[2008][14-2]

Computer aided ship design

Part 3. Optimization Methods

December 2008

Prof. Kyu-Yeul Lee

Department of Naval Architecture and Ocean Engineering,
Seoul National University of College of Engineering

Advanced
Ship
Design
Automation
Laboratory

Integer Programming(정수 계획법)
- Introduction

Advanced
Ship
Design
Automation
Laboratory

정수 계획법(Integer Programming)

- 정수 계획법: 선형계획법의 해 중에서 정수해만을 인정하는 경우, 최적해가 정수가 된다는 것을 보장할 수 있는 방법을 선형정수계획법(정수계획법)이라 함

- 예를 들어 항공회사에서 여객기 구매계획을 위한 모형의 최적해를 찾는 경우
  - 선형계획법을 이용하여 구한 해가 "B747을 13/4대, Air Bus 400을 22/3대 구입" 이라면 이는 실행이 불가능한 답이다
  - 정수계획법을 이용하여 "B747을 3대, Air Bus 400을 7대 구입"이라는 실행 가능한 정수 해를 구할 수 있다.

정수 계획법의 분류

"Stanley Zionts, Linear and integer programming, Prentice-Hall, 1974, p.336"
"Gerard Sierksma, Linear and integer programming, Marcel Dekker, Inc., 1996"
"박순달, 경영과학, 4정판, 민영사, 2003."
"박재흥, 최신경영과학, 시그마그래프, 2004."
"Laurence A. Wolsey, Integer programming, A Wiley-Interscience Publication, 1998"

수학적(Mathematical) 최적화 방법

Cut Algorithm(절단 평면법)

주어진 선형 계획 문제에 정수해는 만족하고, 정수가 아닌 해는 만족하지 않는 제약조건을 추가하면서 문제를 푸는 방법

Enumeration Algorithm(열거법)

주어진 선형 계획 문제의 가능해 영역(feasible region)내의 정수해를 열거하여 최적해를 찾는 방법

예)분지 한계법(분단 탐색법) - Branch and Bound Method

열거법의 일종으로서, 상한(upper bound)와 하한(lower bound)라는 개념을 사용하여 가능한한 가능해를 적게 열거하여 최적해를 찾는 방법

Constructive Algorithm

선형 계획 문제를 특성에 따라 분류하고 그 특성에 맞는 최적 해법을 적용하는 것

문제의 분류	네트워크 이론(Network theory)			할당 문제	●●●
	최단 경로 문제	최소 비용 문제	최대 흐름 문제		
해법	다익스트라(Dijkstra) 해법	그리디(Greedy) 해법	최대 흐름 문제 해법	헝가리법(Hungarian method)	●●●

휴리스틱(Heuristic) 최적화 방법

비록 최적해는 구하지 못하지만 최적해에 가까운 좋은 해를 짧은 시간 내에(즉, 적은 계산량으로) 찾으려는 방법

- 개미 알고리즘
- 유전 알고리즘
- 최근거리 인접점(nearest-neighbor) 알고리즘
- 교점교환 알고리즘
- k-optimal 알고리즘
- ●●●

Integer Programming(정수 계획법)

- Cut Algorithm(절단 평면법)

Advanced
Ship
Design
Automation
Laboratory

정수 계획법의 분류

"Stanley Zionts, Linear and integer programming, Prentice-Hall, 1974, p.336"
"Gerard Sierksma, Linear and integer programming, Marcel Dekker, Inc., 1996"
"박순달, 경영과학, 4정판, 민영사, 2003."
"박재흥, 최신경영과학, 시그마그래프, 2004."
"Laurence A. Wolsey, Integer programming, A Wiley-Interscience Publication, 1998"

수학적(Mathematical) 최적화 방법

Cut Algorithm(절단 평면법)

주어진 선형 계획 문제에 정수해는 만족하고, 정수가 아닌 해는 만족하지 않는 제약조건을 추가하면서 문제를 푸는 방법

Enumeration Algorithm(열거법)

주어진 선형 계획 문제의 가능해 영역(feasible region)내의 정수해를 열거하여 최적해를 찾는 방법

예)분지 한계법(분단 탐색법) - Branch and Bound Method

열거법의 일종으로서, 상한(upper bound)와 하한(lower bound)라는 개념을 사용하여 가능한한 가능해를 적게 열거하여 최적해를 찾는 방법

Constructive Algorithm

선형 계획 문제를 특성에 따라 분류하고 그 특성에 맞는 최적 해법을 적용하는 것

문제의 분류	네트워크 이론(Network theory)			할당 문제	●●●
	최단 경로 문제	최소 비용 문제	최대 흐름 문제		
해법	다익스트라(Dijkstra) 해법	그리디(Greedy) 해법	최대 흐름 문제 해법	헝가리법(Hungarian method)	●●●

휴리스틱(Heuristic) 최적화 방법

비록 최적해는 구하지 못하지만 최적해에 가까운 좋은 해를 짧은 시간 내에(즉, 적은 계산량으로) 찾으려는 방법

- 개미 알고리즘
- 유전 알고리즘
- 최근거리 인접점(nearest-neighbor) 알고리즘
- 교점교환 알고리즘
- k-optimal 알고리즘
- ●●●

정수 계획법의 해법
- 예) 절단 평면법

"박순달, 경영과학, 4정판, 민영사, 2003."

"Stanley Zionts, Linear and integer programming, Prentice-Hall, 1974, p.342 ~ p.387"

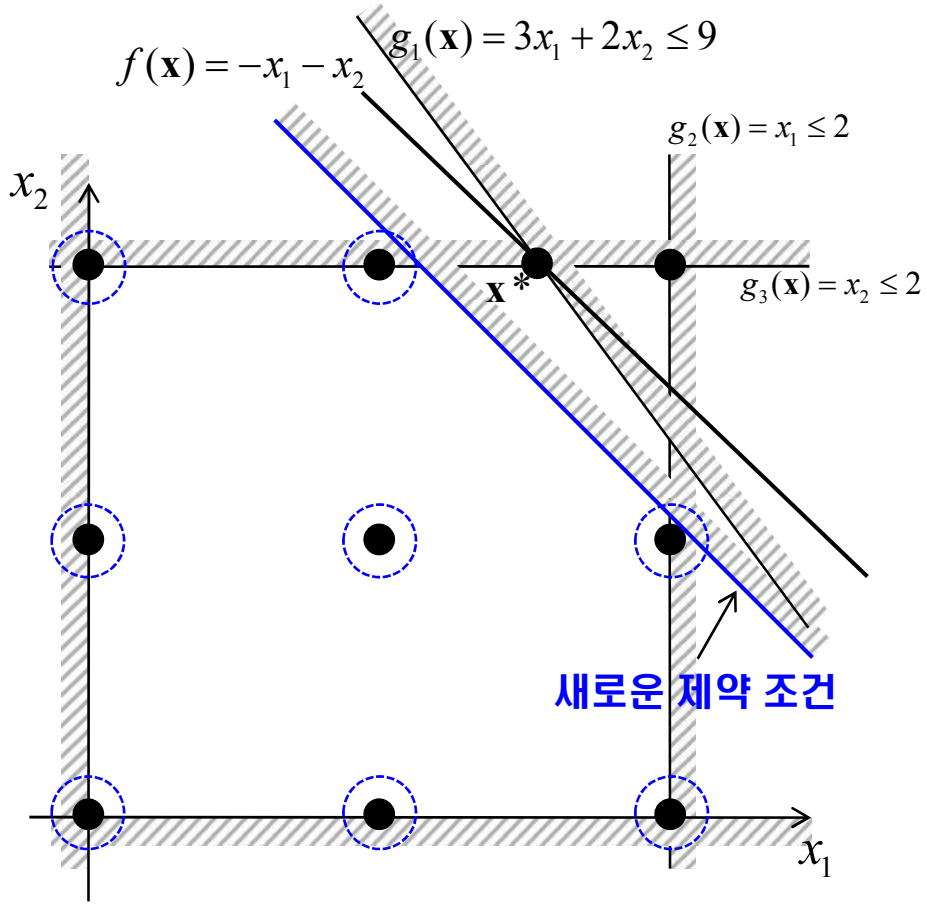
$$\textit{Minimize} \quad f(\mathbf{x}) = -x_1 - x_2$$

$$\textit{Subject to} \quad g_1(\mathbf{x}) = 3x_1 + 2x_2 \le 9$$

$$g_2(\mathbf{x}) = x_1 \le 2$$

$$g_3(\mathbf{x}) = x_2 \le 2$$

$$x_j \ge 0, \ \forall j$$



✓ 위 문제의 최적해는 정수가 아니다.
$\mathbf{x}^* = (5/3 , 2)$

✓ 최적해($\mathbf{x}^*$)와 모든 정수 가능해 사이를 지나는 새로운 제약조건을 구할 수 있다면?

✓ 새로운 제약조건을 절단 평면이라고 한다.

절단 평면법

1. 선형 계획 문제를 풀어 최적해를 구한다.
2. 구한 최적해가 정수가 아니면 절단 평면을 추가하여 문제를 푼다.
3. 정수해가 나올 때까지 과정 1, 2를 반복한다.

○ : 정수 가능해

$\mathbf{x}^*$: 최적 해

정수 계획법의 해법
- 예) 절단 평면법

1. 선형 계획 문제를 푼다.
2. 정수해가 아니면 절단 평면(제약조건)을 추가 한다.

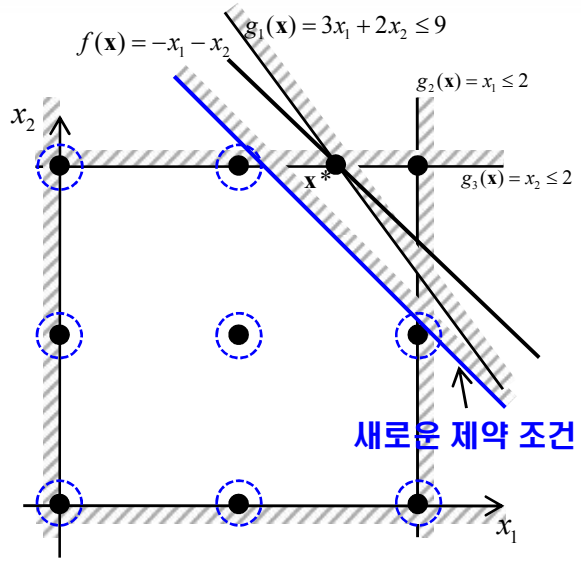
Minimize $f(\mathbf{x}) = -x_1 - x_2$

Subject to $g_1(\mathbf{x}) = 3x_1 + 2x_2 \le 9$

$g_2(\mathbf{x}) = x_1 \le 2$

$g_3(\mathbf{x}) = x_2 \le 2$

$x_j \ge 0,\ \forall j$



: 정수 가능해

$\mathbf{x}^*$: 최적 해

부등호 제약 조건을 등호 제약 조건으로 변환

Minimize $f(\mathbf{x}) = -x_1 - x_2$

Subject to

$$3x_1 + 2x_2 + x_3 = 9$$
$$x_1 + x_4 = 2$$
$$x_2 + x_5 = 2$$
$$x_j \ge 0,\ \forall j$$

	x1	x2	x3	x4	x5	bi	bi/ai
x3	3	2	1	0	0	9	3
x4	1	0	0	1	0	2	2
x5	0	1	0	0	1	2	
Obj.	-1	-1	0	0	0	f+0	-

정수 계획법의 해법
- 예) 절단 평면법

1. 선형 계획 문제를 푼다.
2. 정수해가 아니면 절단 평면(제약조건)을 추가 한다.

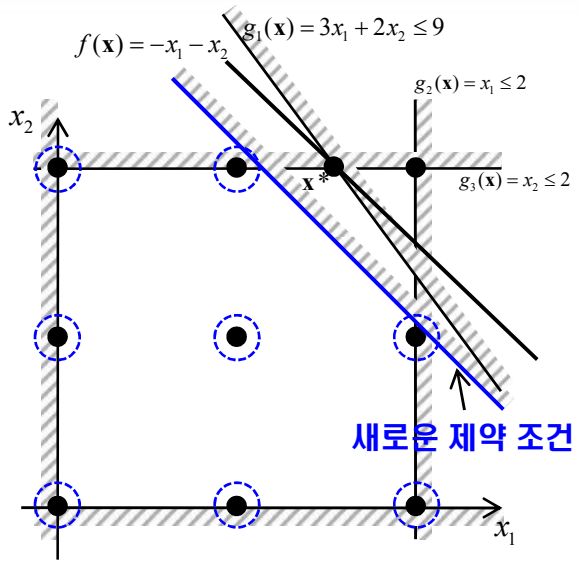
	x1	x2	x3	x4	x5	bi	bi/ai
x3	3	2	1	0	0	9	3
x4	1	0	0	1	0	2	2
x5	0	1	0	0	1	2	
Obj.	-1	-1	0	0	0	f+0	-

	x1	x2	x3	x4	x5	bi	bi/ai
x3	0	2	1	-3	0	3	3/2
x1	1	0	0	1	0	2	
x5	0	1	0	0	1	2	2
Obj.	0	-1	0	0	1	f+2	-

	x1	x2	x3	x4	x5	bi	bi/ai
x2	0	1	1/2	-3/2	0	3/2	
x1	1	0	0	1	0	2	2
x5	0	0	-1/2	3/2	1	1/2	1/3
Obj.	0	0	1/2	-1/2	0	f+7/2	-

$$\textit{Minimize} \quad f(\mathbf{x}) = -x_1 - x_2$$

$$\textit{Subject to} \quad g_1(\mathbf{x}) = 3x_1 + 2x_2 \le 9$$
$$g_2(\mathbf{x}) = x_1 \le 2$$
$$g_3(\mathbf{x}) = x_2 \le 2$$
$$x_j \ge 0, \ \forall j$$



: 정수 가능해

$\mathbf{x}^*$: 최적 해

	x1	x2	x3	x4	x5	bi	bi/ai
x2	0	1	0	0	1	2	
x1	1	0	1/3	0	-2/3	5/3	
x4	0	0	-1/3	1	2/3	1/3	
Obj.	0	0	1/3	0	1/3	f+11/3	-

위 테이블로부터 최적점은 다음과 같다.

$$x_1 = \frac{5}{3}, x_2 = 2$$

정수해가 아니므로 절단 평면을 추가해야 함

정수 계획법의 해법
- 예) 절단 평면법

$$
\begin{aligned}
f(\mathbf{x}) &: z && + \bar{c}_{m+1}x_{m+1} + \cdots + \bar{c}_n x_n = z_0 \\
g_1(\mathbf{x}) &: x_1 && + \bar{a}_{1,m+1}x_{m+1} + \cdots + \bar{a}_{1,n}x_n = \bar{b}_1 \\
g_2(\mathbf{x}) &: x_2 && + \bar{a}_{2,m+1}x_{m+1} + \cdots + \bar{a}_{2,n}x_n = \bar{b}_2 \\
&\vdots \\
g_m(\mathbf{x}) &: && x_m + \bar{a}_{2,m+1}x_{m+1} + \cdots + \bar{a}_{2,n}x_n = \bar{b}_m
\end{aligned}
$$

1. 선형 계획 문제를 푼다.
2. 정수해가 아니면 절단 평면(제약조건)을 추가 한다.

	x1	x2	x3	x4	x5	bi	bi/ai
x2	0	1	0	0	1	2	
x1	1	0	1/3	0	-2/3	5/3	
x4	0	0	-1/3	1	2/3	1/3	
Obj.	0	0	1/3	0	1/3	f+11/3	-

1. 위의 Simplex 테이블은 다음과 같이 나타낼 수 있다.

$$
\begin{aligned}
g_1(\mathbf{x}) &: \quad x_2 \qquad\qquad + x_5 = 2 \\
g_2(\mathbf{x}) &: x_1 + \frac{1}{3}x_3 - \frac{2}{3}x_5 = \frac{5}{3} \\
g_3(\mathbf{x}) &: \quad -\frac{1}{3}x_3 + x_4 + \frac{2}{3}x_5 = \frac{1}{3} \\
f(\mathbf{x}) &: \quad \frac{1}{3}x_3 + \frac{1}{3}x_5 = f + \frac{11}{3}
\end{aligned}
$$

2. 식 g2를 이용하여 절단 평면을 만든다.
(상세 방법은 참고자료: 절단 평면을 계산하는 방법 참고)

$$f_i - \sum_{j=m+1}^{n} f_{i,j}x_j \le 0 \qquad f_{i,j} = \bar{a}_{i,j} - \left[\bar{a}_{i,j}\right] \quad f_i = \bar{b}_i - \left[\bar{b}_i\right]$$

$$\frac{5}{3} - \left[\frac{5}{3}\right] - \left(1 - [1]\right)x_1 - \left(\frac{1}{3} - \left[\frac{1}{3}\right]\right)x_3 - \underbrace{\left(-\frac{2}{3} - \left[-\frac{2}{3}\right]\right)}_{-1}x_5 \le 0$$

$$\frac{2}{3} - (0)x_1 - \left(\frac{1}{3}\right)x_3 - \left(\frac{1}{3}\right)x_5 \le 0$$

$$-\frac{1}{3}x_3 - \frac{1}{3}x_5 \le -\frac{2}{3} \quad \text{--- 절단 평면}$$

3. 본 문제의 원래 제약조건(①, ②)으로부터

$$x_3 = 9 - 3x_1 - 2x_2 \qquad ①'$$

$$x_5 = 2 - x_2 \qquad ②'$$

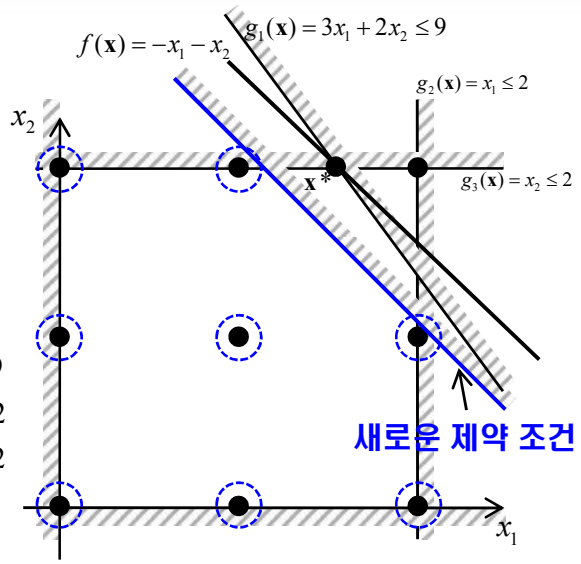
①', ②'을 절단 평면식에 대입 후 정리하면

$$x_1 + x_2 \le 3 \quad \text{--- 절단 평면}$$

4. 본 문제에 절단 평면을 추가하여 문제를 푼다.

$$
\begin{aligned}
\textit{Minimize}\ & f(\mathbf{x}) = -x_1 - x_2 \\
\textit{Subject to}\ & g_1(\mathbf{x}) = 3x_1 + 2x_2 \le 9 \\
& g_2(\mathbf{x}) = x_1 \le 2 \\
& g_3(\mathbf{x}) = x_2 \le 2 \\
& x_j \ge 0,\ \forall j
\end{aligned}
$$

$$
\begin{aligned}
① \quad & 3x_1 + 2x_2 + x_3 = 9 \\
& x_1 + x_4 = 2 \\
② \quad & x_2 + x_5 = 2 \\
& x_j \ge 0,\ \forall j
\end{aligned}
$$



: 정수 가능해
$\mathbf{x}^*$: 최적 해

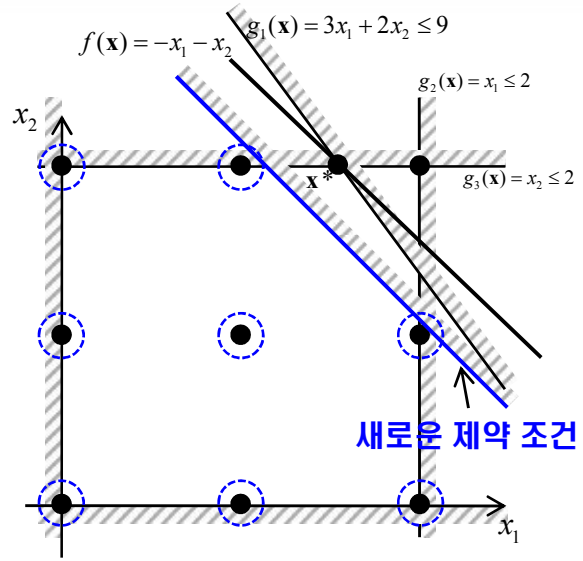
정수 계획법의 해법
- 예) 절단 평면법

1. 선형 계획 문제를 푼다.
2. 정수해가 아니면 절단 평면(제약조건)을 추가 한다.

$$\textit{Minimize} \quad f(\mathbf{x}) = -x_1 - x_2$$

$$\textit{Subject to} \quad g_1(\mathbf{x}) = 3x_1 + 2x_2 \le 9$$
$$g_2(\mathbf{x}) = x_1 \le 2$$
$$g_3(\mathbf{x}) = x_2 \le 2$$
$$g_4(\mathbf{x}) = x_1 + x_2 \le 3 \longleftarrow \text{추가한 절단 평면}$$
$$x_j \ge 0, \ \forall j$$

부등호 제약 조건을 등호 제약 조건으로 변환



○ : 정수 가능해
$\mathbf{x}^*$: 최적 해

$$\textit{Minimize} \quad f(\mathbf{x}) = -x_1 - x_2$$

$$\textit{Subject to} \quad 3x_1 + 2x_2 + x_3 = 9$$
$$x_1 + x_4 = 2$$
$$x_2 + x_5 = 2$$
$$x_1 + x_2 + x_6 = 3$$
$$x_j \ge 0, \ \forall j$$

	x1	x2	x3	x4	x5	x6	bi	bi/ai
x3	3	2	1	0	0	0	9	3
x4	1	0	0	1	0	0	2	2
x5	0	1	0	0	1	0	2	-
x6	1	1	0	0	0	1	3	3
Obj.	-1	-1	0	0	0	f+0	f+0	-

정수 계획법의 해법
- 예) 절단 평면법

1. 선형 계획 문제를 푼다.
2. 정수해가 아니면 절단 평면(제약조건)을 추가 한다.

