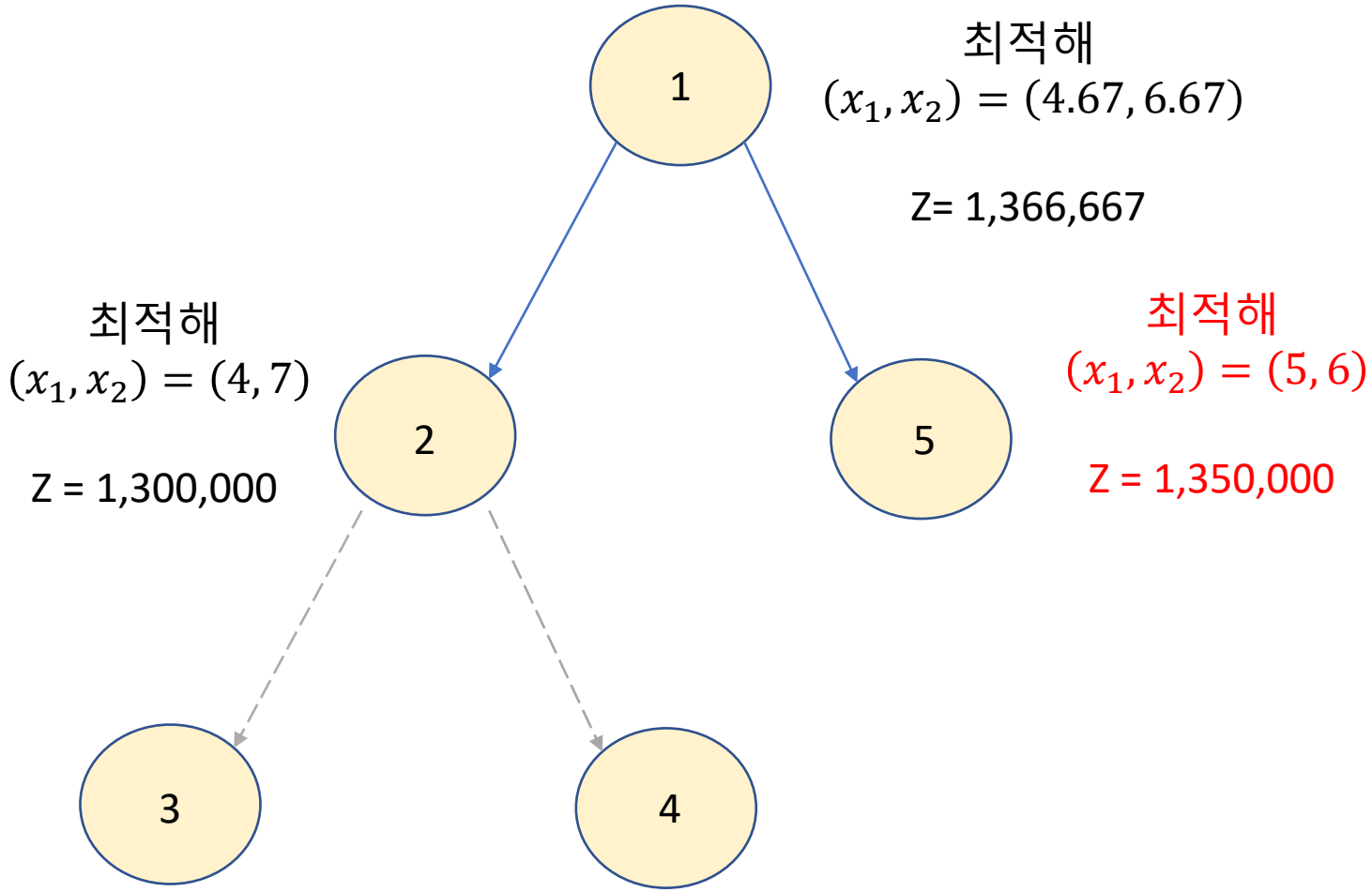
❖ 분단탐색법(Branch-and-Bound Method)

$$
\begin{aligned}
Max\ \ & 150{,}000x_1 + 100{,}000x_2 \\
s.t.\ \ & \\
2x_1 + x_2 &\le 16 \\
x_1 + 2x_2 &\le 18 \\
x_1 &\le 6 \\
x_1 &\ge 5 \\
x_1, x_2 &\ge 0
\end{aligned}
$$