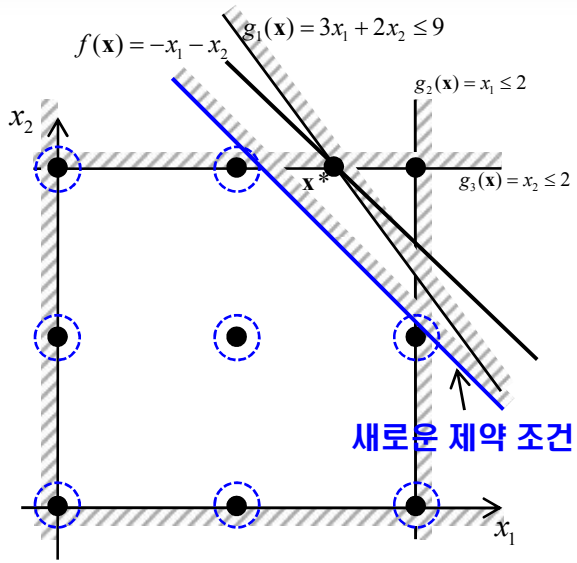
	x1	x2	x3	x4	x5	x6	bi	bi/ai
x3	3	2	1	0	0	0	9	3
x4	1	0	0	1	0	0	2	2
x5	0	1	0	0	1	0	2	-
x6	1	1	0	0	0	1	3	3
Obj.	-1	-1	0	0	0	0	f+0	-

	x1	x2	x3	x4	x5	x6	bi	bi/ai
x3	0	2	1	-3	0	0	3	3/2
x1	1	0	0	1	0	0	2	
x5	0	1	0	0	1	0	2	2
x6	0	1	0	-1	0	1	1	1
Obj.	0	-1	0	1	0	0	f+2	-

	x1	x2	x3	x4	x5	x6	bi	bi/ai
x3	0	0	1	-1	0	-2	1	
x1	1	0	0	1	0	0	2	
x5	0	0	0	1	1	-1	1	
x2	0	1	0	-1	0	1	1	
Obj.	0	0	0	0	0	1	f+3	-

$$\textit{Minimize} \quad f(\mathbf{x}) = -x_1 - x_2$$

$$\textit{Subject to} \quad g_1(\mathbf{x}) = 3x_1 + 2x_2 \le 9$$
$$g_2(\mathbf{x}) = x_1 \le 2$$
$$g_3(\mathbf{x}) = x_2 \le 2$$
$$x_j \ge 0, \ \forall j$$
$$g_4(\mathbf{x}) = x_1 + x_2 \le 3$$



: 정수 가능해

$\mathbf{x}^*$: 최적 해

최적점은 다음과 같다.

$$x_1 = 2, x_2 = 1$$

정수해 이므로 문제 풀이를 종료 한다.

Integer Programming(정수 계획법)
- branch and bound method(분단 탐색법)

Advanced
Ship
Design
Automation
Laboratory

[1)]분단탐색법(branch and bound method) : 열거법의 일종으로서 발생 가능한 전 대안을 평가하는 전체를 조사하지 않고 부분을 조사하는 방법

정수 계획법의 분류

"Stanley Zionts, Linear and integer programming, Prentice-Hall, 1974, p.336"
"Gerard Sierksma, Linear and integer programming, Marcel Dekker, Inc., 1996"
"박순달, 경영과학, 4정판, 민영사, 2003."
"박재흥, 최신경영과학, 시그마그래프, 2004."
"Laurence A. Wolsey, Integer programming, A Wiley-Interscience Publication, 1998"

수학적(Mathematical) 최적화 방법

Cut Algorithm(절단 평면법)

주어진 선형 계획 문제에 정수해는 만족하고, 정수가 아닌 해는 만족하지 않는 제약조건을 추가하면서 문제를 푸는 방법

Enumeration Algorithm(열거법)

주어진 선형 계획 문제의 가능해 영역(feasible region)내의 정수해를 열거하여 최적해를 찾는 방법

예)분지 한계법(분단 탐색법) - Branch and Bound Method

열거법의 일종으로서, 상한(upper bound)와 하한(lower bound)라는 개념을 사용하여 가능한한 가능해를 적게 열거하여 최적해를 찾는 방법

Constructive Algorithm

선형 계획 문제를 특성에 따라 분류하고 그 특성에 맞는 최적 해법을 적용하는 것

문제의 분류	네트워크 이론(Network theory)			할당 문제	●●●
	최단 경로 문제	최소 비용 문제	최대 흐름 문제		
해법	다익스트라(Dijkstra) 해법	그리디(Greedy) 해법	최대 흐름 문제 해법	헝가리법(Hungarian method)	●●●

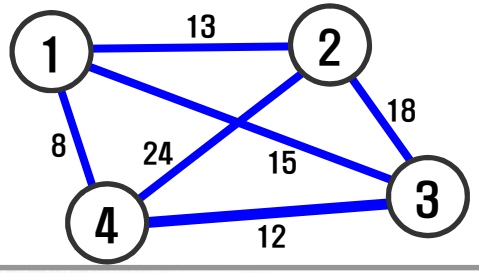
휴리스틱(Heuristic) 최적화 방법

비록 최적해는 구하지 못하지만 최적해에 가까운 좋은 해를 짧은 시간 내에(즉, 적은 계산량으로) 찾으려는 방법

- 개미 알고리즘
- 유전 알고리즘
- 최근거리 인접점(nearest-neighbor) 알고리즘
- 교점교환 알고리즘
- k-optimal 알고리즘
- ●●●

외판원문제 수학적 최적화 모델 정식화

외판원 문제[1)]



목적함수

Minimize 외판원의 총 이동거리 최소화

$$\begin{aligned} F = \; & 13x_{12} + 15x_{13} + 8x_{14} \\ & + 13x_{21} + 18x_{23} + 24x_{24} \\ & + 15x_{31} + 18x_{32} + 12x_{34} \\ & + 8x_{41} + 24x_{42} + 12x_{43} \end{aligned}$$

$$F = \sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}x_{ij}$$
$$(i, j = 1, 2, \cdots N)$$

설계변수

$$x_{12}, x_{13}, x_{14}, x_{21}, x_{23}, x_{24}, x_{31}, x_{32}, x_{34}, x_{41}, x_{42}, x_{43}$$

제약조건

x_{ij} : 노드 i에서 노드 j로 이동하면 1, 그렇지않으면 0

$x_{ij} = 0$ 또는 1

외판원은 각 노드에서 출발하여 단 한 노드에만 갈 수 있다.

$$x_{12}+x_{13}+x_{14}=1 \qquad x_{21}+x_{23}+x_{24}=1$$
$$x_{31}+x_{32}+x_{34}=1 \qquad x_{41}+x_{42}+x_{43}=1$$

외판원은 도착하는 노드를 기준으로 볼 때 단 한 노드에서부터 출발하여 온다.

$$x_{21}+x_{31}+x_{41}=1 \qquad x_{12}+x_{32}+x_{42}=1$$
$$x_{13}+x_{23}+x_{43}=1 \qquad x_{14}+x_{24}+x_{34}=1$$

외판원은 모든 노드를 방문하여 처음 출발지로 돌아와야 한다.

$$x_{12}+x_{21}\le 1 \qquad x_{13}+x_{31}\le 1 \qquad x_{12}+x_{23}+x_{31}\le 2 \qquad x_{12}+x_{24}+x_{41}\le 2$$
$$x_{23}+x_{32}\le 1 \qquad x_{24}+x_{42}\le 1 \qquad x_{13}+x_{32}+x_{21}\le 2 \qquad x_{13}+x_{34}+x_{41}\le 2$$
$$x_{14}+x_{41}\le 1 \qquad x_{34}+x_{43}\le 1 \qquad x_{14}+x_{42}+x_{21}\le 2 \qquad x_{14}+x_{43}+x_{31}\le 2$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad \sum_{i=1}^{N} x_{ij} = 1$$

i노드에서 단 한번만 출발 j노드에 단 한번만 도착

x_{ij}는하나의Hamiltion순환로[2)]를형성

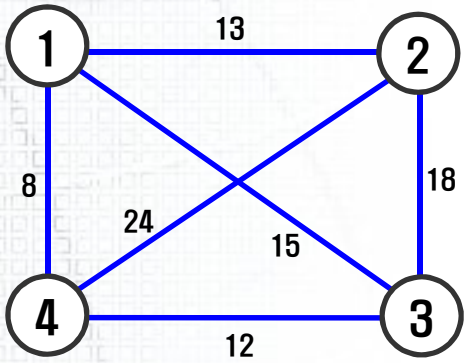
[1)] **외판원 문제** (TSP : Traveling Salesman Problem) : **외판원 문제는 한 명의 외판원이 최단시간에 주어진 노드(고객)들을 정확하게 한번씩 방문하고 다시 출발지점으로 돌아오는 외판원 경로를 찾는 문제이다.**

[2)] Hamilton 순환로 : 모든 노드를 정확히 한번만 지나서 처음 출발한 노드로 가는 path.

"박순달, 경영과학, 4정판, 민영사, 2003, p155 ~ p.162"
"강맹규, 네트워크와 알고리즘, 박영사, 2001"
"Laurence A. Wolsey, Integer programming, A Wiley-Interscience Publication, 1998, p91 ~ p.95"

외판원문제 - 분단탐색법(branch and bound method)

분단 탐색법 : 열거법의 일종으로서 발생 가능한 전 대안을 평가하는 전체를 조사하지 않고 부분을 조사하는 방법.
어느 노드에서 분할이 되어야 하고 어느 점은 분할할 필요가 없는지를 결정하는 기준이 필요.

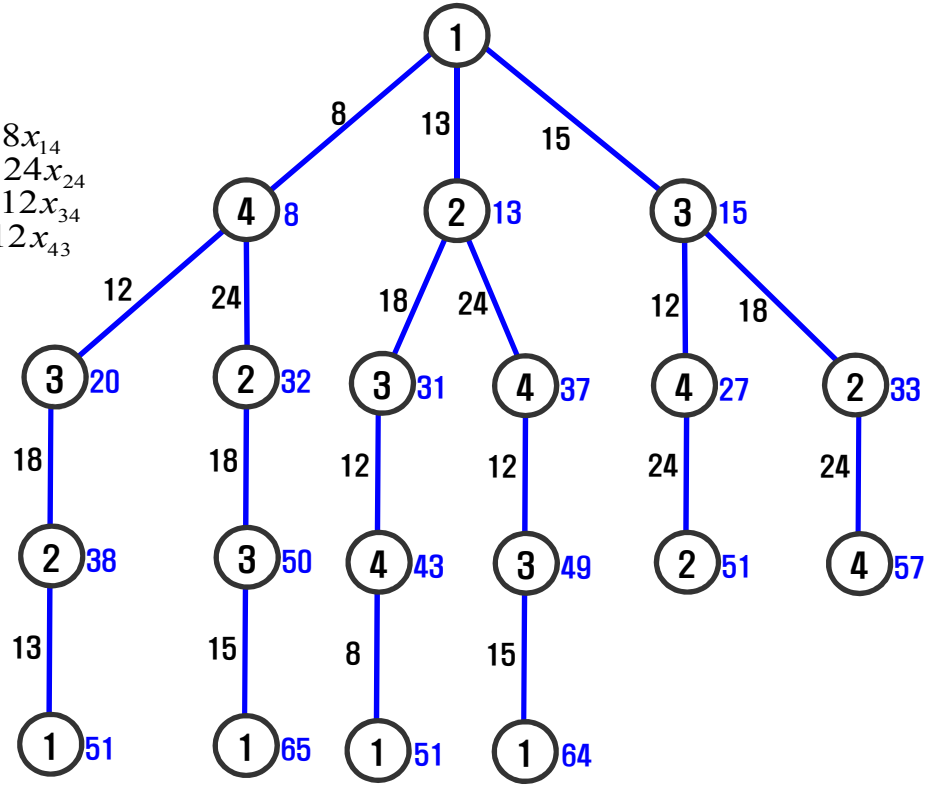


Minimize

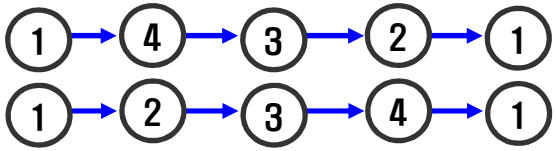
$$F = 13x_{12} + 15x_{13} + 8x_{14} + 13x_{21} + 18x_{23} + 24x_{24} + 15x_{31} + 18x_{32} + 12x_{34} + 8x_{41} + 24x_{42} + 12x_{43}$$

각 노드 사이의 거리

	1	2	3	4
1	∞	13	15	8
2	13	∞	18	24
3	15	18	∞	12
4	8	24	12	∞



방문 순서



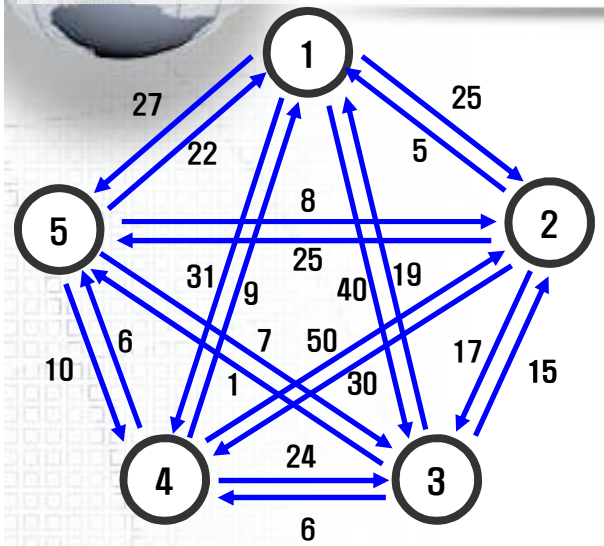
8+12+18+13=51

13+18+12+8=51

분단 탐색 순서

1. 최소해를 가질 가능성이 높은 노드부터 분단(branch)
2. 분단된 노드에서 탐색(bound)으로 해 생성
3. 더 이상 분단할 노드가 없거나, 현 최소해 보다 탐색된 해가 크면 이전 노드로 이동하여 분단 및 탐색 작업 반복

외판원 문제 - 분단탐색법(branch and bound method)
(Little, Murty, Sweeney와 Karel 알고리즘 적용)



각 행의 최소 거리로
해당 행의 거리 차감

	1	2	3	4	5	
1	∞	25	40	31	27	25
2	5	∞	17	30	25	5
3	19	15	∞	6	1	1
4	9	50	24	∞	6	6
5	22	8	7	10	∞	7

25 + 5 + 1 + 6 + 7 + 3 = 47
하한(lower bound) : 47

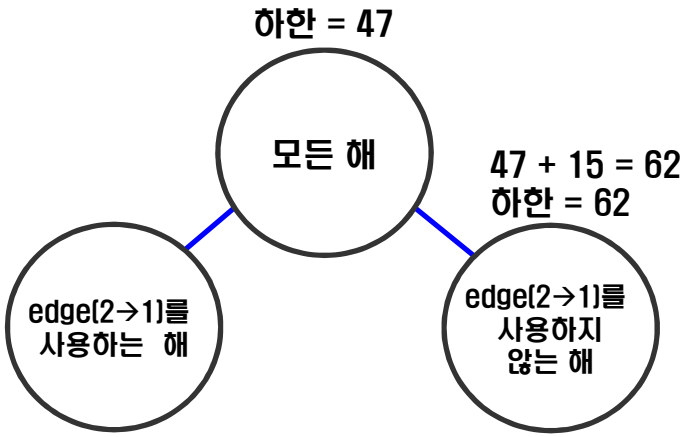
각 열의 최소 거리로
해당 열의 거리 차감

	1	2	3	4	5
1	∞	0	15	6	2
2	0	∞	12	25	20
3	18	14	∞	5	0
4	3	44	18	∞	0
5	15	1	0	3	∞
	0	0	0	3	0

분단을 위한 edge 선정

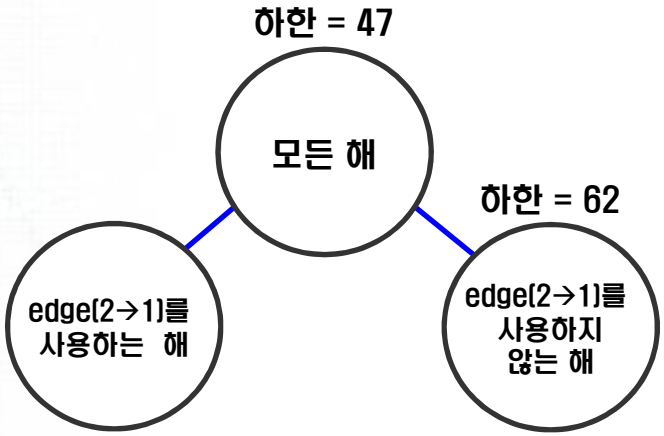
	1	2	3	4	5
1	∞	0	15	3	2
2	0	∞	12	22	20
3	18	14	∞	2	0
4	3	44	18	∞	0
5	15	1	0	0	∞

D_{12} = 2 + 1 = 1
D_{21} = 12 + 3 = 15
D_{35} = 2 + 0 = 2
D_{45} = 3 + 0 = 3
D_{53} = 0 + 12 = 12
D_{54} = 0 + 2 = 2



D_{ij} : i번째 행의 최소 거리와 j번째 열의 최소 거리의 합

외판원 문제 - 분단탐색법(branch and bound method)
(Little, Murty, Sweeney와 Karel 알고리즘 적용)



분단을 위한 edge 선정

	1	2	3	4	5
1	∞	0	15	3	2
2	0	∞	12	22	20
3	18	14	∞	2	0
4	3	44	18	∞	0
5	15	1	0	0	∞

edge(2→1)를 사용하지 않는 해의 하한 구하기

	1	2	3	4	5
1	∞	0	15	3	2
2	∞	∞	12	22	20
3	18	14	∞	2	0
4	3	44	18	∞	0
5	15	1	0	0	∞

12

3

C_{21}= ∞ 로 수정

행과 열 정리 (거리 차감)

edge(2→1)를 사용하지 않는 해의 하한 구하기

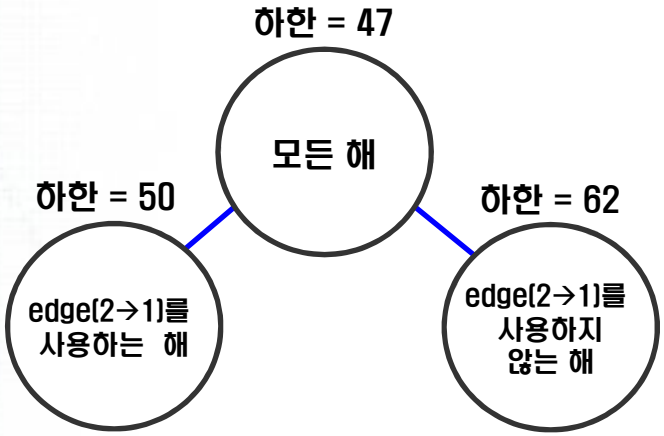
	1	2	3	4	5
1	∞	0	15	3	2
2	∞	∞	0	10	8
3	15	14	∞	2	0
4	0	44	18	∞	0
5	12	1	0	0	∞

행과 열 정리 (거리 차감)

47+12+3=62

하한 = 62

외판원 문제 - 분단탐색법(branch and bound method)
(Little, Murty, Sweeney와 Karel 알고리즘 적용)



분단을 위한 edge 선정

	1	2	3	4	5
1	∞	0	15	3	2
2	0	∞	12	22	20
3	18	14	∞	2	0
4	3	44	18	∞	0
5	15	1	0	0	∞

edge(2→1)를 사용하는 해의 하한 구하기

	1	2	3	4	5
1	∞	∞	15	3	2
2	0	∞	12	22	20
3	18	14	∞	2	0
4	3	44	18	∞	0
5	15	1	0	0	∞

C_{12}= ∞ 로 수정
(1→2는 경로에 포함될 수 없으므로)

2행과 1열 삭제

행과 열 정리 (거리 차감)

	2	3	4	5	
1	∞	15	3	2	2
3	14	∞	2	0	
4	44	18	∞	0	
5	1	0	0	∞	
	1				

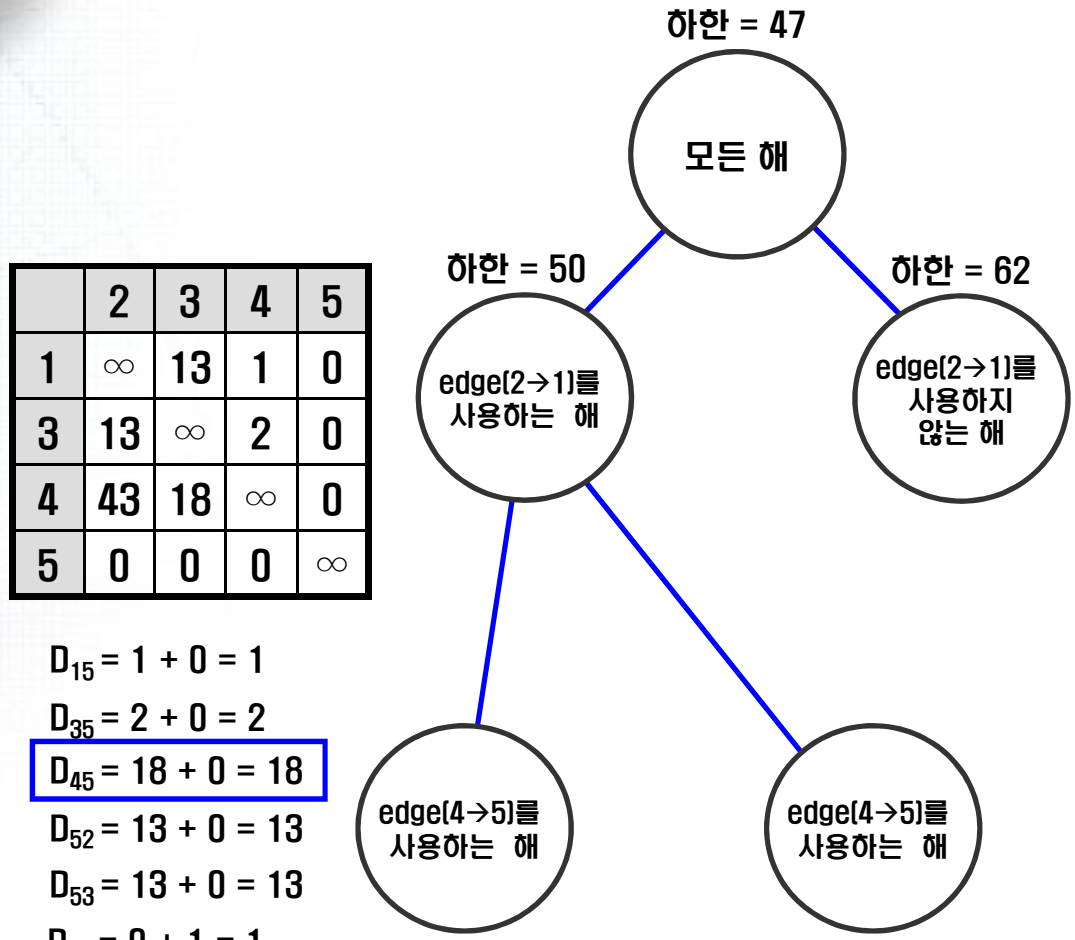
하한 계산

	2	3	4	5
1	∞	13	1	0
3	13	∞	2	0
4	43	18	∞	0
5	0	0	0	∞

47+2+1=50

하한 = 50

외판원 문제 - 분단탐색법(branch and bound method)
(Little, Murty, Sweeney와 Karel 알고리즘 적용)