5.3 정수계획법의 해법들

❖ 분단탐색법(Branch-and-Bound Method)



5.3 정수계획법의 해법들

❖ 발견적 해법(Heuristic Method)

- ➢ 선형계획완화법 (Linear Programming Relaxation)
- ➢ 메타 휴리스틱기법 (Meta Heuristic Method)
  - ✓ 시뮬레이티드 어닐링 기법 (Simulated Annealing)
  - ✓ 타부 탐색 기법 (Tabu Search)
  - ✓ 유전자 알고리즘 기법 (Genetic algorithm)

5.4 배낭 문제

❖ 배낭문제

- n개의 물건과 각 물건 i의 무게 w_i와 가치 v_i가 주어지고, 배낭의 용량이 C일 때, 배낭에 담을 수 있는 물건의 최대 가치는?
- 단, 배낭에 담은 물건의 무게의 합이 C를 초과하지 말아야 하고, 각 물건은 1개씩만 있다.
- 이러한 배낭 문제를 0-1 배낭 문제라고 한다.

5.4 배낭 문제

❖ 배낭문제 분류

➢ 물건을 쪼갤 수 있는 배낭문제(Fraction Knapsack Problem)

✓ 가치가 큰 물건부터 담고, 남은 무게 만큼 물건을 쪼개는 방식으로 그리디 알고리즘으로 해결

➢ 물건을 쪼갤 수 없는 배낭문제(0/1 Knapsack Problem)

✓ Brute-Force : 모든 경우의 수 고려(열거법, $O(2^n)$)

✓ Greedy : 가격이 높은 보석, 혹은 (가격/무게) 값이 높은 보석부터

✓ 동적계획법(DP, Dynamic Programming)을 활용해 해결

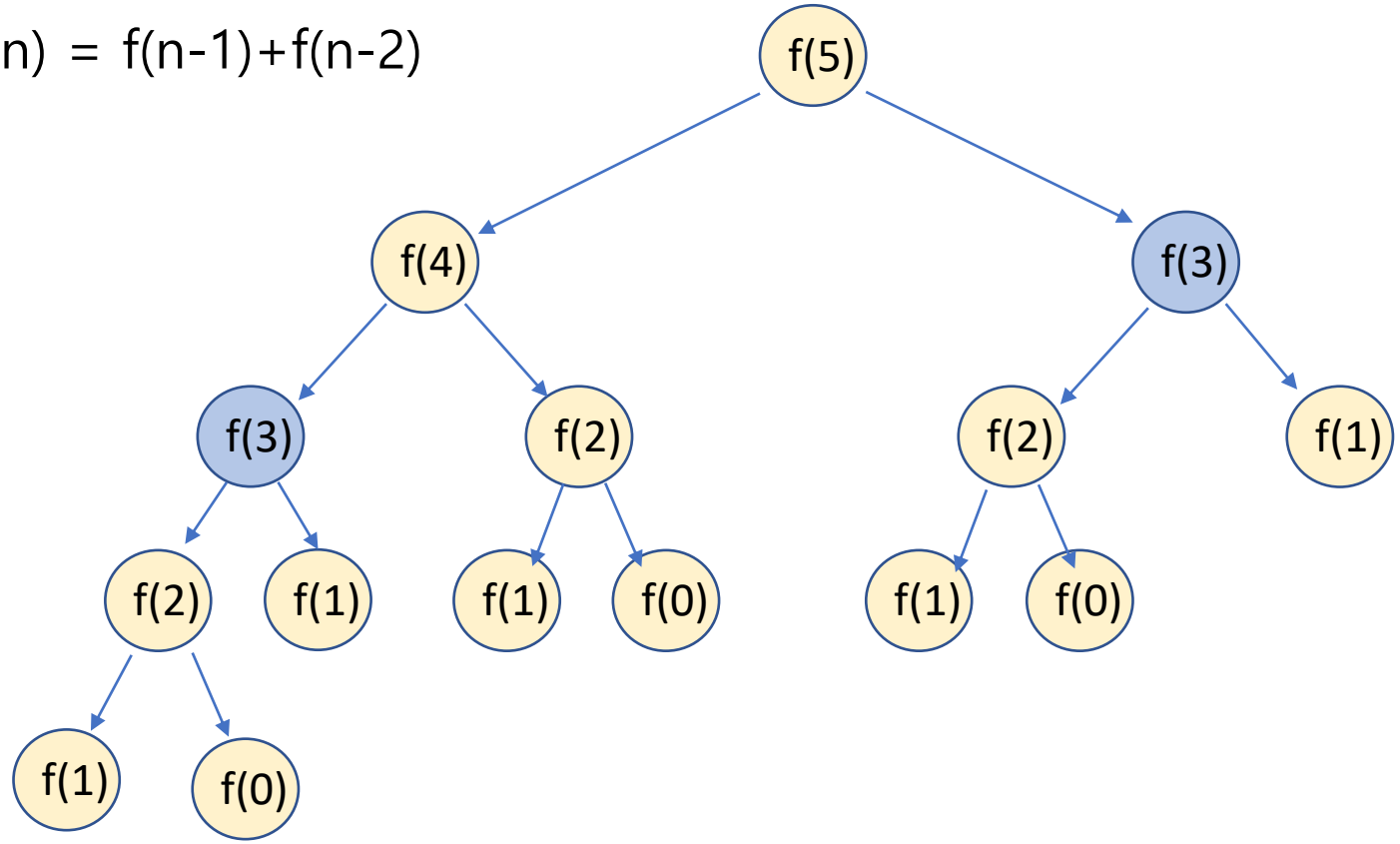
- Devide-and-Conquer

5.4 배낭 문제

❖ 배낭문제 분류

➢Devide-and-Conquer 예(피보나치 수열)

- 1, 1, 2, 3, 5, 8, 13, …
- f(n) = f(n-1)+f(n-2)



5.4 배낭 문제

❖ 부분 문제

- 문제에는 물건, 물건의 무게, 물건의 가치, 배낭의 용량, 모두 4가지의 요소가 있다.
- 물건과 물건의 무게는 부분 문제를 정의하는데 필요
  - ✓ K[i, w] = 물건 1~i까지만 고려하고, (임시) 배낭의 용량이 w일 때의 최대 가치
  - ✓ 단, i = 1, 2, ···, n이고, w = 1, 2, 3, ···, C
  - ✓ 문제의 최적해 = K[n, C]