	2	3	4	5
1	∞	13	1	0
3	13	∞	2	0
4	43	18	∞	0
5	0	0	0	∞

$D_{15} = 1 + 0 = 1$

$D_{35} = 2 + 0 = 2$

$D_{45} = 18 + 0 = 18$

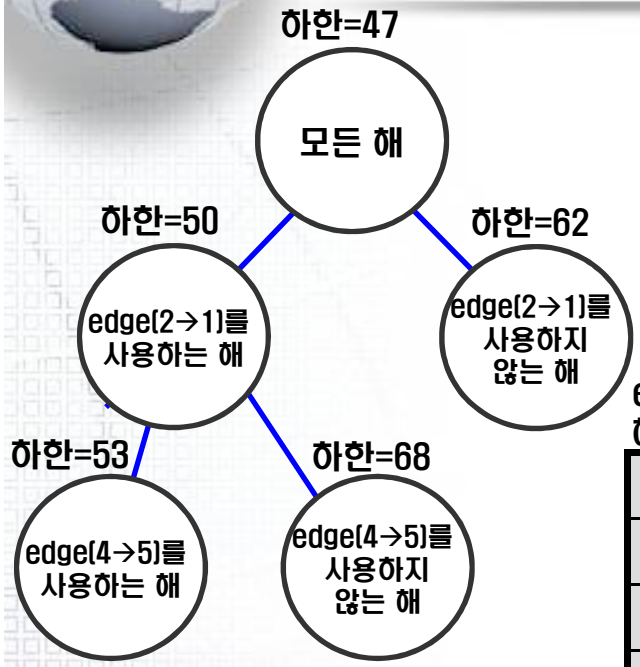
$D_{52} = 13 + 0 = 13$

$D_{53} = 13 + 0 = 13$

$D_{54} = 0 + 1 = 1$

	1	2	3	4	5
1	∞	0	15	3	2
2	∞	∞	0	10	8
3	15	14	∞	2	0
4	0	44	18	∞	0
5	12	1	0	0	∞

외판원 문제 - 분단탐색법(branch and bound method)
(Little, Murty, Sweeney와 Karel 알고리즘 적용)



	2	3	4	5
1	∞	13	1	0
3	13	∞	2	0
4	43	18	∞	0
5	0	0	0	∞

$D_{15} = 1 + 0 = 1$

$D_{35} = 2 + 0 = 2$

$D_{45} = 18 + 0 = 18$

$D_{52} = 13 + 0 = 13$

$D_{53} = 13 + 0 = 13$

$D_{54} = 0 + 1 = 1$

edge(4→5)를 사용하지 않는 해의 하한 구하기

	2	3	4	5
1	∞	13	1	0
3	13	∞	2	0
4	43	18	∞	∞
5	0	0	0	∞

$C_{45} = \infty$ 로 수정

행과 열 정리 (거리 차감)

18

해의 계산

	2	3	4	5
1	∞	13	1	0
3	13	∞	2	0
4	25	0	∞	∞
5	0	0	0	∞

50+18 =50

하한 = 68

edge(4→5)를 사용하는 해의 하한 구하기

	2	3	4	5
1	∞	13	1	0
3	13	∞	2	0
4	43	18	∞	0
5	0	0	0	∞

$C_{54} = \infty$ 로 수정 (5→4는 경로에 포함될 수 없으므로)

4행과 5열 삭제

행과 열 정리 (거리 차감)

	2	3	4	
1	∞	13	1	1
3	13	∞	2	2
5	0	0	∞	

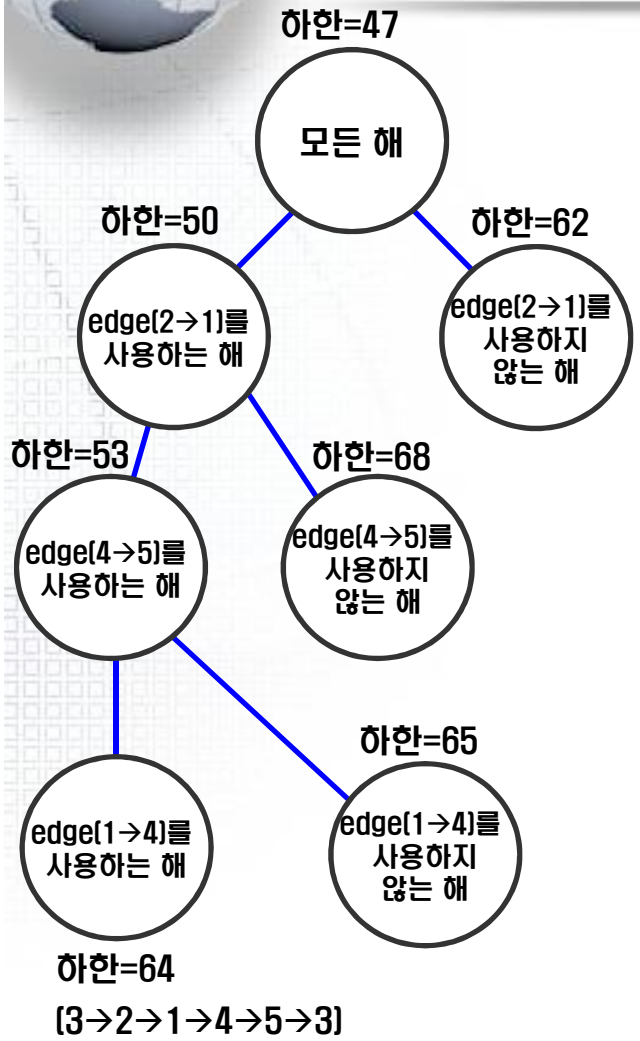
하한 계산

	2	3	4
1	∞	12	0
3	11	∞	0
5	0	0	∞

50+1+2=53

하한 = 53

외판원 문제 - 분단탐색법(branch and bound method)
(Little, Murty, Sweeney와 Karel 알고리즘 적용)



edge(4→5)를 사용하는 해

	2	3	4
1	∞	12	0
3	11	∞	0
5	0	0	∞

$D_{14} = 12 + 0 = 12$

$D_{34} = 11 + 0 = 11$

$D_{52} = 11 + 0 = 11$

$D_{53} = 12 + 0 = 12$

edge(1→4)를 사용하지 않는 해의 하한 구하기

	2	3	4
1	∞	12	∞
3	11	∞	0
5	0	0	∞

12

C_{14}= ∞ 로 수정

행과 열 정리 (거리 차감)

해의 계산

	2	3	4
1	∞	0	∞
3	11	∞	0
5	0	0	∞

53+12 =65

하한 = 65

edge(1→4)를 사용하는 해의 하한 구하기

	2	3	4
1	∞	12	0
3	11	∞	0
5	0	0	∞

1행과 4열 삭제

행과 열 정리 (거리 차감)

	2	3
3	11	∞
5	0	0

11

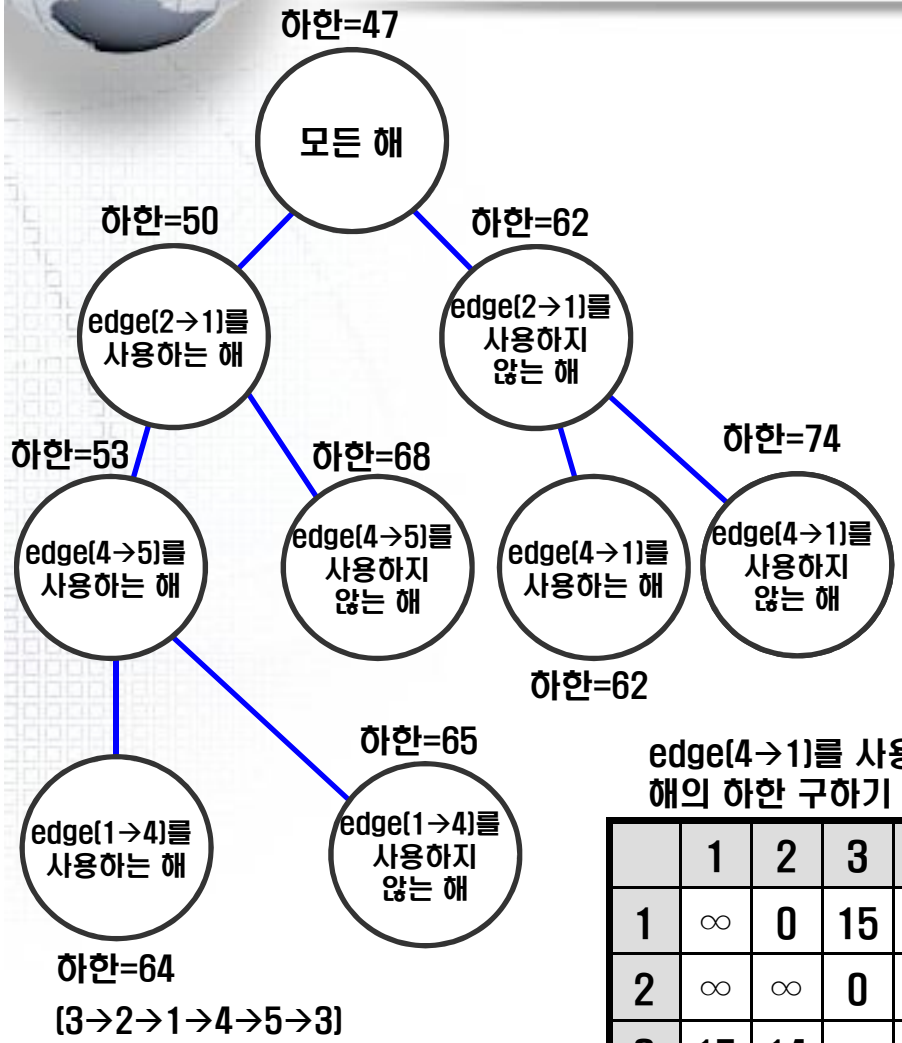
해의 계산

	2	3
3	0	∞
5	0	0

53+11 =64

하한 = 64

외판원 문제 - 분단탐색법(branch and bound method)
(Little, Murty, Sweeney와 Karel 알고리즘 적용)



edge(2→1)를 사용하지 않는 해

	1	2	3	4	5
1	∞	0	15	3	2
2	∞	∞	0	10	8
3	15	14	∞	2	0
4	0	44	18	∞	0
5	12	1	0	0	∞

$D_{12} = 2 + 1 = 3$
$D_{23} = 8 + 0 = 8$
$D_{35} = 2 + 0 = 2$
$D_{41} = 0 + 12 = 12$
$D_{45} = 0 + 0 = 0$
$D_{53} = 0 + 0 = 0$
$D_{54} = 0 + 2 = 2$

edge(4→1)를 사용하지 않는 해의 하한 구하기

62 + 12 = 74
하한 = 74

edge(4→1)를 사용하는 해의 하한 구하기

	1	2	3	4	5
1	∞	0	15	3	2
2	∞	∞	0	10	8
3	15	14	∞	2	0
4	0	44	18	∞	0
5	12	1	0	0	∞

$C_{14} = \infty$ 로 수정
(1→4는 경로에 포함될 수 없으므로)

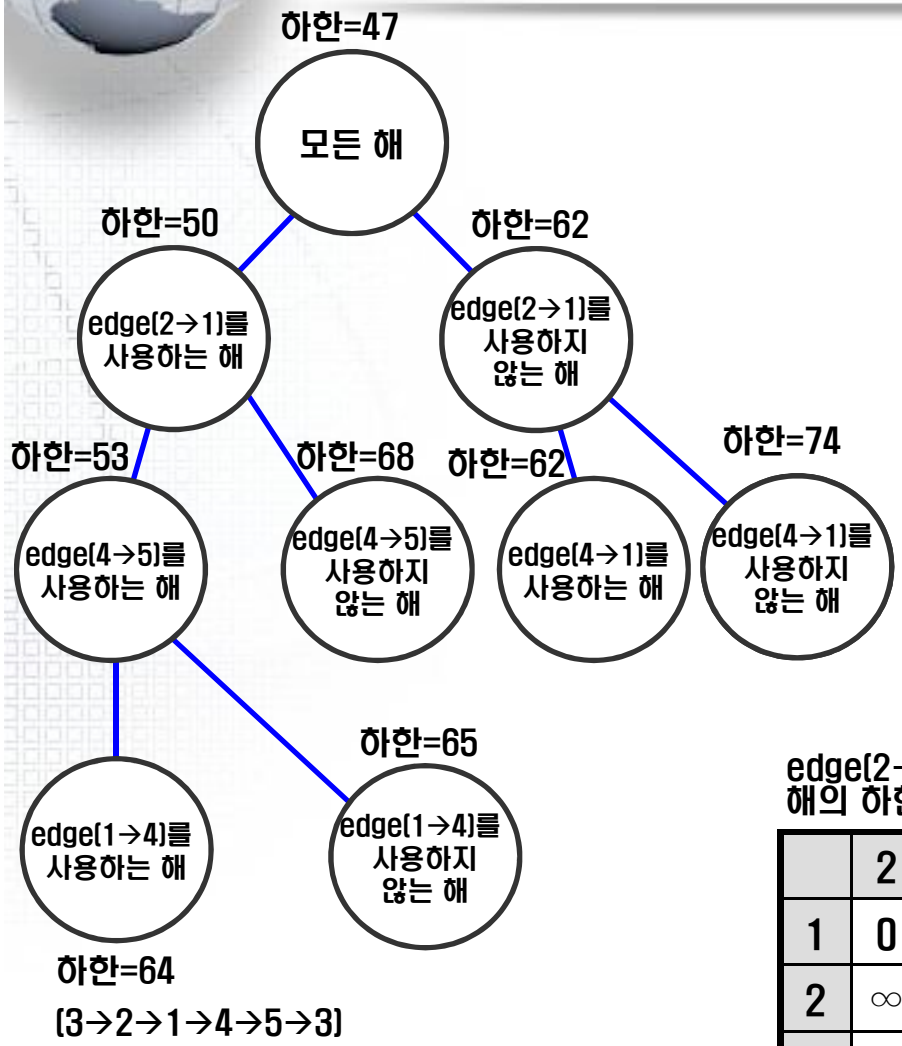
4행과 1열 삭제

해의 계산

	2	3	4	5
1	0	15	∞	2
2	∞	0	10	8
3	14	∞	2	0
5	1	0	0	∞

하한 = 62

외판원 문제 - 분단탐색법(branch and bound method)
(Little, Murty, Sweeney와 Karel 알고리즘 적용)



edge(4→1)를 사용하는 해

	2	3	4	5
1	0	15	∞	2
2	∞	0	10	8
3	14	∞	2	0
5	1	0	0	∞

$D_{12} = 2 + 1 = 3$
$D_{23} = 8 + 0 = 8$
$D_{35} = 2 + 2 = 4$
$D_{53} = 0 + 0 = 0$
$D_{54} = 0 + 2 = 2$

edge(2→3)를 사용하지 않는 해의 하한 구하기 — 62 + 8 = 70, 하한 = 70

edge(2→3)를 사용하는 해의 하한 구하기

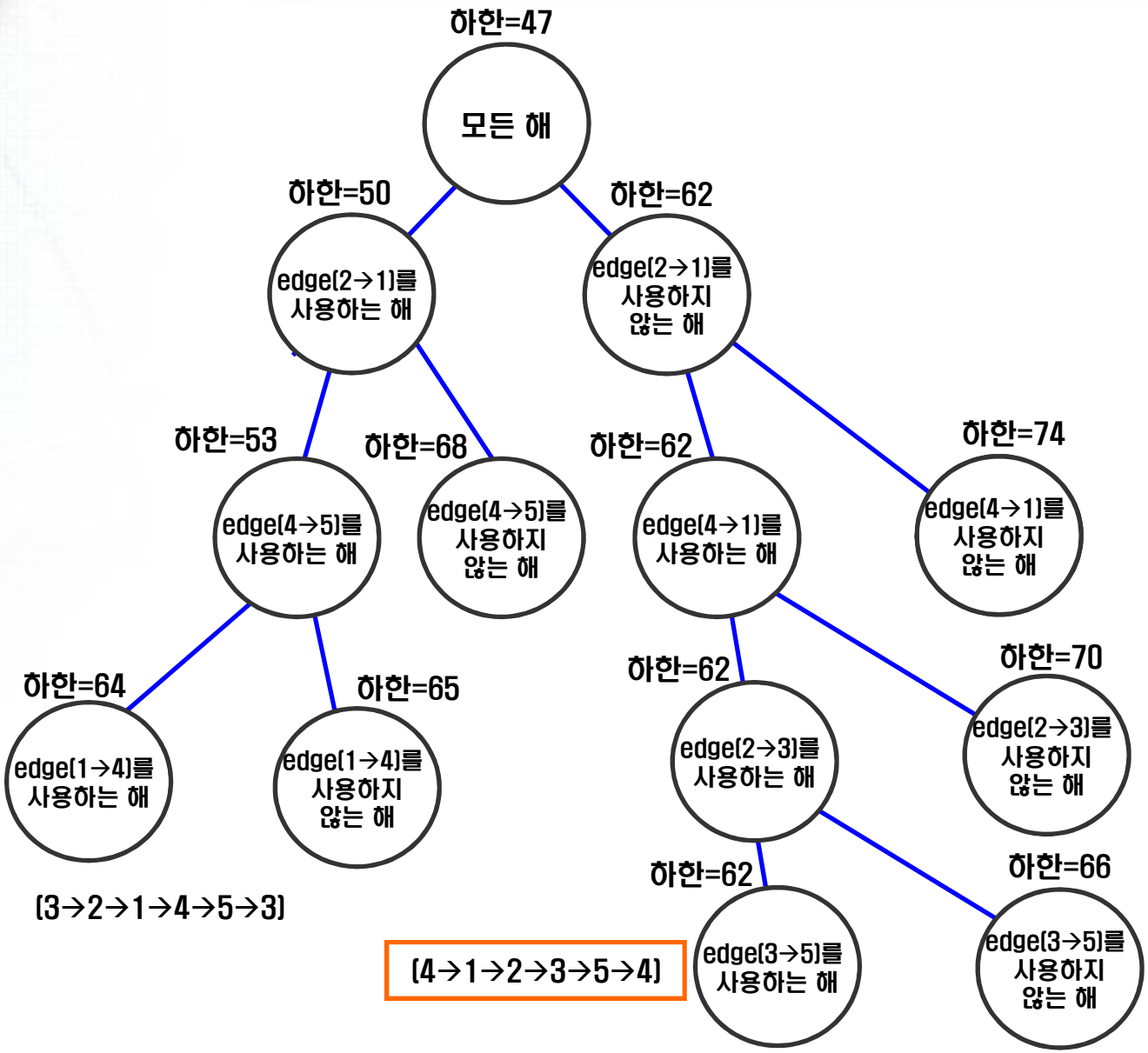
	2	3	4	5
1	0	15	∞	2
2	∞	0	10	8
3	14	∞	2	0
5	1	0	0	∞

C_{32}= ∞ 로 수정
(3→2는 경로에 포함될 수 없으므로)
2행과 3열 삭제

	2	4	5
1	0	∞	2
3	∞	2	0
5	1	0	∞

하한=62

외판원 문제 - 분단탐색법(branch and bound method)
(Little, Murty, Sweeney와 Karel 알고리즘 적용)



외판원 문제 - 분단탐색법(branch and bound method)
(Bellmore-Malone의 알고리즘 적용)

	1	2	3	4	5	6	7	8
1	∞	76	43	38	51	42	19	80
2	42	∞	49	26	78	52	39	87
3	48	28	∞	36	53	44	68	61
4	72	31	29	∞	42	49	50	38
5	30	52	38	47	∞	64	75	82
6	66	51	83	51	22	∞	37	71
7	77	62	93	54	69	38	∞	26
8	42	58	66	76	41	52	83	∞

헝가리법 사용

행과 열 정리
(거리 차감)

	1	2	3	4	5	6	7	8
1	∞	48	14	12	29	4	0	54
2	12	∞	20	0	56	14	20	61
3	18	0	∞	10	31	6	49	35
4	42	3	0	∞	20	11	31	12
5	0	24	9	21	∞	26	56	56
6	36	23	54	25	0	∞	18	45
7	47	34	64	28	47	0	∞	0
8	12	30	37	50	19	14	64	∞

열에서 12 차감

직선수(7)와
노드수(8)가
같지 않음

	1	2	3	4	5	6	7	8
1	∞	48	14	12	29	4	0	54
2	12	∞	20	0	56	14	20	61
3	18	0	∞	10	31	6	49	35
4	42	3	0	∞	20	11	31	12
5	0	24	9	21	∞	26	56	56
6	36	23	54	25	0	∞	18	45
7	47	34	64	28	47	0	∞	0
8	0	18	25	38	7	2	52	∞

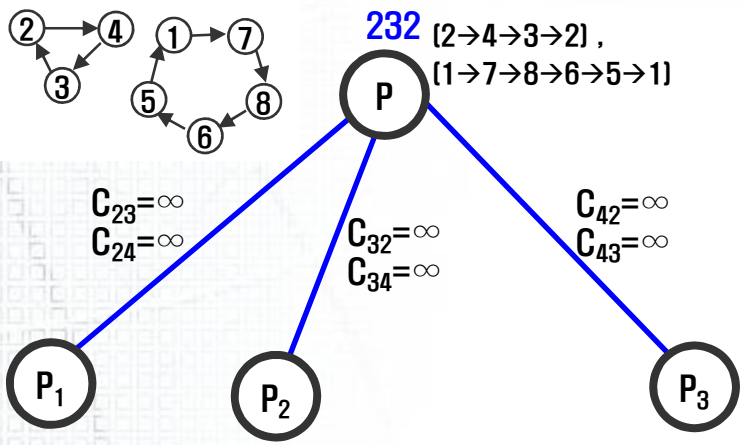
직선으로 삭제되지 않은 가장 작은 거리 인 2를 삭제되지 않은
모든 거리에서 차감, 두 번 지워진 거리에서는 2만큼 더함

	1	2	3	4	5	6	7	8
1	∞	48	14	12	29	2	0	52
2	12	∞	20	0	56	12	20	59
3	18	0	∞	10	31	4	49	33
4	42	3	0	∞	20	9	31	10
5	0	24	9	21	∞	24	56	54
6	36	23	54	25	0	∞	18	43
7	49	36	66	30	49	0	∞	0
8	0	18	25	38	7	0	52	∞

외판원 경로
생성하지 못함

2 → 4 → 3 → 2, 1 → 7 → 8 → 6 → 5 → 1
26 + 29 + 28 + 19 + 26 + 52 + 22 + 30 = 232

외판원 문제 - 분단탐색법(branch and bound method)
(Bellmore-Malone의 알고리즘 적용)



	1	2	3	4	5	6	7	8
1	∞	48	14	12	29	2	0	52
2	12	∞	20	0	56	12	20	59
3	18	0	∞	10	31	4	49	33
4	42	3	0	∞	20	9	31	10
5	0	24	9	21	∞	24	56	54
6	36	23	54	25	0	∞	18	43
7	49	36	66	30	49	0	∞	0
8	0	18	25	38	7	0	52	∞

$C_{23}=\infty$,
$C_{24}=\infty$

2행에서 가장 작은 거리 12 선택하여 해당 행 거리에서 차감
4열에서 가장 작은 거리 10 선택하여 해당 열 거리에서 차감

	1	2	3	4	5	6	7	8
1	∞	48	14	2	29	2	0	52
2	0	∞	∞	∞	44	0	8	47
3	18	0	∞	0	31	4	49	33
4	42	3	0	∞	20	9	31	10
5	0	24	9	11	∞	24	56	54
6	36	23	54	15	0	∞	18	43
7	49	36	66	20	49	0	∞	0
8	0	18	25	28	7	0	52	∞

직선 수(7)와 노드 수(8)가 같지 않음

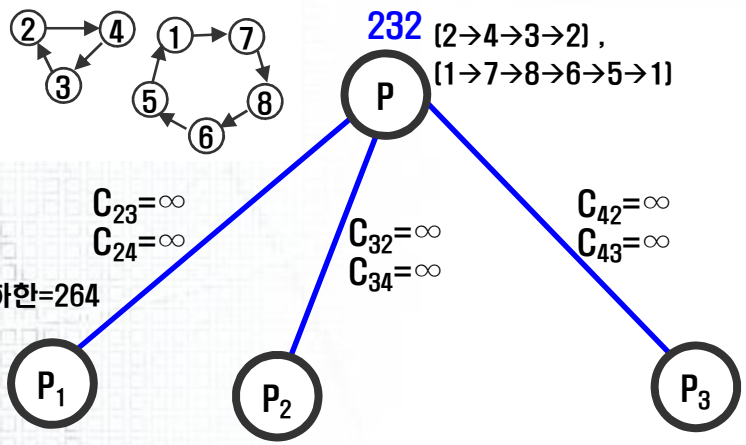
직선으로 삭제되지 않은 가장 작은 거리인 2를 삭제되지 않은 모든 거리에서 차감, 두 번 지워진 거리에서는 2만큼 더함

	1	2	3	4	5	6	7	8
1	∞	46	14	0	29	2	0	52
2	0	∞	∞	∞	44	0	8	47
3	20	0	∞	0	33	6	51	35
4	42	1	0	∞	20	9	31	10
5	0	22	9	9	∞	24	56	54
6	36	21	54	13	0	∞	18	43
7	49	34	66	18	49	0	∞	0
8	0	16	25	26	7	0	52	∞

직선 수(7)와 노드 수(8)가 같지 않음

직선으로 삭제되지 않은 가장 작은 거리인 1를 삭제되지 않은 모든 거리에서 차감, 두 번 지워진 거리에서는 1만큼 더함

외판원 문제 - 분단탐색법(branch and bound method)
(Bellmore-Malone의 알고리즘 적용)



	1	2	3	4	5	6	7	8
1	∞	46	14	0	29	2	0	52
2	0	∞	∞	∞	44	0	8	47
3	20	0	∞	0	33	0	51	35
4	42	1	0	∞	20	9	31	10
5	0	22	9	9	∞	24	56	54
6	36	21	54	13	0	∞	18	43
7	49	34	66	18	49	0	∞	0
8	0	16	25	26	7	0	52	∞

직선 수(7)와 노드 수(8)가 같지 않음

직선으로 삭제되지 않은 가장 작은 거리인 1를 삭제되지 않은 모든 거리에서 차감, 두 번 지워진 거리에서는 1만큼 더함

	1	2	3	4	5	6	7	8
1	∞	46	15	0	30	10	0	60
2	0	∞	∞	∞	37	0	0	47
3	28	0	∞	0	34	14	51	43
4	49	0	0	∞	20	16	30	17
5	0	14	2	1	∞	24	48	54
6	43	20	54	12	0	∞	17	50
7	49	26	59	10	42	0	∞	0
8	0	8	18	18	0	0	44	∞

직선 수(8)와 노드 수(8)가 같음

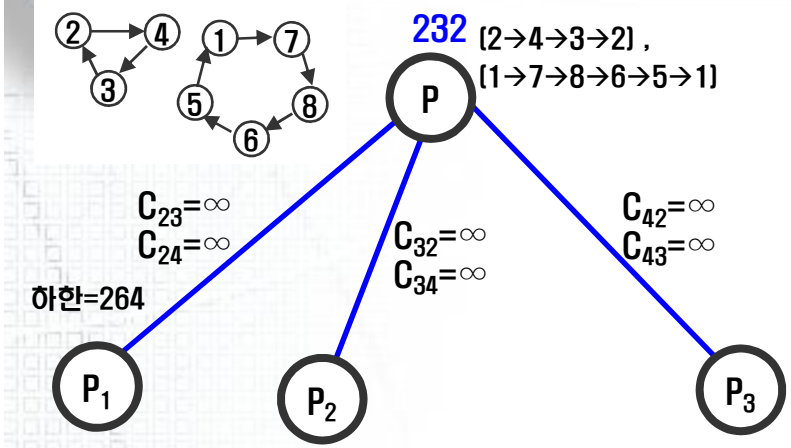
	1	2	3	4	5	6	7	8
1	∞	46	15	0	30	9	0	59
2	0	∞	∞	∞	44	0	7	47
3	21	0	∞	0	34	7	51	36
4	42	0	0	∞	20	9	38	16
5	0	21	9	8	∞	24	55	54
6	36	20	54	12	0	∞	17	43
7	49	33	66	17	49	0	∞	0
8	0	15	25	25	7	0	51	∞

직선 수(7)와 노드 수(8)가 같지 않음

직선으로 삭제되지 않은 가장 작은 거리인 7을 삭제되지 않은 모든 거리에서 차감, 두 번 지워진 거리에서는 7만큼 더함

1 → 4 → 3 → 2 → 7 → 8 → 6 → 5 → 1
38 + 29 + 28 + 39 + 26 + 52 + 22 + 30 = 264

외판원 문제 - 분단탐색법(branch and bound method)
(Bellmore-Malone의 알고리즘 적용)



	1	2	3	4	5	6	7	8
1	∞	48	14	12	29	2	0	52
2	12	∞	20	0	56	12	20	59
3	18	0	∞	10	31	4	49	33
4	42	3	0	∞	20	9	31	10
5	0	24	9	21	∞	24	56	54
6	36	23	54	25	0	∞	18	43
7	49	36	66	30	49	0	∞	0
8	0	18	25	38	7	0	52	∞

2 → 4 → 3 → 2, 1 → 7 → 8 → 6 → 5 → 1
26 + 29 + 28 + 19 + 26 + 52 + 22 + 30 = 232

	1	2	3	4	5	6	7	8
1	∞	48	14	12	29	2	0	52
2	12	∞	20	0	56	12	20	59
3	18	∞	∞	∞	31	4	49	33
4	42	3	0	∞	20	9	31	10
5	0	24	9	21	∞	24	56	54
6	36	23	54	25	0	∞	18	43
7	49	36	66	30	49	0	∞	0
8	0	18	25	38	7	0	52	∞

$C_{32}=\infty$,
$C_{34}=\infty$

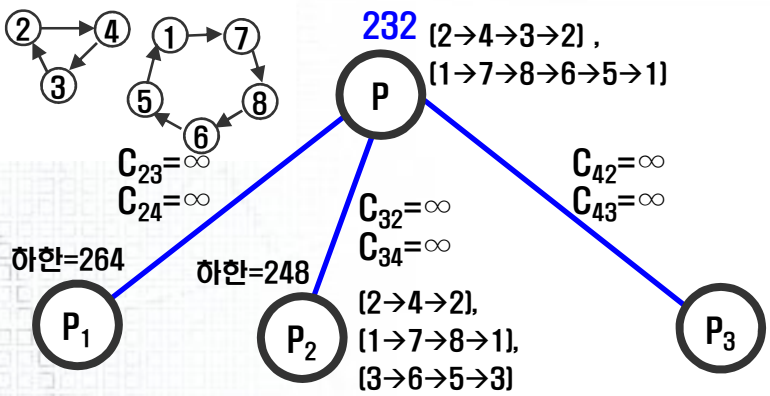
3행에서 가장 작은 거리 4 선택하여 해당 행 거리에서 삭제
2열에서 가장 작은 거리 3 선택하여 해당 열 거리에서 삭제

	1	2	3	4	5	6	7	8
1	∞	45	14	12	29	2	0	52
2	12	∞	20	0	56	12	20	59
3	14	∞	∞	∞	27	0	45	29
4	42	0	0	∞	20	9	31	10
5	0	21	9	21	∞	24	56	54
6	36	20	54	25	0	∞	18	43
7	49	33	66	30	49	0	∞	0
8	0	15	25	38	7	0	52	∞

직선 수(7)와
노드 수(8)가
같지 않음

직선으로 삭제되지 않은 가장 작은 거리인 9를 삭제되지 않은
모든 거리에서 차감, 두 번 지워진 거리에서는 9만큼 더함

외판원 문제 - 분단탐색법(branch and bound method)
(Bellmore-Malone의 알고리즘 적용)



	1	2	3	4	5	6	7	8
1	∞	45	14	12	29	2	0	52
2	12	∞	20	0	56	12	20	59
3	14	∞	∞	∞	27	0	45	29
4	42	0	0	∞	29	9	31	19
5	0	21	9	21	∞	24	56	54
6	36	20	54	25	0	∞	18	43
7	49	33	66	30	49	0	∞	0
8	0	15	25	38	7	0	52	∞

직선 수(7)와
노드 수(8)가
같지 않음

직선으로 삭제되지 않은 가장 작은 거리인 9를 삭제되지 않은
모든 거리에서 차감, 두 번 지워진 거리에서는 9만큼 더함

	1	2	3	4	5	6	7	8
1	∞	36	5	12	29	2	0	52
2	12	∞	11	0	56	12	20	59
3	14	∞	∞	∞	27	0	45	29
4	51	0	0	∞	29	18	40	19
5	0	12	0	21	∞	24	56	54
6	36	11	45	25	0	∞	18	43
7	49	24	57	30	49	0	∞	0
8	0	6	16	38	7	0	52	∞

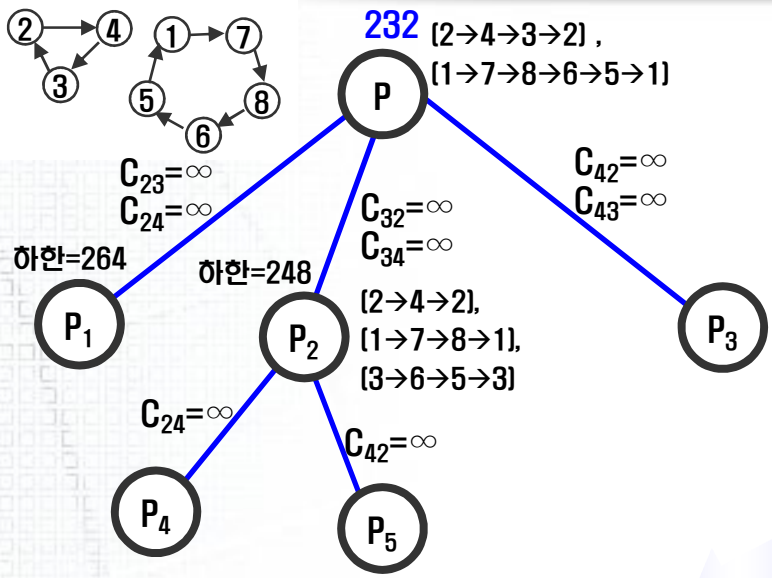
직선 수(8)와
노드 수(8)가
같음

2 → 4 → 2,
1 → 7 → 8 → 1,
3 → 6 → 5 → 3

26 + 31
+ 19 + 26 + 42
+ 44 + 22 + 38
= 248

P_2 하한=248

외판원 문제 - 분단탐색법(branch and bound method)
(Bellmore-Malone의 알고리즘 적용)



P_2

	1	2	3	4	5	6	7	8
1	∞	36	5	12	29	2	0	52
2	12	∞	11	0	56	12	20	59
3	14	∞	∞	∞	27	0	45	29
4	51	0	0	∞	29	18	40	19
5	0	12	0	21	∞	24	56	54
6	36	11	45	25	0	∞	18	43
7	49	24	57	30	49	0	∞	0
8	0	6	16	38	7	0	52	∞

$C_{24}=\infty$

	1	2	3	4	5	6	7	8
1	∞	36	5	12	29	2	0	52
2	12	∞	11	∞	56	12	20	59
3	14	∞	∞	∞	27	0	45	29
4	51	0	0	∞	29	18	40	19
5	0	12	0	21	∞	24	56	54
6	36	11	45	25	0	∞	18	43
7	49	24	57	30	49	0	∞	0
8	0	6	16	38	7	0	52	∞

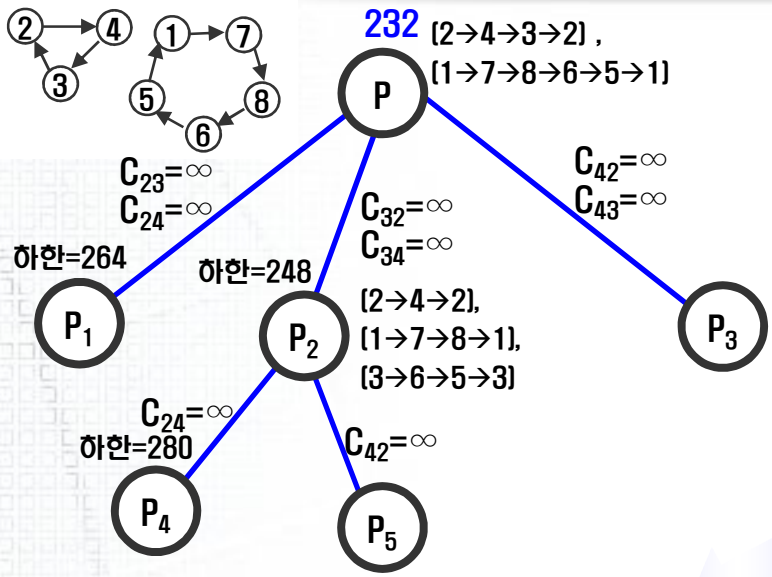
2행에서 가장 작은 거리 11 선택하여 해당 행 거리에서 삭제
4열에서 가장 작은 거리 12 선택하여 해당 열 거리에서 삭제

	1	2	3	4	5	6	7	8
1	∞	36	5	0	29	2	0	52
2	1	∞	0	∞	45	1	9	48
3	14	∞	∞	∞	27	0	45	29
4	51	0	0	∞	29	18	40	19
5	0	12	0	9	∞	24	56	54
6	36	11	45	13	0	∞	18	43
7	49	24	57	18	49	0	∞	0
8	0	6	16	26	7	0	52	∞

직선 수(7)와 노드 수(8)가 같지 않음

직선으로 삭제되지 않은 가장 작은 거리인 9를 삭제되지 않은 모든 거리에서 차감, 두 번 지워진 거리에서는 9만큼 더함

외판원 문제 - 분단탐색법(branch and bound method)
(Bellmore-Malone의 알고리즘 적용)



	1	2	3	4	5	6	7	8
1	∞	36	5	0	29	2	0	52
2	1	∞	0	∞	45	1	9	43
3	14	∞	∞	∞	27	0	45	29
4	51	0	0	∞	29	18	40	19
5	0	12	0	9	∞	24	56	54
6	36	11	45	13	0	∞	18	43
7	49	24	57	18	49	0	∞	0
8	0	6	16	26	7	0	52	∞

직선 수(7)와 노드 수(8)가 같지 않음

직선으로 삭제되지 않은 가장 작은 거리인 9를 삭제되지 않은 모든 거리에서 차감, 두 번 지워진 거리에서는 9만큼 더함

	1	2	3	4	5	6	7	8
1	∞	45	14	0	38	11	0	61
2	1	∞	0	∞	45	1	0	43
3	14	∞	∞	∞	27	0	36	29
4	51	0	0	∞	29	18	31	19
5	0	12	0	0	∞	24	47	54
6	36	11	45	4	0	∞	9	43
7	49	24	57	9	49	0	∞	0
8	0	6	16	17	7	0	43	∞

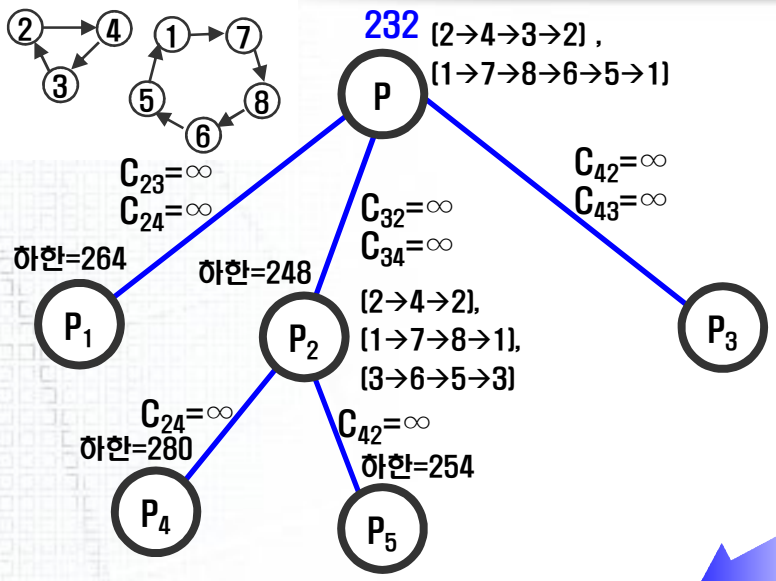
직선 수(8)와 노드 수(8)가 같음

1 → 4 → 2 → 7 → 8 → 1,
3 → 6 → 5 → 3

38 + 31 + 39 + 26 + 42
+ 44 + 22 + 38
= 280

P_4 하한=280

외판원 문제 - 분단탐색법(branch and bound method)
(Bellmore-Malone의 알고리즘 적용)



P_2

	1	2	3	4	5	6	7	8
1	∞	36	5	12	29	2	0	52
2	12	∞	11	0	56	12	20	59
3	14	∞	∞	∞	27	0	45	29
4	51	0	0	∞	29	18	40	19
5	0	12	0	21	∞	24	56	54
6	36	11	45	25	0	∞	18	43
7	49	24	57	30	49	0	∞	0
8	0	6	16	38	7	0	52	∞

2 → 4 → 2,
1 → 7 → 8 → 1,
3 → 6 → 5 → 3

26 + 31
+ 19 + 26 + 42
+ 44 + 22 + 38
= 248

	1	2	3	4	5	6	7	8
1	∞	36	5	12	29	2	0	52
2	12	∞	11	0	56	12	20	59
3	14	∞	∞	∞	27	0	45	29
4	51	∞	0	∞	29	18	40	19
5	0	12	0	21	∞	24	56	54
6	36	11	45	25	0	∞	18	43
7	49	24	57	30	49	0	∞	0
8	0	6	16	38	7	0	52	∞

$C_{42}=\infty$

2열에서 가장 작은 거리 6 선택하여 해당 열 거리에서 삭제

	1	2	3	4	5	6	7	8
1	∞	30	5	12	29	2	0	52
2	12	∞	11	0	56	12	20	59
3	14	∞	∞	∞	27	0	45	29
4	51	∞	0	∞	29	18	40	19
5	0	6	0	21	∞	24	56	54
6	36	5	45	25	0	∞	18	43
7	49	18	57	30	49	0	∞	0
8	0	0	16	38	7	0	52	∞