5.4 배낭 문제

❖ Knapsack 알고리즘

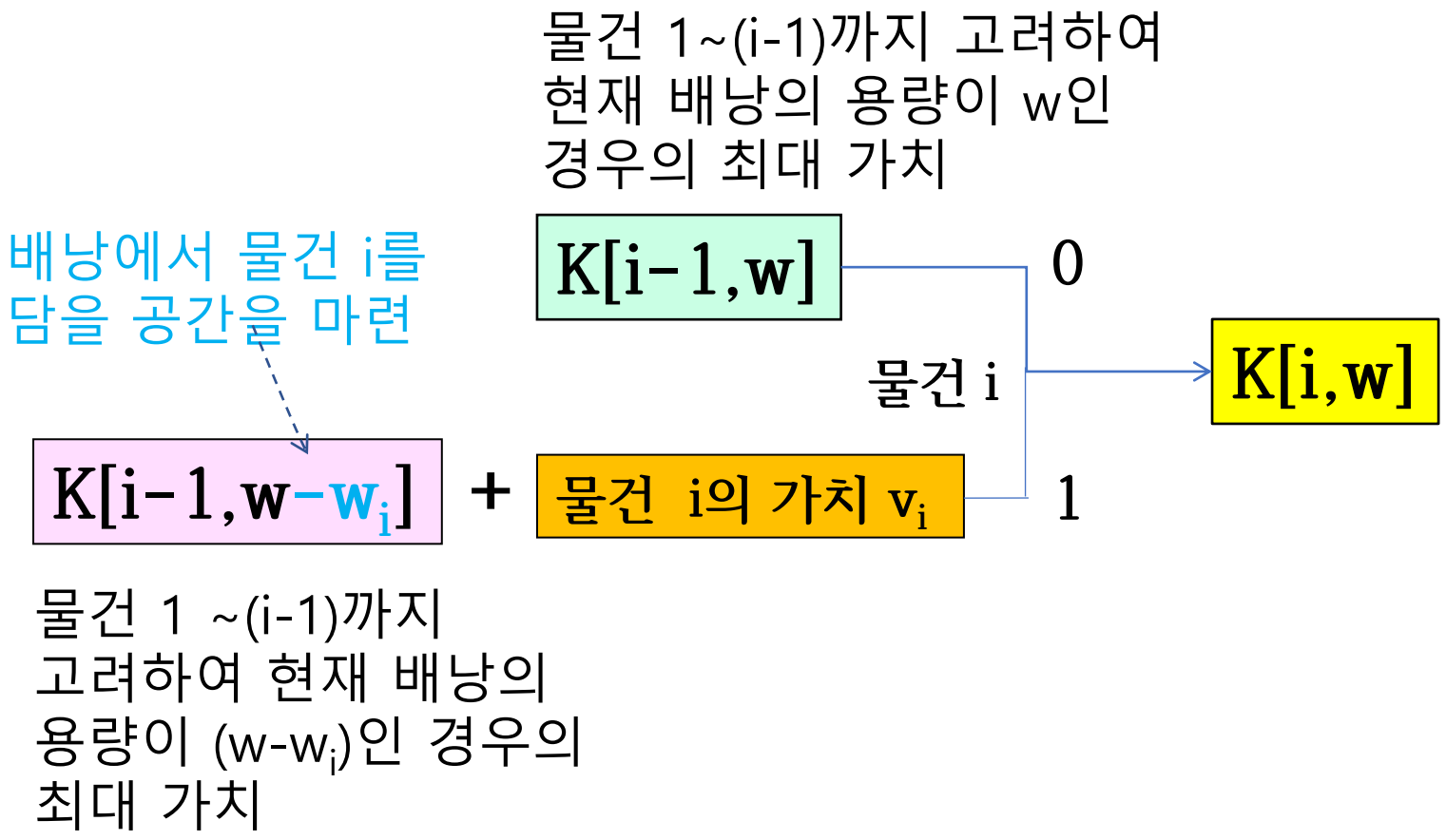
➢ 입력: 배낭의 용량 C, n개의 물건과 각 물건 i의 무게 wi와 가치 vi, 단, i = 1, 2, ···, n

➢ 출력: K[n, C]

```
1. for i = 0 to n    K[i,0]=0     // 배낭의 용량이 0일 때
2. for w = 0 to C   K[0,w]=0   // 물건 0이란 어떤 물건도 고려하지 않을 때
3. for i = 1 to n
4.     for w = 1 to C          // w는 배낭의 (임시) 용량
5.         if ( wi > w )       // 물건 i의 무게가 임시 배낭 용량을 초과하면
6.             K[i, w] = K[i-1, w]
7.         else                // 물건 i를 배낭에 담지 않을 경우와 담을 경우 고려
8.             K[i, w] = max{K[i-1, w], K[i-1, w-wi]+vi}
9. return K[n, C]
```

5.4 배낭 문제

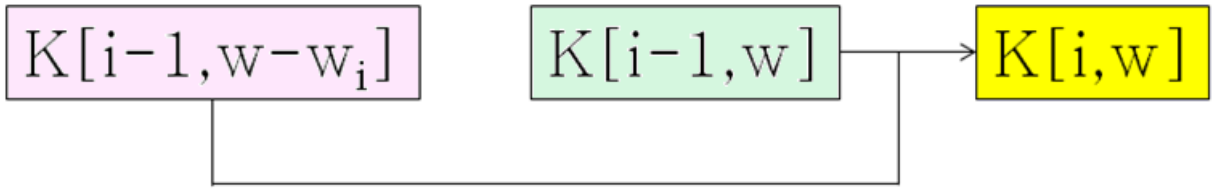
❖ Knapsack 알고리즘



5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 부분 문제 간의 함축적 순서



$$K[i,w] = \begin{cases} K[i-1,w] & if\ w_i > w \\ \max\{\, v_i + K[i-1,w-w_i], K[i-1,w]\} & else \end{cases}$$