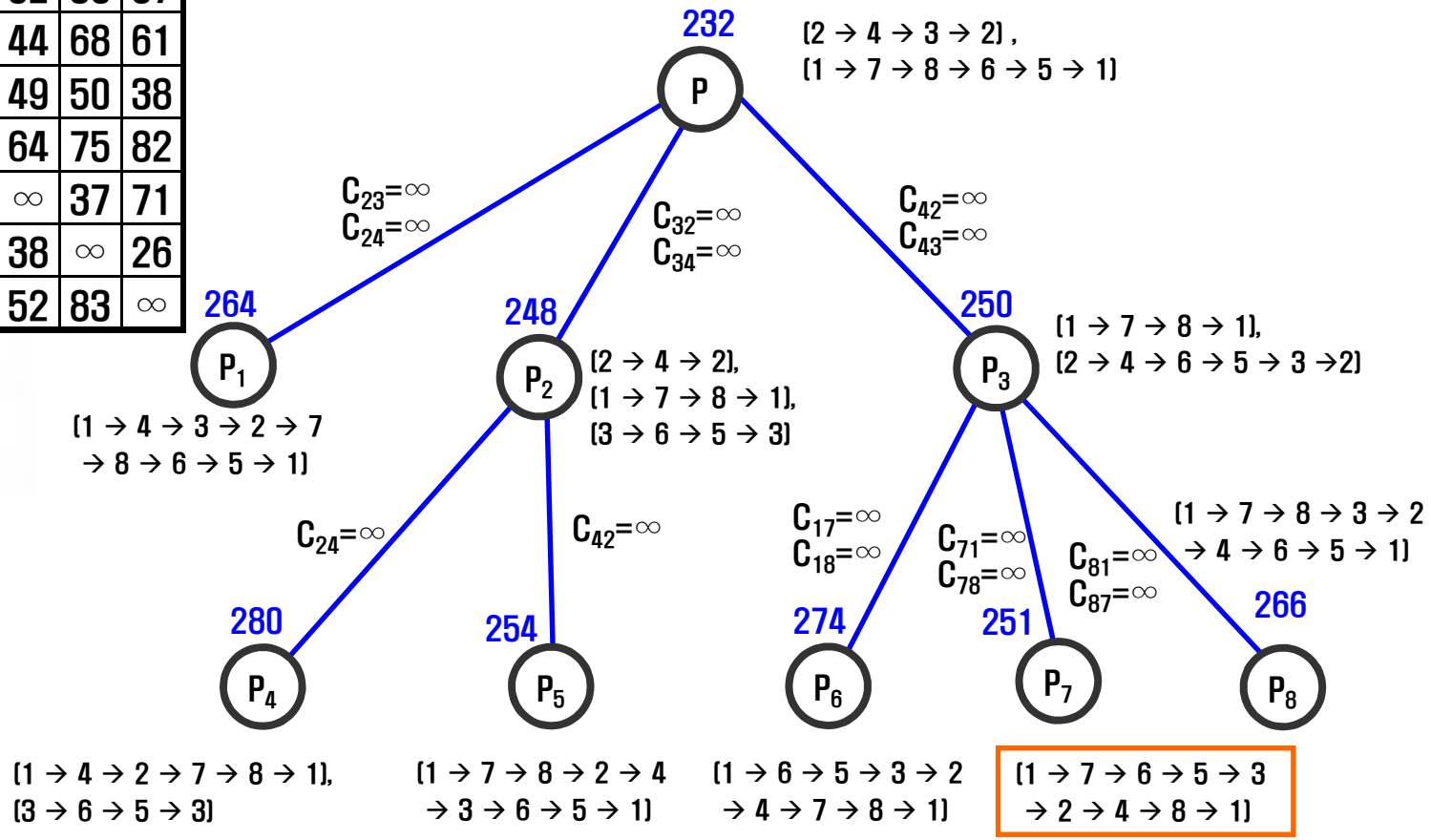
직선 수(8)와
노드 수(8)가
같음

1 → 7 → 8 → 2 → 4 → 3 → 6 → 5 → 1
19 + 26 + 58 + 26 + 29 + 44 + 22 + 30 = 254

P_5 하한=254

외판원 문제 - 분단탐색법(branch and bound method)
(Bellmore-Malone의 알고리즘 적용)

	1	2	3	4	5	6	7	8
1	∞	76	43	38	51	42	19	80
2	42	∞	49	26	78	52	39	87
3	48	28	∞	36	53	44	68	61
4	72	31	29	∞	42	49	50	38
5	30	52	38	47	∞	64	75	82
6	66	51	83	51	22	∞	37	71
7	77	62	93	54	69	38	∞	26
8	42	58	66	76	41	52	83	∞



Integer Programming(정수 계획법)

- Network theory(네트워크 이론)

Advanced
Ship
Design
Automation
Laboratory

정수 계획법의 분류

"Stanley Zionts, Linear and integer programming, Prentice-Hall, 1974, p.336"
"Gerard Sierksma, Linear and integer programming, Marcel Dekker, Inc., 1996"
"박순달, 경영과학, 4정판, 민영사, 2003."
"박재흥, 최신경영과학, 시그마그래프, 2004."
"Laurence A. Wolsey, Integer programming, A Wiley-Interscience Publication, 1998"

수학적(Mathematical) 최적화 방법

Cut Algorithm(절단 평면법)

주어진 선형 계획 문제에 정수해는 만족하고, 정수가 아닌 해는 만족하지 않는 제약조건을 추가하면서 문제를 푸는 방법

Enumeration Algorithm(열거법)

주어진 선형 계획 문제의 가능해 영역(feasible region)내의 정수해를 열거하여 최적해를 찾는 방법

예)분지 한계법(분단 탐색법) - Branch and Bound Method

열거법의 일종으로서, 상한(upper bound)와 하한(lower bound)라는 개념을 사용하여 가능한한 가능해를 적게 열거하여 최적해를 찾는 방법

Constructive Algorithm

선형 계획 문제를 특성에 따라 분류하고 그 특성에 맞는 최적 해법을 적용하는 것

문제의 분류	네트워크 이론(Network theory)			할당 문제	●●●
	최단 경로 문제	최소 비용 문제	최대 흐름 문제		
해법	다익스트라(Dijkstra) 해법	그리디(Greedy) 해법	최대 흐름 문제 해법	헝가리법(Hungarian method)	●●●

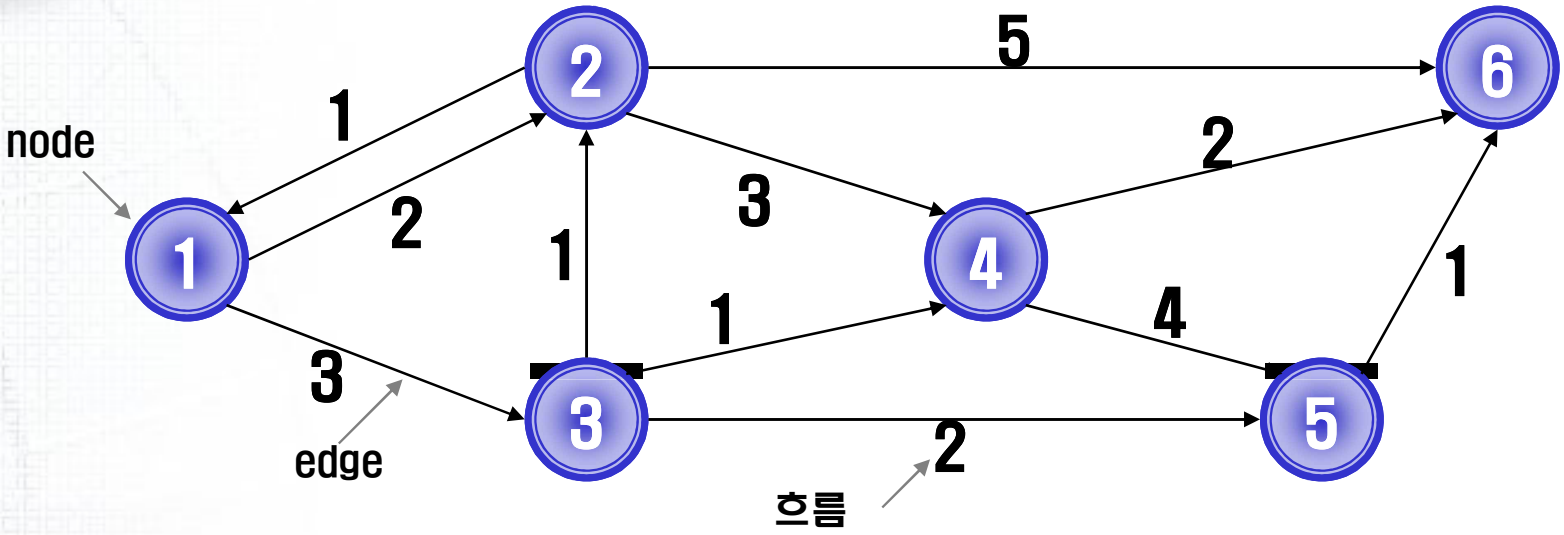
휴리스틱(Heuristic) 최적화 방법

비록 최적해는 구하지 못하지만 최적해에 가까운 좋은 해를 짧은 시간 내에(즉, 적은 계산량으로) 찾으려는 방법

개미 알고리즘	유전 알고리즘	최근거리 인접점 (nearest-neighbor) 알고리즘	교점교환 알고리즘	k-optimal 알고리즘	●●●

네트워크

네트워크의 구성요소와 구조



- 네트워크의 종류
  - 단방향 네트워크(directed network)
  - 양방향 네트워크(undirected network)
- 경로(path, route) : 위 그림의 ①→③→②→④ 와 같은 연속되는 edge
- 고리(loop, cycle) : ①→③→②→①과 같이 처음으로 되돌아오는 경로
- 나무(tree) 네트워크 : 고리가 없는 네트워크

"박재흥, 최신경영과학, 시그마그래프, 2004, p.163 ~ p.168"

대표적인 3가지 네트워크 모형

- 최단경로문제(shortest route problem)
  : 두 지점 사이의 최단경로(가장 작은 비용 또는 가장 짧은 거리나 시간에 도착할 수 있는 경로)를 찾는 문제

- 최소걸침나무문제(minimum spanning tree problem)
  : 네트워크상의 모든 연결하는 방법 중에서 가장 작은 비용 또는 시간으로 연결할 수 있는 방법을 찾는 문제
  (설비배치문제, 네트워크 설계문제)

- 최대흐름문제(maximal flow problem)
  : 네트워크상의 한 지점에서 다른 지점으로 보낼 수 있는 최대의 유통량을 찾는 문제(교통흐름 분석문제, 송유관 설계문제)

최단 경로 문제
-다익스트라법(Dijkstra method)

"박재흥, 최신경영과학, 시그마그래프, 2004, p.168 ~ p.172"

다익스트라법(Dijkstra method) – 최단 경로문제의 대표적 해법

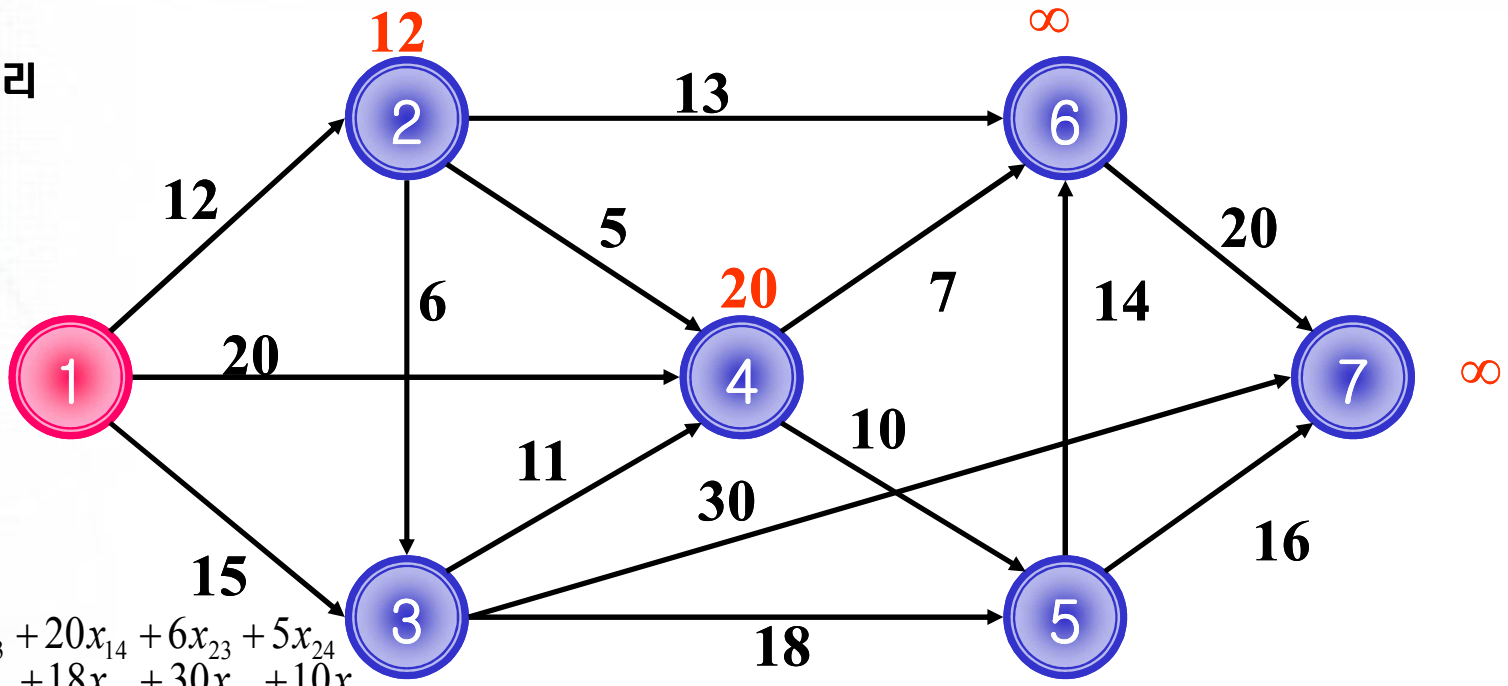
1단계 처음에 출발node를 선택하여 각 node까지의 **임시최단거리**를 표시하되, 직접 연결되는 edge가 없으면 ∞로 표시한다.

2단계 선택되지 않은 node에 대하여, 가장 작은 임시최단거리를 갖는 node를 선택하고 연결하여 **영구최단거리**로 삼는다.
도착node가 선택되면 끝내고, 아니면 3단계로 간다.

3단계 선택되지 않은 node에 대해, 직전에 선택된 node와 연결될 때의 거리가 기존의 임시최단거리보다 작으면 임시최단거리를 수정하여 2단계로 간다.

최단 경로 문제
-다익스트라법(Dijkstra method)

초기 임시최단거리



- ⑤, ⑥, ⑦번 node는 직접 연결되는 경로가 없으므로 임시최단거리를 ∞로 한다.

목적함수

Minimize

$$F = 12x_{12} + 15x_{13} + 20x_{14} + 6x_{23} + 5x_{24} + 13x_{26} + 11x_{34} + 18x_{35} + 30x_{37} + 10x_{45} + 7x_{46} + 14x_{56} + 16x_{57} + 20x_{67}$$

설계변수

$$x_{12}, x_{13}, x_{14}, x_{23}, x_{24}, x_{26}, x_{34}, x_{35}, x_{37}, x_{45}, x_{46}, x_{56}, x_{57}, x_{67}$$

제약조건

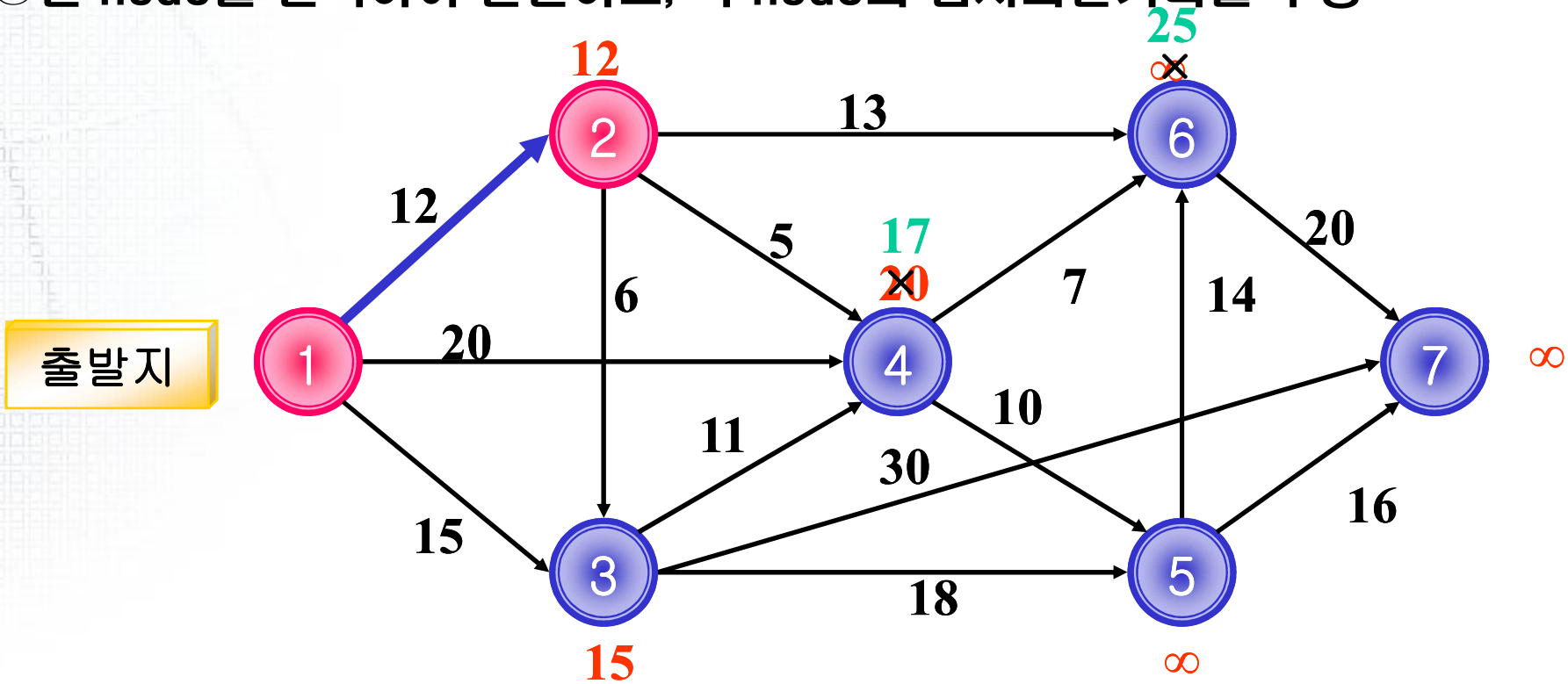
$$x_{12} + x_{13} + x_{14} = 1$$
$$x_{23} + x_{24} + x_{26} - x_{12} = 0$$
$$x_{34} + x_{35} + x_{37} - x_{23} - x_{13} = 0$$
$$x_{45} + x_{46} - x_{14} - x_{24} - x_{34} = 0$$
$$x_{56} + x_{57} - x_{35} - x_{45} = 0$$
$$x_{67} - x_{26} - x_{46} - x_{56} = 0$$
$$-x_{37} - x_{57} - x_{67} = -1$$

$$x_{ij} \geq 0, \forall i, j$$

$$x_{ij} = 0 \text{ 또는 } 1, \forall i, j$$

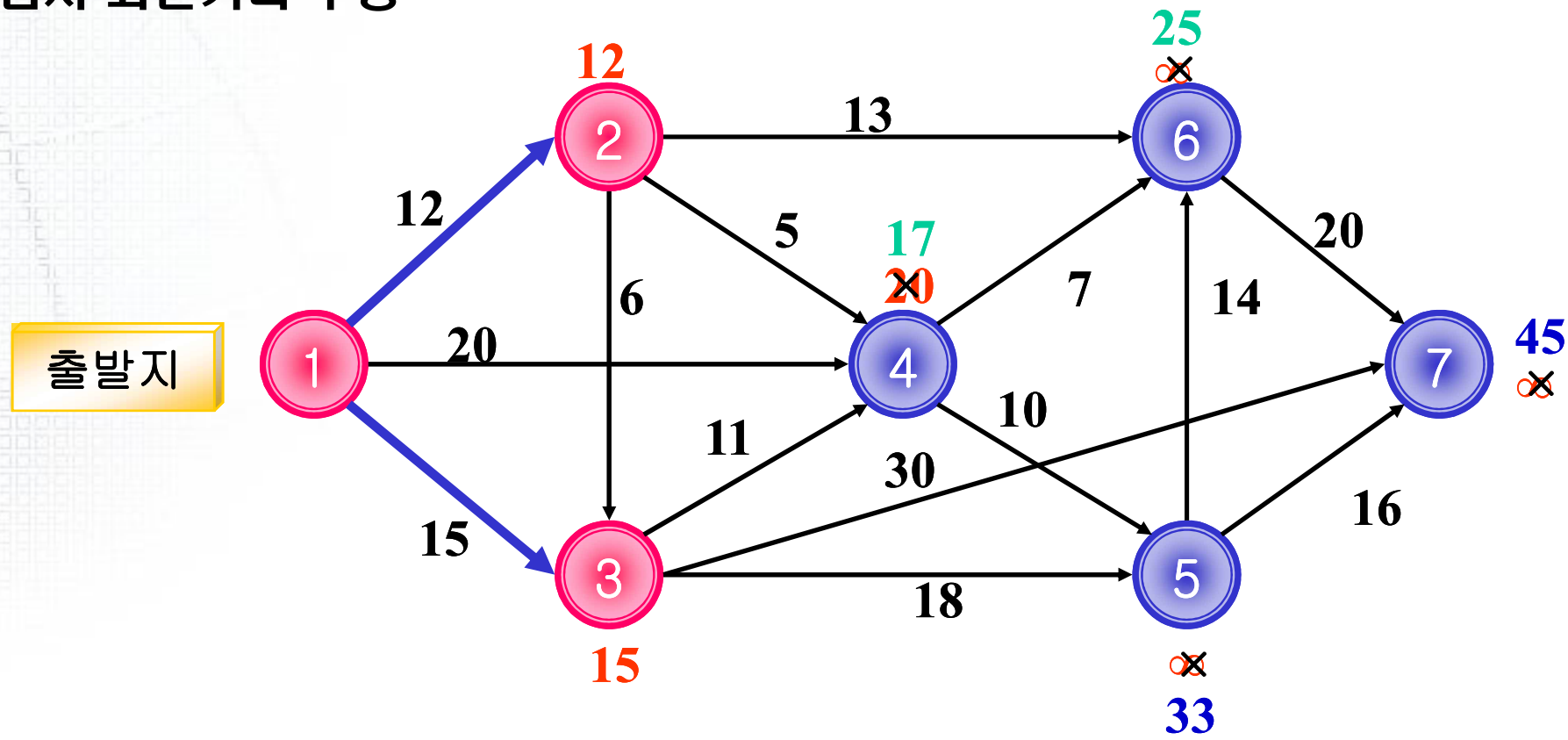
최단 경로 문제
-다익스트라법(Dijkstra method)

임시 최단거리 중 ②번 node의 임시최단거리가 12로 가장 작으므로
②번 node를 선택하여 연결하고, 각 node의 임시최단거리를 수정



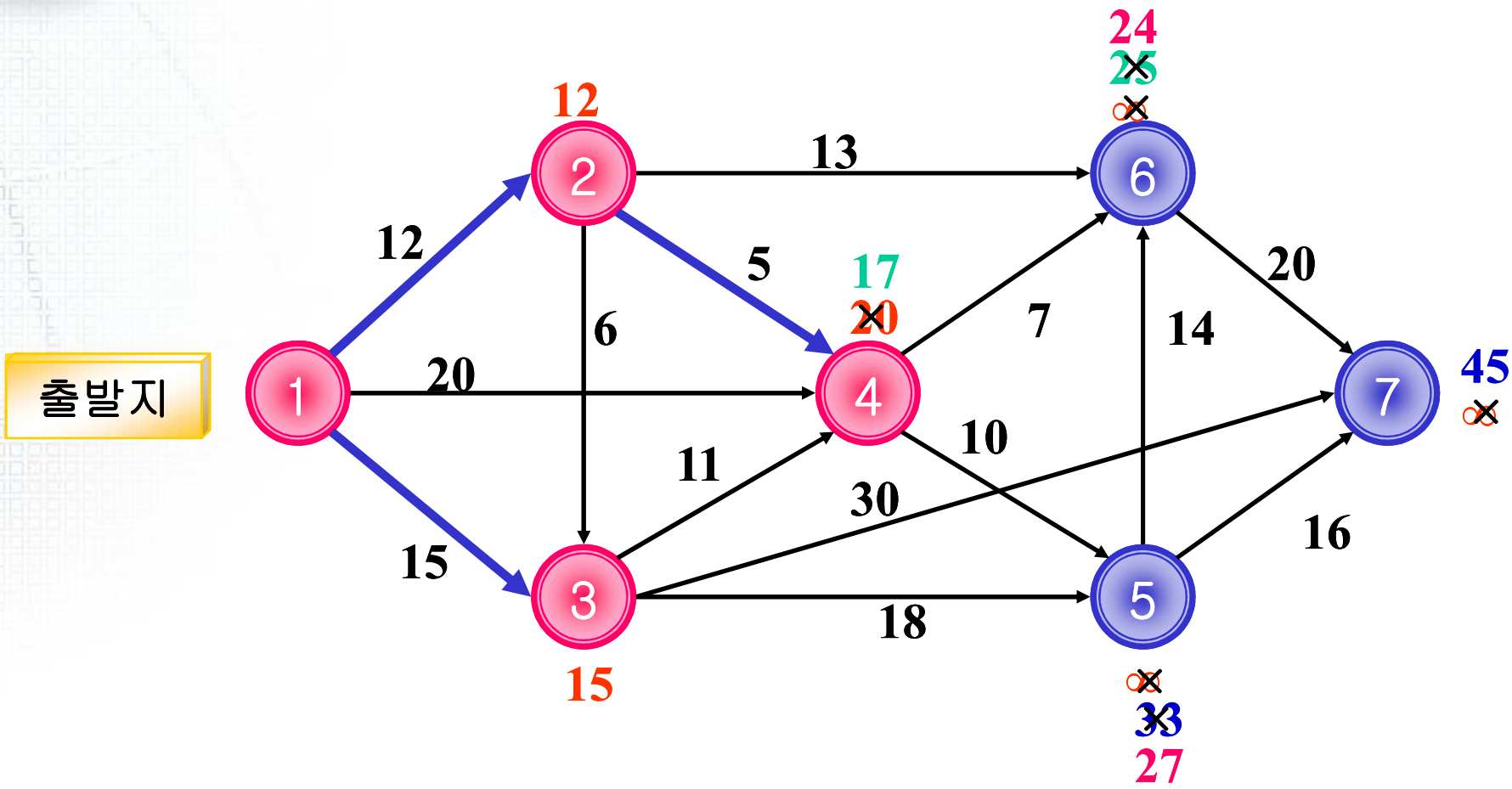
최단 경로 문제
-다익스트라법(Dijkstra method)

임시 최단거리 중 가장 작은 값(15)을 갖는 ③번 node를 선택하여 연결하고, 임시 최단거리 수정



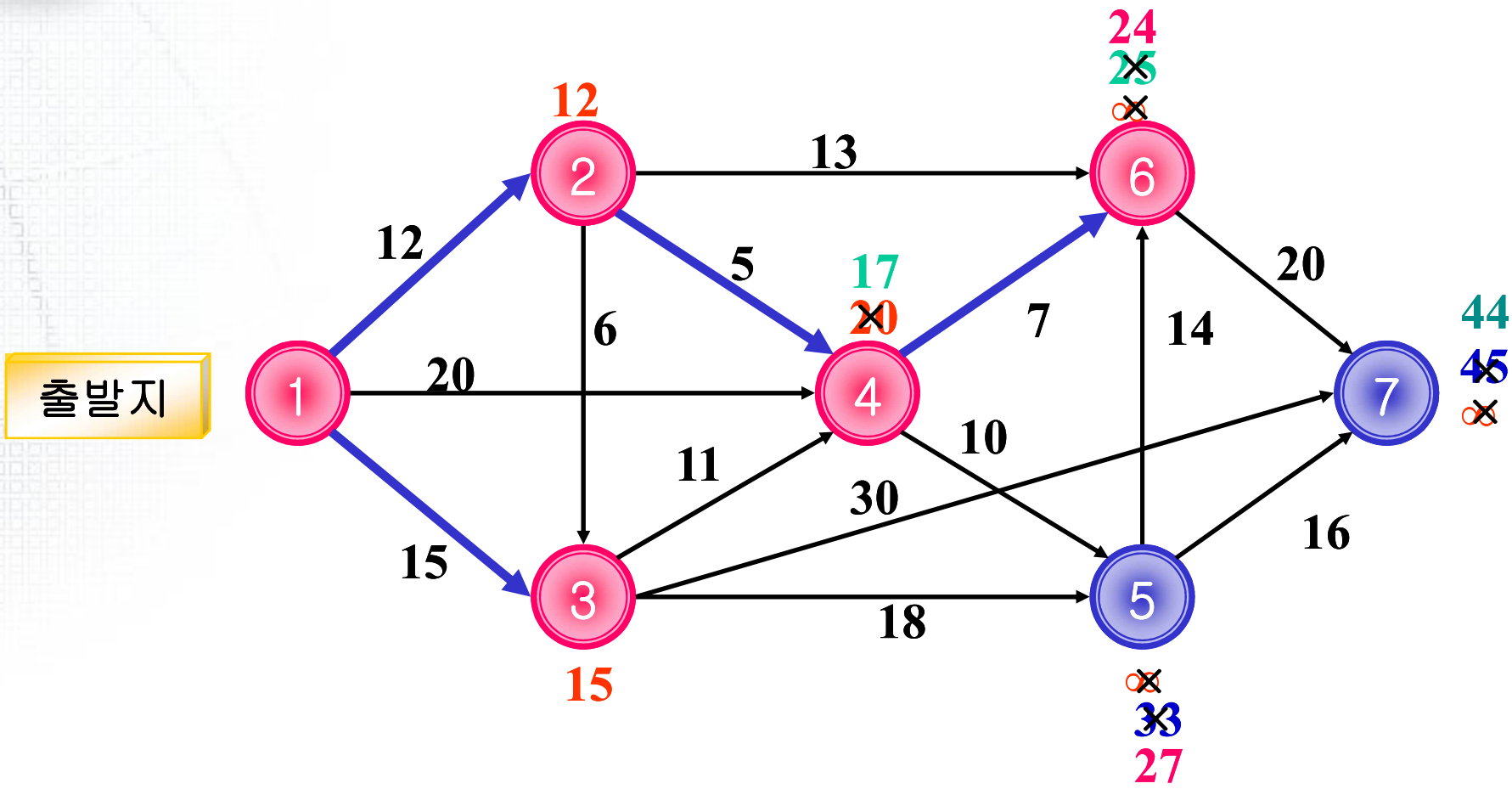
최단 경로 문제
-다익스트라법(Dijkstra method)

세 번째 수정된 임시최단거리



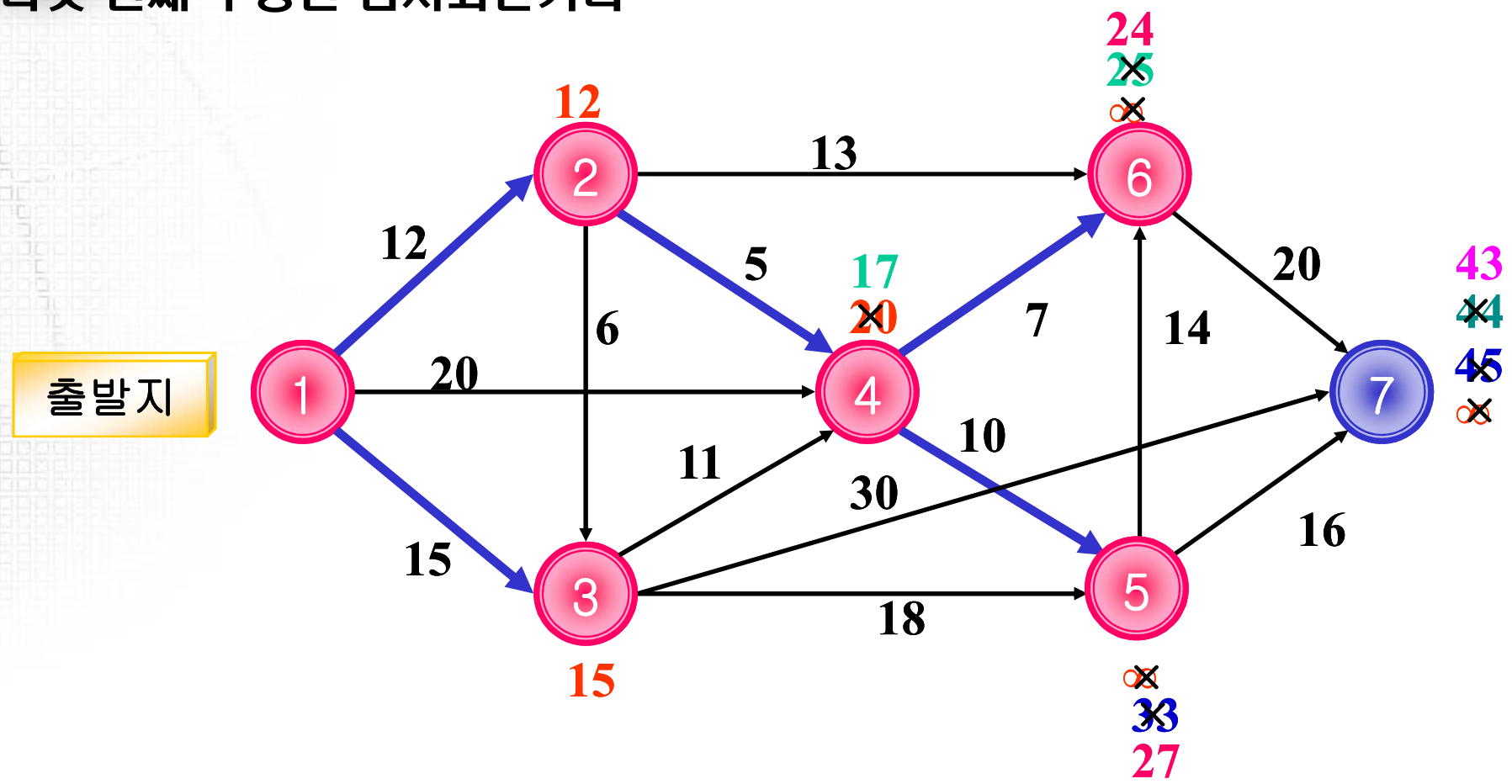
최단 경로 문제
-다익스트라법(Dijkstra method)

네 번째 수정된 임시최단거리



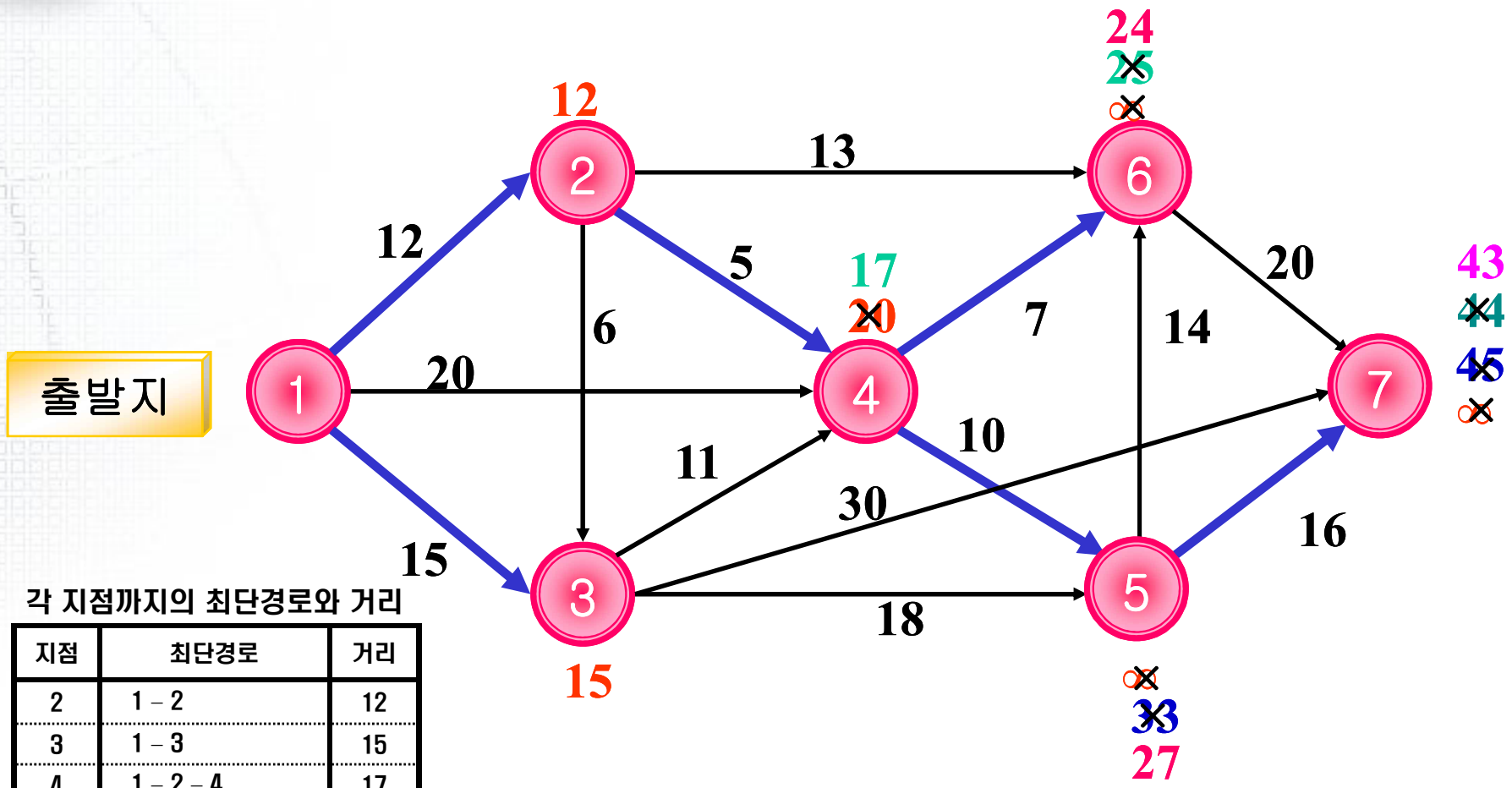
최단 경로 문제
-다익스트라법(Dijkstra method)

다섯 번째 수정된 임시최단거리



최단 경로 문제
-다익스트라법(Dijkstra method)

최종적으로 얻어진 각 지점까지의 최단경로와 최단거리



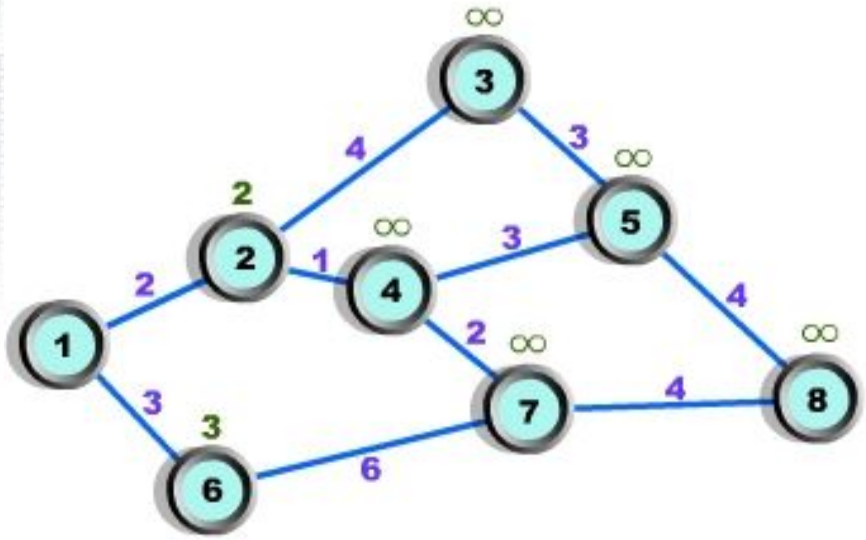
각 지점까지의 최단경로와 거리

지점	최단경로	거리
2	1 – 2	12
3	1 – 3	15
4	1 – 2 – 4	17
5	1 – 2 – 4 – 5	27
6	1 – 2 – 4 – 6	24
7	1 – 2 – 4 – 5 – 7	43

다익스트라(Dijkstra) 알고리즘

- 가중 그래프에서 출발점 V에서부터 다른 모든 vertex까지 가중치 값이 최소인 거리를 찾는 방법.
  즉, 항상 가장 가까운 거리를 갖는 vertex로의 edge를 선택한다.
  그리고, 각 인접 vertex에 대해 오직 하나의 후보 edge만을 추적한다.

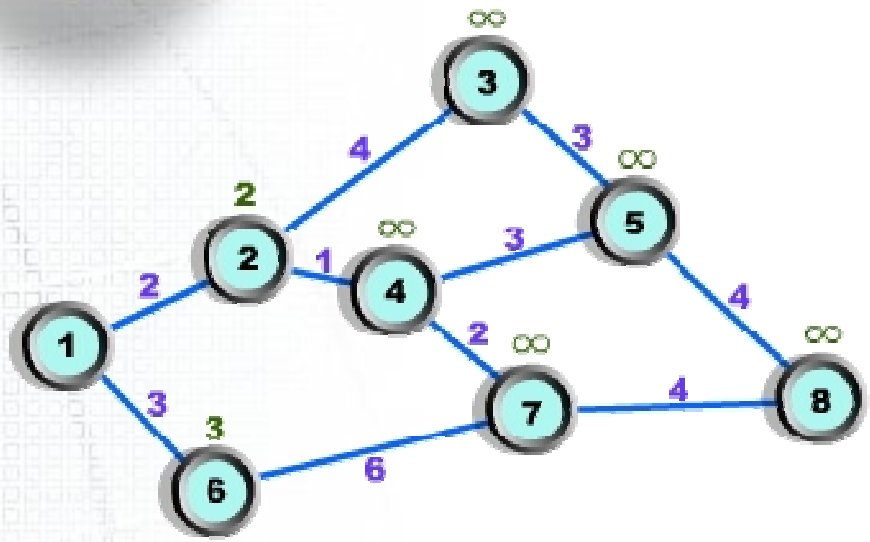
다익스트라 알고리즘 동작원리



옆과 같은 그래프에서 1에서 시작해서 8로 가는 최단거리를 다익스트라 알고리즘을 이용해서 구해보자.

먼저 위의 그래프를 인접행렬로 바꾼다.

다익스트라(Dijkstra) 알고리즘 – 동작원리(1)



	1	2	3	4	5	6	7	8
1	0	2	∞	∞	∞	3	∞	∞
2	2	0	4	1	∞	∞	∞	∞
3	∞	4	0	∞	3	∞	∞	∞
4	∞	1	∞	0	3	∞	2	∞
5	∞	∞	3	3	0	∞	∞	4
6	3	∞	∞	∞	∞	0	6	∞
7	∞	∞	∞	2	∞	6	0	4
8	∞	∞	∞	∞	4	∞	4	0

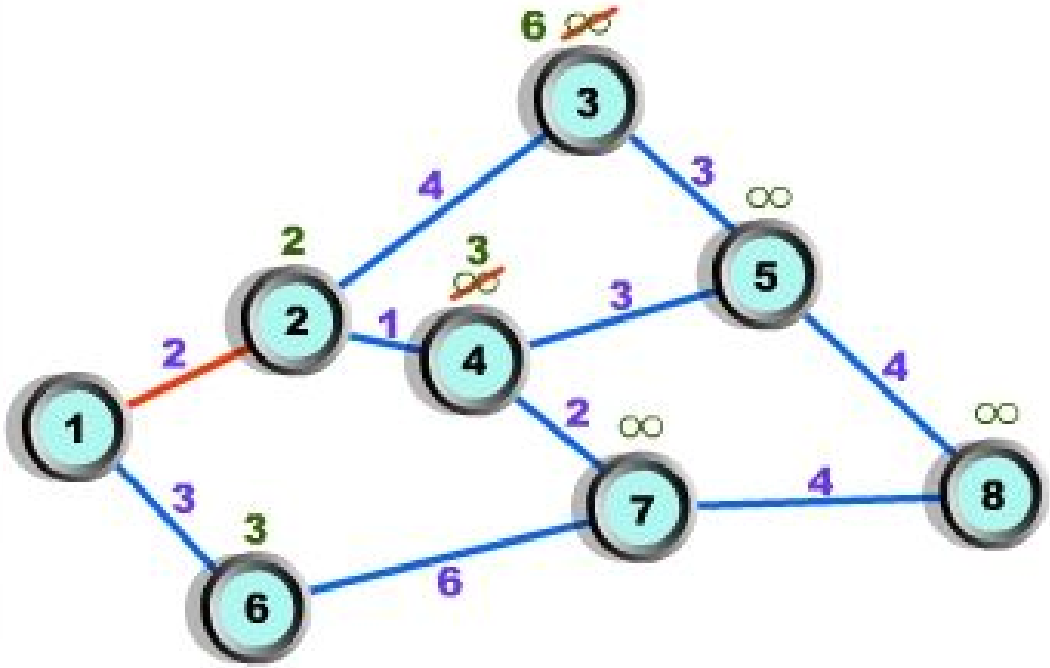
∞ 값은 연결되지 않은 부분을 나타낸 것으로 실제로는 충분히 큰 값을 넣는다.