5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 수행과정

배낭의 용량 C=10kg

물건	1	2	3	4
무게	5	3	6	4
가치	10	60	20	40

5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 수행과정

- ✓ 2차원 배열을 만들고 각 행과 열에 i번째 물건까지 넣을 수 있고, 배낭에 넣을 수 있는 최대무게가 j일 때, 최대 이윤을 저장해 나감
- ✓ 이 때, 행 i 는 물건 i까지 넣을 수 있을 때를 의미하고, 열 j는 가방의 최대 무게를 의미
- ✓ Line 1~2: 0번 행과 0번 열의 각 원소를 0으로 초기화

배낭용량		**w =**	0	1	2	3	4	5	6	7	8	9	10
무게	가치	물건	0	0	0	0	0	0	0	0	0	0	0
5	10	1	0										
3	60	2	0										
6	20	3	0										
4	40	4	0										

5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 수행과정

✓ w =1, 2, 3, 4일 때, 각각 물건 1을 담을 수 없다.

K[1, 1] = 0, K[1, 2] = 0, K[1, 3] = 0, K[1, 4] = 0

배낭용량		w =	0	1	2	3	4	5	6	7	8	9	10
무게	가치	물건	0	0	0	0	0	0	0	0	0	0	0
5	10	1	0	0	0	0	0						
3	60	2	0										
6	20	3	0										
4	40	4	0										

5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 수행과정

✓ w = 5 일 때,

K[1, 5] = 10

배낭용량		w =	0	1	2	3	4	5	6	7	8	9	10
무게	가치	물건	0	0	0	0	0	0	0	0	0	0	0
5	10	1	0	0	0	0	0	10					
3	60	2	0										
6	20	3	0										
4	40	4	0										

5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 수행과정

✓ w = 6, 7, 8, 9, 10일 때

K[1, 6] = K[1, 7] = K[1, 8] = K[1, 9] = K[1, 10] = 10

배낭용량		w =	0	1	2	3	4	5	6	7	8	9	10
무게	가치	물건	0	0	0	0	0	0	0	0	0	0	0
5	10	1	0	0	0	0	0	10	10	10	10	10	10
3	60	2	0										
6	20	3	0										
4	40	4	0										

5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 수행과정

✓ w = 1, 2 (배낭의 용량이 각각 1, 2 kg)일 때

K[2, 1] = 0, K[2 ,2] = 0

배낭용량		w =	0	1	2	3	4	5	6	7	8	9	10
무게	가치	물건	0	0	0	0	0	0	0	0	0	0	0
5	10	1	0	0	0	0	0	10	10	10	10	10	10
3	60	2	0	0	0								
6	20	3	0										
4	40	4	0										

5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 수행과정

✓ w = 3, 4 (배낭의 용량이 3kg)일 때

K[2,3] = 600

배낭용량		w =	0	1	2	3	4	5	6	7	8	9	10
무게	가치	물건	0	0	0	0	0	0	0	0	0	0	0
5	10	1	0	0	0	0	0	10	10	10	10	10	10
3	60	2	0	0	0	60	60						
6	20	3	0										
4	40	4	0										

5.4 배낭 문제

❖ Knapsack 알고리즘

물건 1을 배낭에서 빼낸 후, 물건 2를 담는다.
그 때의 가치가 60이다

➢ 수행과정

✓ w = 5 (배낭의 용량이 5kg)일 때

K[2,5] = max{K[i-1,w], K[i-1,w-wi]+vi}
= max{K[2-1,5], K[2-1,5-4]+60}
= max{K[1,5], K[1,2]+60}
= max{10, 0+60}
= max{10, 60} = 60