1. 처음에 출발node를 선택하여 각 node까지의 임시최단거리를 표시, 직접 연결되는 edge가 없으면 ∞로 표시한다.

2. 선택되지 않은 node에 대하여, 가장 작은 임시최단거리를 갖는 node를 선택하고 연결 하여 영구최단거리로 삼는다. 도착node가 선택되면 끝내고, 아니면 3단계로 간다.

3. 선택되지 않은 node에 대해, 직전에 선택된 node와 연결될 때의 거리가 기존의 임시최단거리보다 작으면 임시최단거리를 수정하여 2단계로 간다.

다익스트라(Dijkstra) 알고리즘 – 동작원리(2)



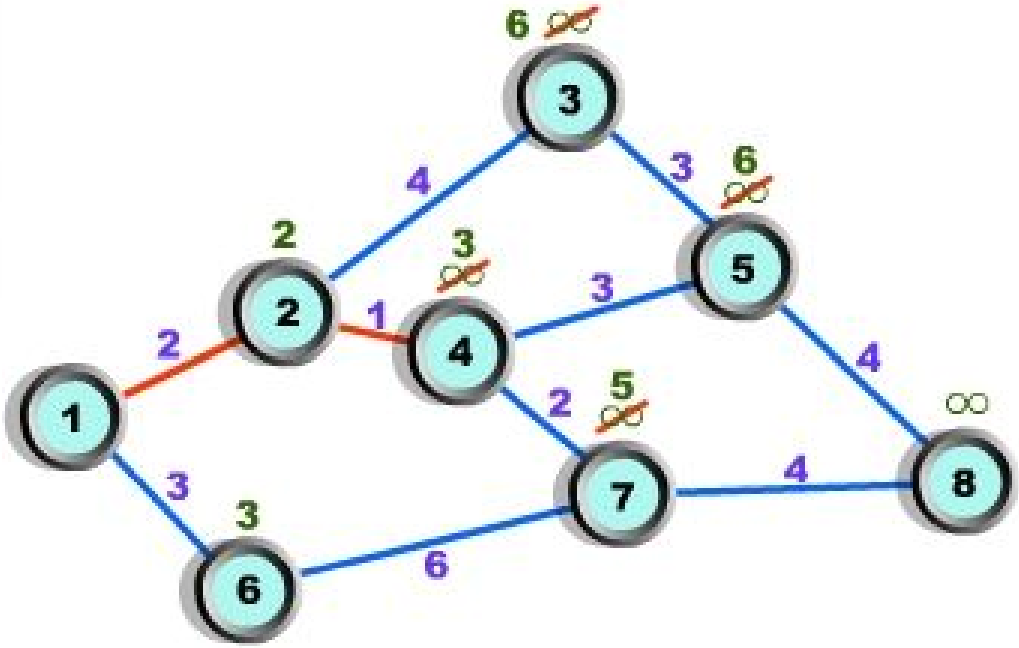
1 단계 – 경로 1에서 시작한다.

경로 2나 경로 6으로 갈 수 있는데 가장 가까운 경로인 2로 간다. 가중치는 2이다.
경로 2로 갔으므로 다음에 경로 3이나 경로 4로 갈 수 있다.

따라서 무한대였던 경로 3과 경로 4의 임시최단거리를 수정해야 한다.

경로 3의 임시최단거리는 무한대에서 2 + 4 = 6 으로 수정
경로 4의 임시최단거리는 무한대에서 2 + 1 = 3 으로 수정

다익스트라(Dijkstra) 알고리즘 – 동작원리(3)

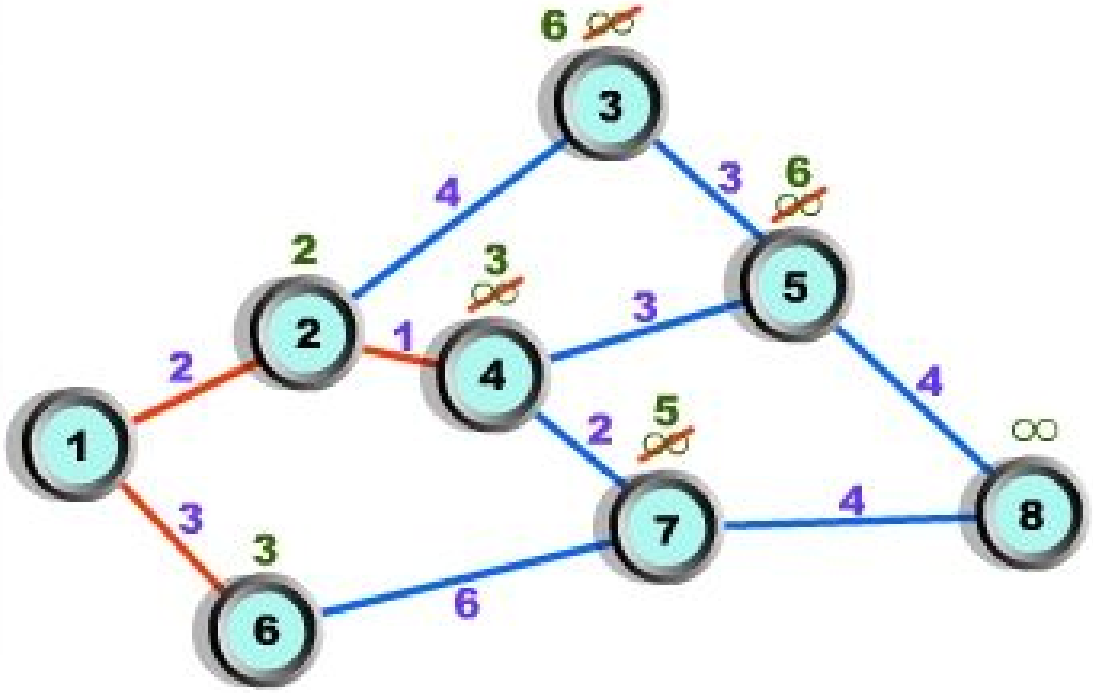


2 단계 – 경로 4와 경로 6의 가중치 값은 3으로, 가장 작다.

먼저 경로 4로 가게 되면 경로 5와 7의 임시최단거리가 수정된다.

경로 5의 임시최단거리는 무한대에서 3 + 3 = 6 으로 수정
경로 7의 임시최단거리는 무한대에서 3 + 2 = 5 로 수정

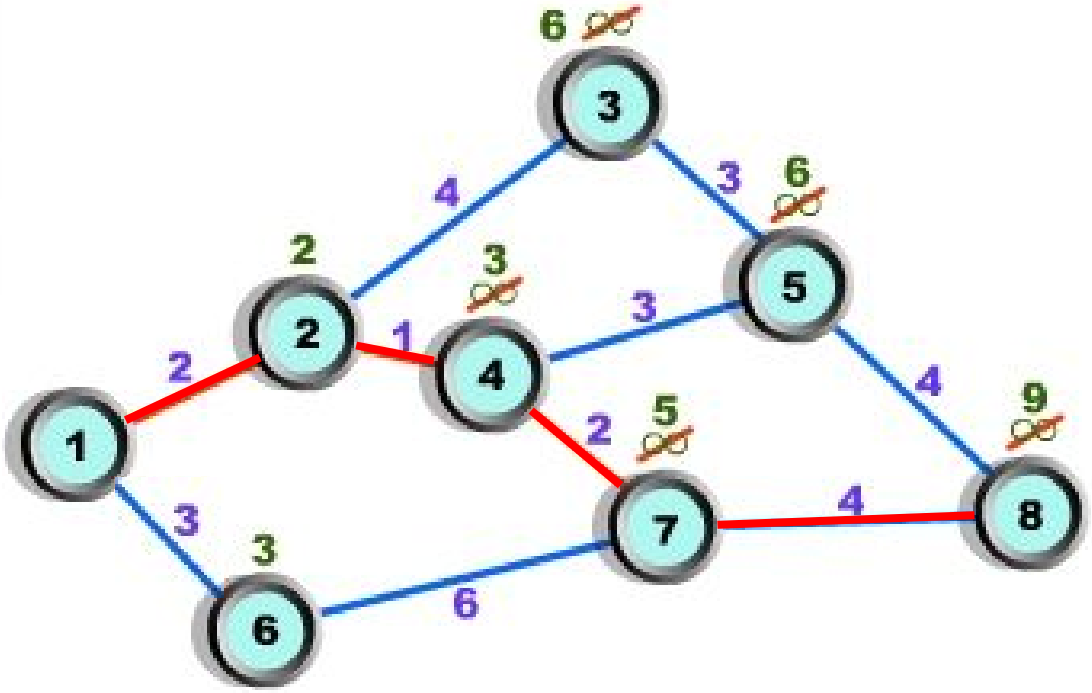
다익스트라(Dijkstra) 알고리즘 – 동작원리(4)



3 단계 – 경로 6으로 간다.

경로 6에서도 경로 7로 갈 수 있지만 임시최단거리는 수정하지 않는다.
(현재 경로 7의 가중치값은 5인데 반해, 경로 1에서 경로 6을 거쳐 경로 7로 가게되면 그 거리는 3 + 6 = 9 가 되기 때문)
값이 더 크므로 경로 7의 임시최단거리에는 아무런 영향을 미치지 못한다.

다익스트라(Dijkstra) 알고리즘 – 동작원리(5)



4단계 – 경로 7의 임시최단거리가 5 이므로 경로 7로 간다.

경로 8의 임시최단거리를 5 + 4 = 9로 수정

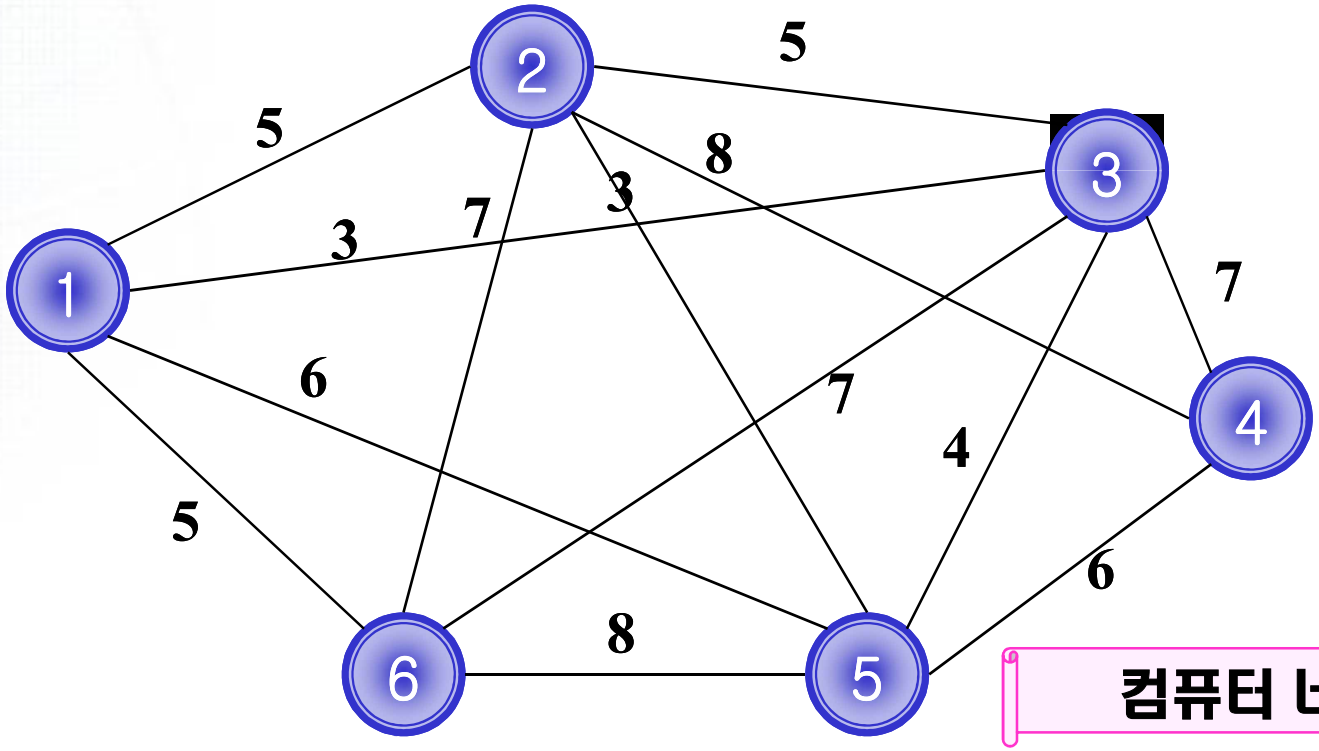
경로 3, 5에서 각각 최단거리를 비교해 보면 그 위치에서의 가중치 값이 더 크므로 임시최단거리는 수정되지 않는다.
경로 1 – 2 – 4 – 7 – 8 로 갈 때 가중치가 9로 최소로 확정 된다.

최소 비용 문제(최소 걸침 나무 문제) - 그리디(Greedy) 해법

"박재흥, 최신경영과학, 시그마그래프, 2004, p.172 ~ p.175"

네트워크상의 모든 node를 연결하되, 연결된 총 길이를 최소로 하는 문제 – 수송시스템이나 컴퓨터 네트워크의 설계에 주로 이용

예제 모형 : 6개 지역에 분산되어 있는 컴퓨터를 네트워크로 연결하는 문제



컴퓨터 네트워크

최소 비용 문제(최소 걸침 나무 문제) - 그리디(Greedy) 해법

"박재흥, 최신경영과학, 시그마그래프, 2004, p.172 ~ p.175"

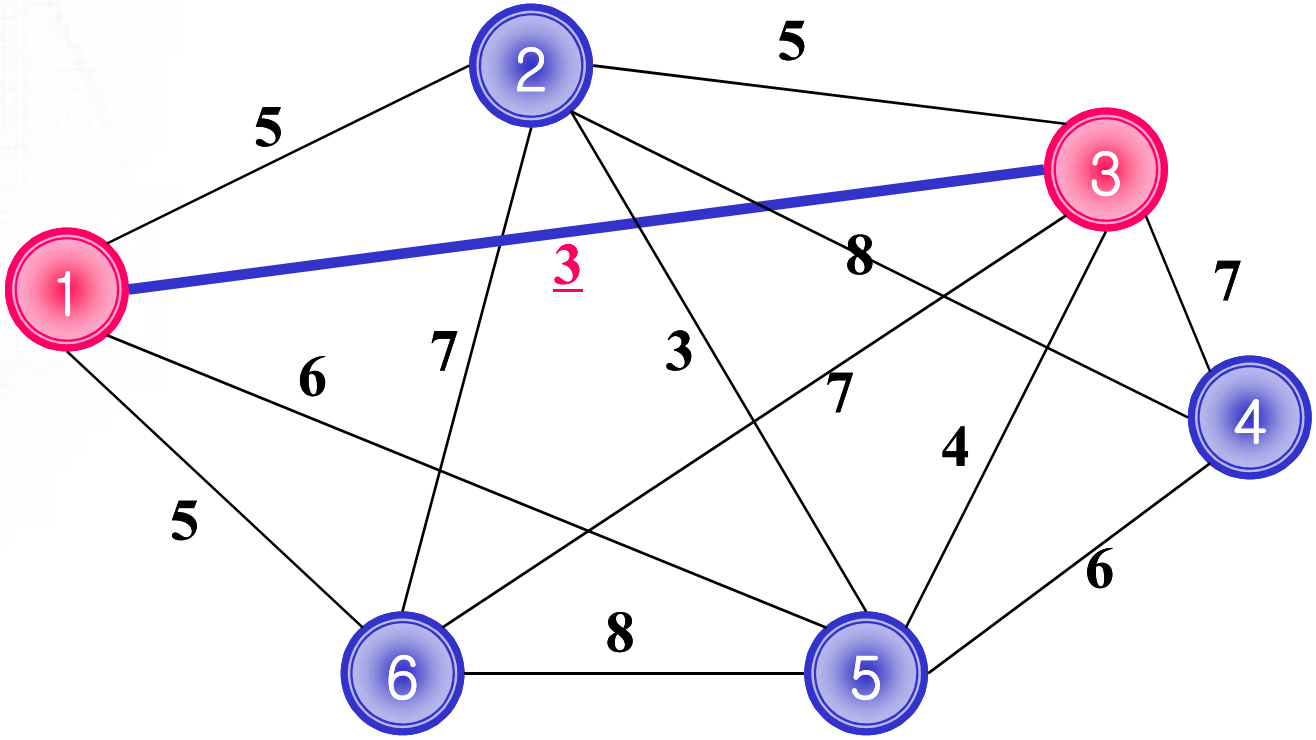
그리디 해법(greedy algorithm)
- 최소걸침나무문제의 대표적 해법

1단계 임의의 node에서 출발하여, 그 node와 가장 가까운 node를 선택하고 연결한다.

2단계 선택되지 않은 node들에 중에서, 선택된 node들과의 거리가 가장 짧은 node를 선택하고 이를 연결한다. 모든 node가 선택될 때 까지 이를 반복한다.