배낭용량		w =	0	1	2	3	4	5	6	7	8	9	10
무게	가치	물건	0	0	0	0	0	0	0	0	0	0	0
5	10	1	0	0	0	0	0	10	10	10	10	10	10
3	60	2	0	0	0	60	60	60					
6	20	3	0										
4	40	4	0										

5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 수행과정

✓ w = 6, 7일 때

✓ 각각의 경우도 물건 1을 빼내고 물건 2를 배낭에 담는 것이 더 큰 가치

✓ K[2,6] = K[2,7] = 60

max{K[1,6], K[1,3]+60}

max{K[1,7], K[1,4]+60}

배낭용량		w =	0	1	2	3	4	5	6	7	8	9	10
무게	가치	물건	0	0	0	0	0	0	0	0	0	0	0
5	10	1	0	0	0	0	0	10	10	10	10	10	10
3	60	2	0	0	0	60	60	60	60	60			
6	20	3	0										
4	40	4	0										

5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 수행과정

물건 1, 2 둘 다를 담을 수 있다.

✓ w = 8 (배낭의 용량이 8kg)일 때

$$K[2,8] = \max\{K[i-1,w], K[i-1,w-wi]+vi\}$$
$$= \max\{K[2-1,8], K[2-1,8-3]+60\}$$
$$= \max\{K[1,8], K[1,5]+60\}$$
$$= \max\{10, 10+60\}$$
$$= \max\{10, 70\} = 70$$

배낭용량		w =	0	1	2	3	4	5	6	7	8	9	10
무게	가치	물건	0	0	0	0	0	0	0	0	0	0	0
5	10	1	0	0	0	0	0	10	10	10	10	10	10
3	60	2	0	0	0	60	60	60	60	60	70		
6	20	3	0										
4	40	4	0										

5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 수행과정

✓ w = 9, 10 (배낭의 용량이 9, 10kg)일 때

✓ 물건 1, 2를 배낭에 둘 다 담을 수 있다

max{K[1,9], K[1,6]+60}

배낭용량		w =	0	1	2	3	4	5	6	7	8	9	10
무게	가치	물건	0	0	0	0	0	0	0	0	0	0	0
5	10	1	0	0	0	0	0	10	10	10	10	10	10
3	60	2	0	0	0	60	60	60	60	60	70	70	70
6	20	3	0										
4	40	4	0										

5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 수행과정

✓ w = 1, 2, 3 일 때

K[3, 1]=0, K[3 ,2] = 0, K[3, 3] = 0

배낭용량		w =	0	1	2	3	4	5	6	7	8	9	10
무게	가치	물건	0	0	0	0	0	0	0	0	0	0	0
5	10	1	0	0	0	0	0	10	10	10	10	10	10
3	60	2	0	0	0	60	60	60	60	60	70	70	70
6	20	3	0	0	0	60							
4	40	4	0										

5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 수행과정

✓ w = 4, 5, 6, 7 일 때

max{K[i-1,w], K[i-1,w-wi]+vi}

K[2, 4], K[2 ,5], K[2, 6]], K[2, 7] = 60

배낭용량		w =	0	1	2	3	4	5	6	7	8	9	10
무게	가치	물건	0	0	0	0	0	0	0	0	0	0	0
5	10	1	0	0	0	0	0	10	10	10	10	10	10
3	60	2	0	0	0	60	60	60	60	60	70	70	70
6	20	3	0	0	0	60	60	60	60	60			
4	40	4	0										

5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 수행과정

✓ w = 8, 9 일 때

$$\max\{K[i-1,w],\ K[i-1,w-wi]+vi\}$$

$$K[2,\ 8],\ K[2\ ,2]+20$$

배낭용량		w =	0	1	2	3	4	5	6	7	8	9	10
무게	가치	물건	0	0	0	0	0	0	0	0	0	0	0
5	10	1	0	0	0	0	0	10	10	10	10	10	10
3	60	2	0	0	0	60	60	60	60	60	70	70	70
6	20	3	0	0	0	60	60	60	60	60	70	80	
4	40	4	0										

5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 수행과정

✓ w = 10 일 때

$$\max\{K[i-1,w],\ K[i-1,w-wi]+vi\}$$

K[2, 10], K[2,4]+20

배낭용량		w =	0	1	2	3	4	5	6	7	8	9	10
무게	가치	물건	0	0	0	0	0	0	0	0	0	0	0
5	10	1	0	0	0	0	0	10	10	10	10	10	10
3	60	2	0	0	0	60	60	60	60	60	70	70	70
6	20	3	0	0	0	60	60	60	60	60	70	80	80
4	40	4	0										

5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 수행과정

✓ w = 1, 2, 3 일 때

max{K[i-1,w], K[i-1,w-wi]+vi}

배낭용량		w =	0	1	2	3	4	5	6	7	8	9	10
무게	가치	물건	0	0	0	0	0	0	0	0	0	0	0
5	10	1	0	0	0	0	0	10	10	10	10	10	10
3	60	2	0	0	0	60	60	60	60	60	70	70	70
6	20	3	0	0	0	60	60	60	60	60	70	80	80
4	40	4	0	0	0	60							

5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 수행과정

✓ w = 4, 5, 6, 7 일 때

max{K[i-1,w], K[i-1,w-wi]+vi}

배낭용량		w =	0	1	2	3	4	5	6	7	8	9	10
무게	가치	물건	0	0	0	0	0	0	0	0	0	0	0
5	10	1	0	0	0	0	0	10	10	10	10	10	10
3	60	2	0	0	0	60	60	60	60	60	70	70	70
6	20	3	0	0	0	60	60	60	60	60	70	80	80
4	40	4	0	0	0	60	60	60	60	100			

5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 수행과정

✓ w = 8, 9, 10 일 때

max{K[i-1,w], K[i-1,w-wi]+vi}

K[3, 8], K[3,4]+40

K[3, 9], K[3,5]+40

K[3, 10], K[3,6]+40

배낭용량		w =	0	1	2	3	4	5	6	7	8	9	10
무게	가치	물건	0	0	0	0	0	0	0	0	0	0	0
5	10	1	0	0	0	0	0	10	10	10	10	10	10
3	60	2	0	0	0	60	60	60	60	60	70	70	70
6	20	3	0	0	0	60	60	60	60	60	70	80	80
4	40	4	0	0	0	60	60	60	60	100	100	100	100

5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 수행과정 결과

배낭용량		w =	0	1	2	3	4	5	6	7	8	9	10
무게	가치	물건	0	0	0	0	0	0	0	0	0	0	0
5	10	1	0	0	0	0	0	10	10	10	10	10	10
3	60	2	0	0	0	60	60	60	60	60	70	70	70
6	20	3	0	0	0	60	60	60	60	60	70	80	80
4	40	4	0	0	0	60	60	60	60	100	100	100	100

3kg
60만원

4kg
40만원

10kg

5.4 배낭 문제

❖ Knapsack 알고리즘

➢ 시간 복잡도

✓ 1개의 부분 문제에 대한 해를 구할 때의 시간 복잡도

- line 5에서의 무게를 1번 비교한 후 line 6에서는 1개의 부분 문제의 해를 참조하고, line 8에서는 2개의 해를 참조한 계산이므로 O(1) 시간

✓ 부분 문제의 수는 배열 K의 원소 수인 n x C

- C = 배낭의 용량

✓ Knapsack 알고리즘의 시간 복잡도

- O(1) x n x C = O(nC)

교육부 Ministry of Education

NRF 한국연구재단 National Research Foundation of Korea

thank you

본 과제(결과물)는 교육부와 한국연구재단의 재원으로 지원을 받아 수행된
디지털신기술인재양성 혁신공유대학사업의 연구결과입니다.

Convergence and Open Sharing System COSS 인공지능

전남대학교 CHONNAM NATIONAL UNIVERSITY

성균관대학교 SUNGKYUNKWAN UNIVERSITY

KNU 경북대학교 KYUNGPOOK NATIONAL UNIVERSITY

전주대학교 JEONJU UNIVERSITY

서울시립대학교 UNIVERSITY OF SEOUL

서울과학기술대학교 SEOUL NATIONAL UNIVERSITY OF SCIENCE & TECHNOLOGY

영진전문대학교 YEUNGJIN UNIVERSITY