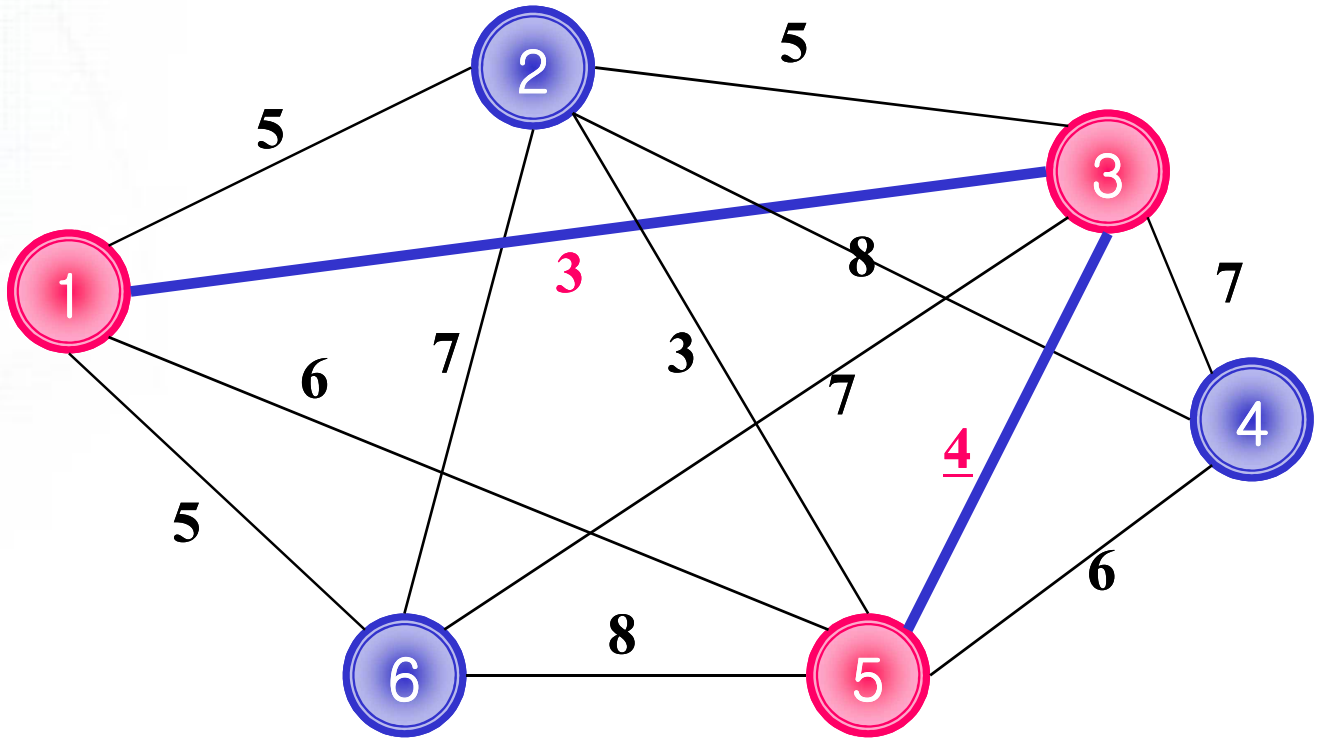
최소 비용 문제(최소 걸침 나무 문제) - 그리디(Greedy) 해법

첫 번째 연결 : 1번 node에서 출발, 가장 가까운 node가 3번이므로 선택하여 연결



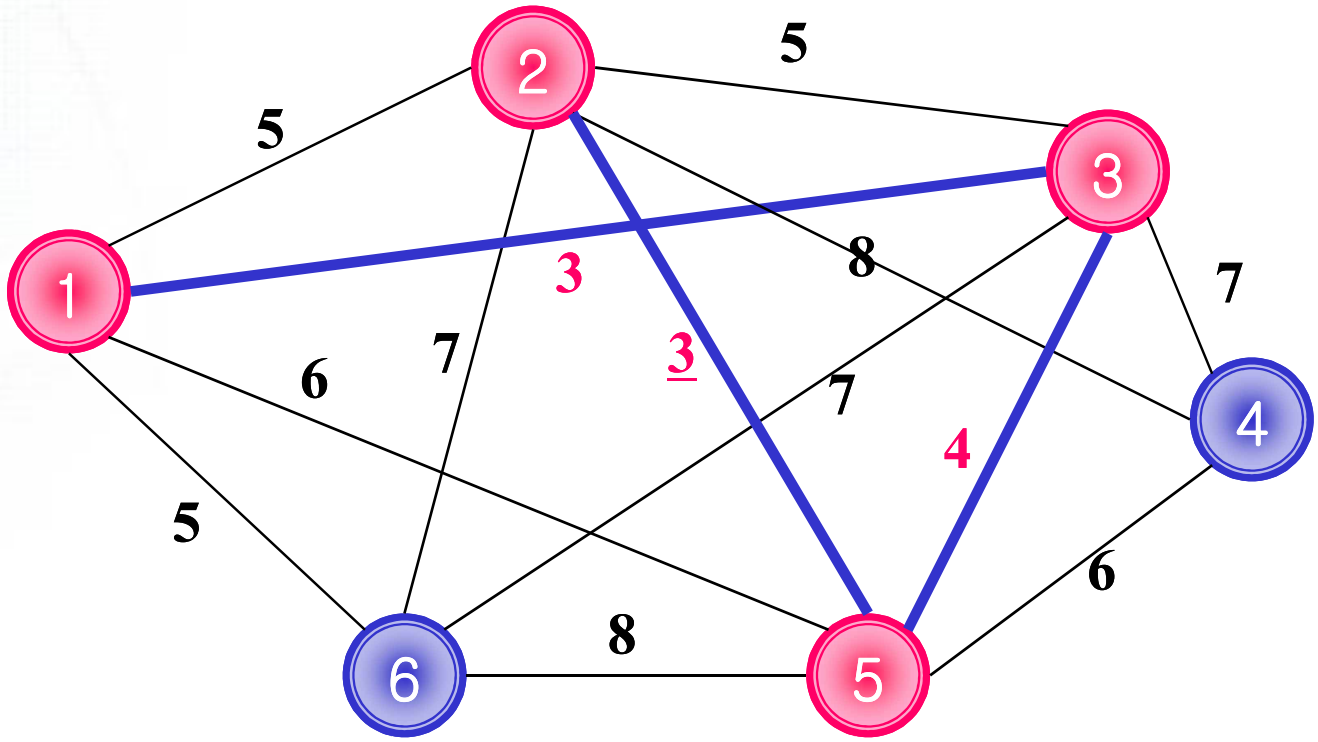
최소 비용 문제(최소 걸침 나무 문제)
- 그리디(Greedy) 해법

두 번째 연결 : 선택된 node ①, ③번에서 가장 가까운 거리에 있는 선택되지 않은 node가 ⑤번이므로 선택하고 이를 ③번 node와 연결



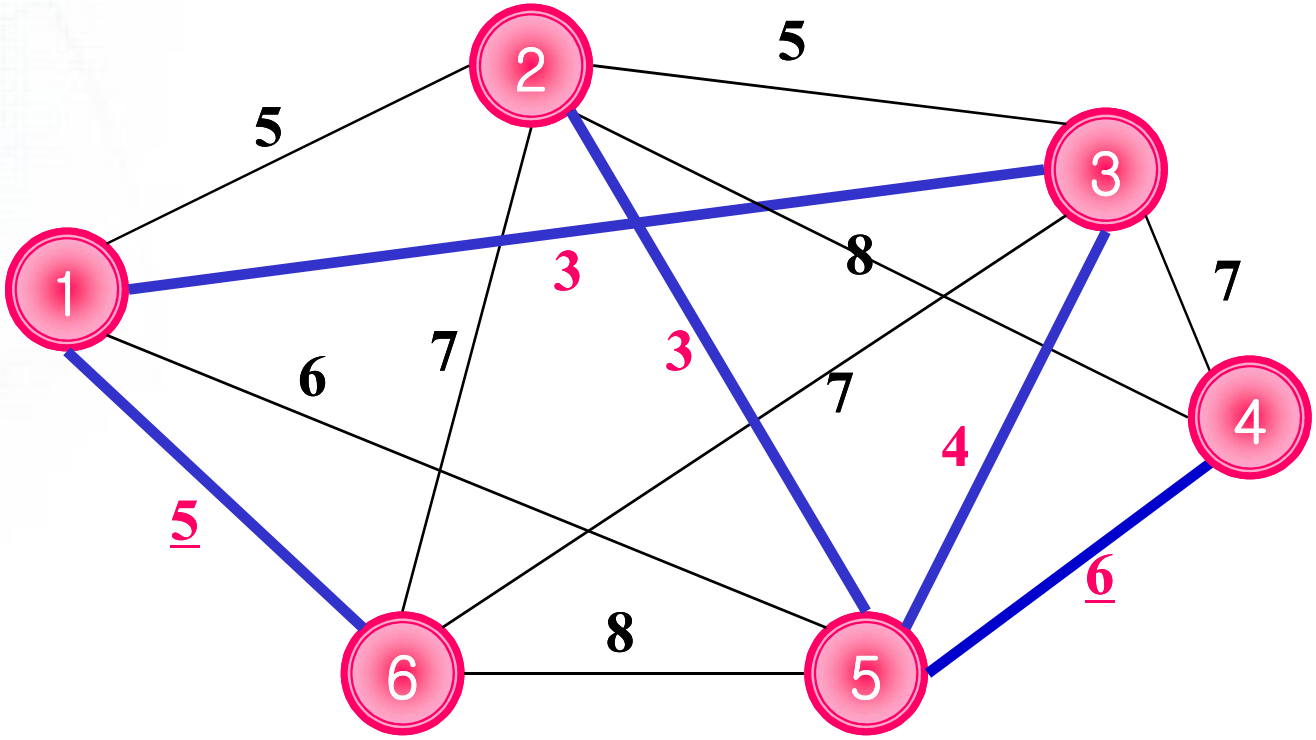
최소 비용 문제(최소 걸침 나무 문제)
- 그리디(Greedy) 해법

세 번째 연결 : 선택된 node ①, ③, ⑤번에서 아직 선택되지 않은 ②, ④, ⑥ 번 node로 연결되는 경로중 가장 작은 거리를 갖는 node ②를 선택



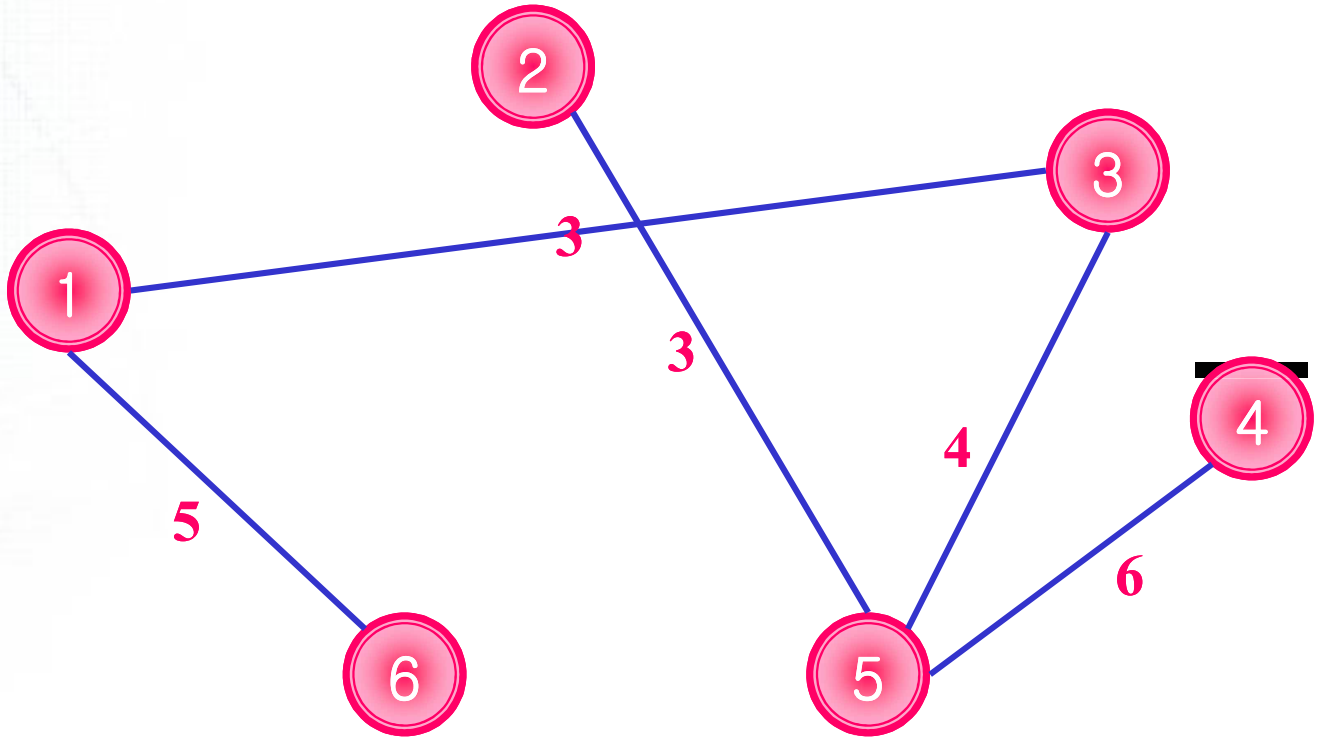
최소 비용 문제(최소 걸침 나무 문제)
- 그리디(Greedy) 해법

네 번째, 다섯 번째 연결 : 마찬가지 방법으로 ⑥번, ④번 node 선택



최소 비용 문제(최소 걸침 나무 문제)
- 그리디(Greedy) 해법

최종적으로 연결된 네트워크 : 총 거리는 3 + 3 + 4 + 5 + 6 = 21

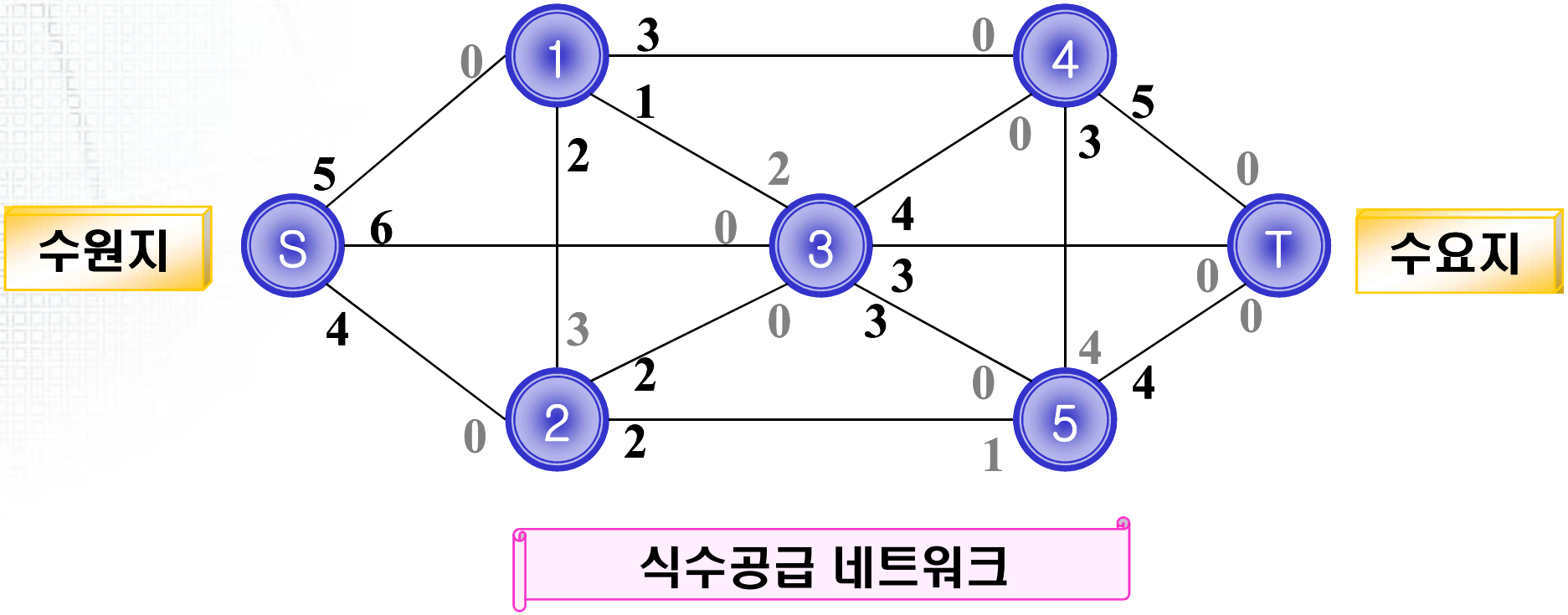


n개의 node가 주어지면 항상 n-1개의 edge로 연결되는 해를 갖는다.
최적 연결은 시작하는 node가 어느 node인가에 관계가 없다.

최소 비용 문제(최소 걸침 나무 문제)
- 그리디(Greedy) 해법

각 edge에 흐를 수 있는 용량이 한정되어 있을 때 흘려 보낼 수 있는 최대의 유통량을 구하는 문제
흐름의 예 : 원유, 식수, 가스 등의 유동체나 교통량, 정보량, 통신량 등

예제 모형 : T 지역의 식수공급 문제



식수공급 네트워크

최대 흐름 문제

최대흐름문제의 해법

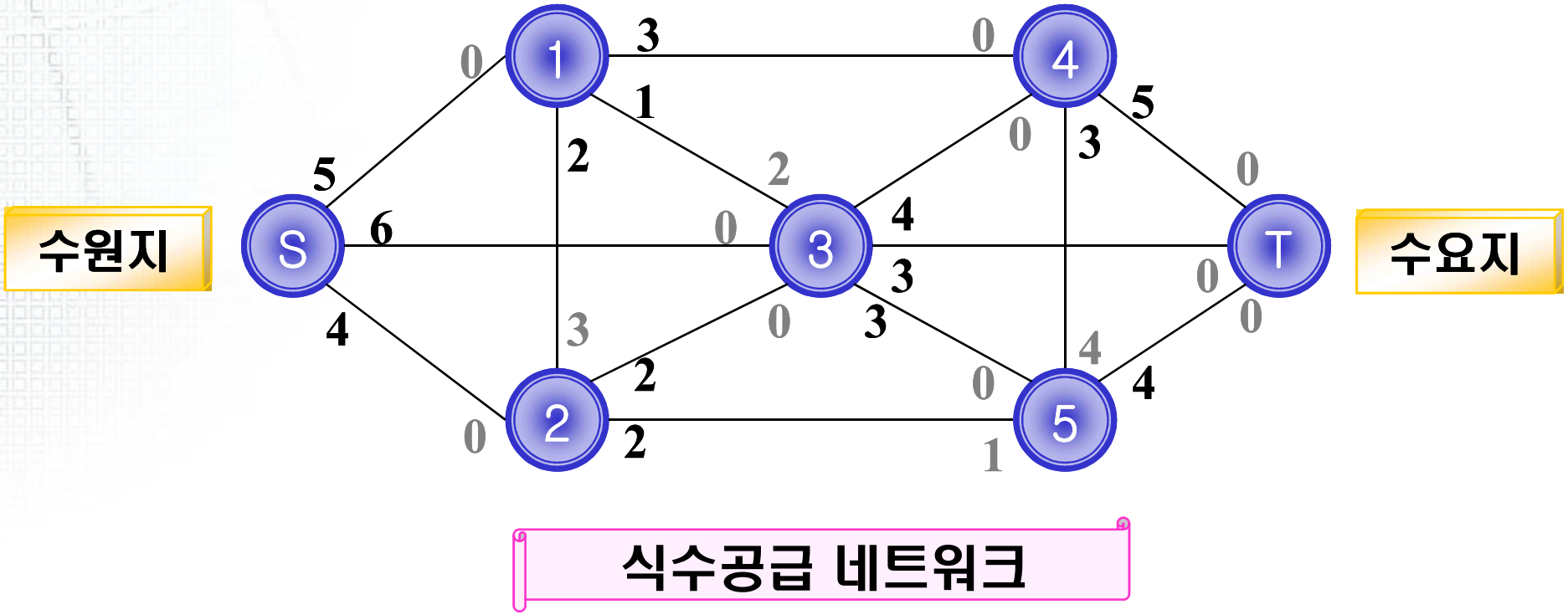
1단계 공급지에서 수요지로 양의 용량을 갖는 경로를 선택한다. 이러한 경로를 선택할 수 없으면, 현재의 흐름량이 최대이다.

2단계 선택한 경로에 포함된 edge의 용량중 최소값을 그 경로의 흐름량으로 배정한다.

3단계 각 edge의 용량에 대해, 위에서 결정된 흐름량을 순방향으로는 빼주고 역방향으로는 더해준 다음 1단계로 간다.

최대 흐름 문제

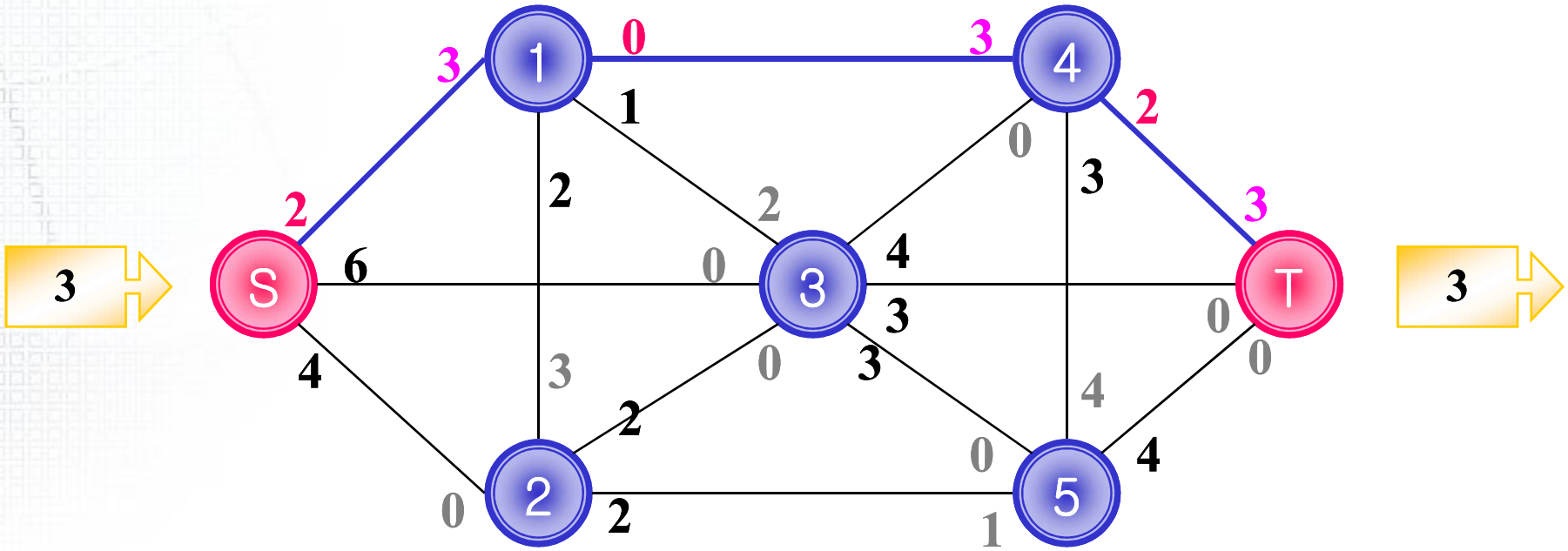
"박재흥, 최신경영과학, 시그마그래프, 2004, p.175 ~ p.179"



식수공급 네트워크

최대 흐름 문제

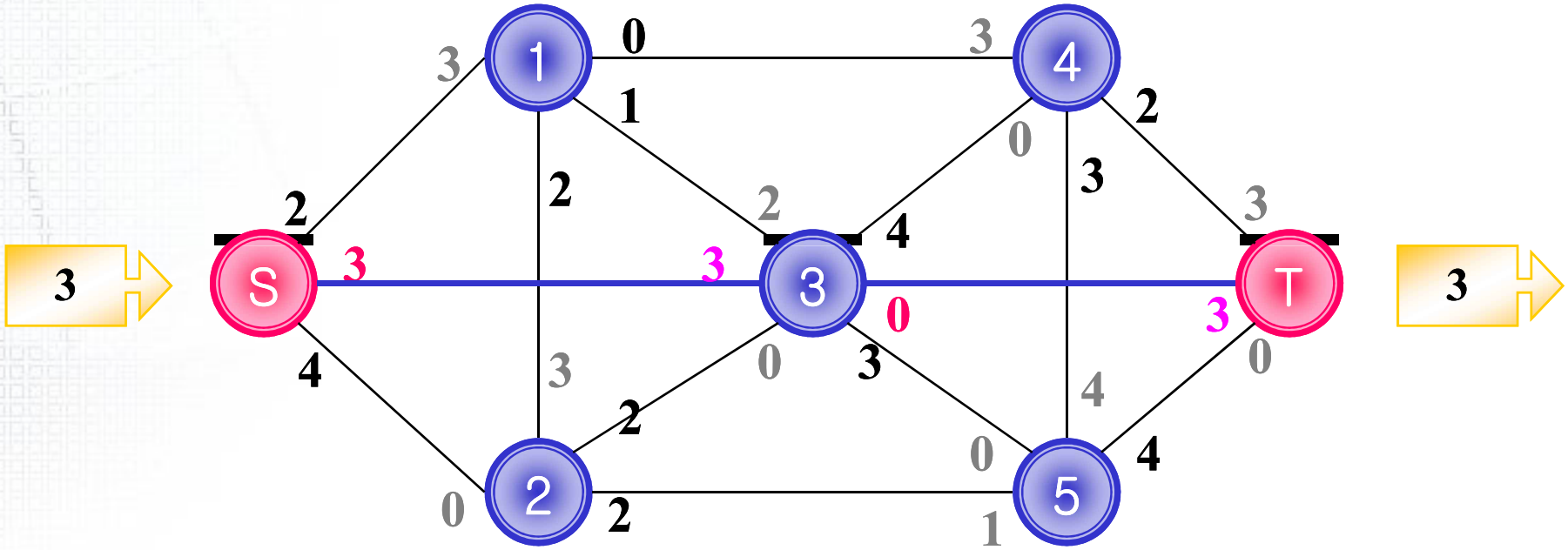
첫 번째 배정
- 양의 흐름용량을 갖는 경로 선택 (S→①→④→T 경로)
- 각 edge의 용량이 5, 3, 5이므로 3을 배정
- 흐름량 3을 순방향의 용량에서는 빼주고 역방향의 용량에는 더해준다.



첫 번째 배정 후의 용량 변화

최대 흐름 문제

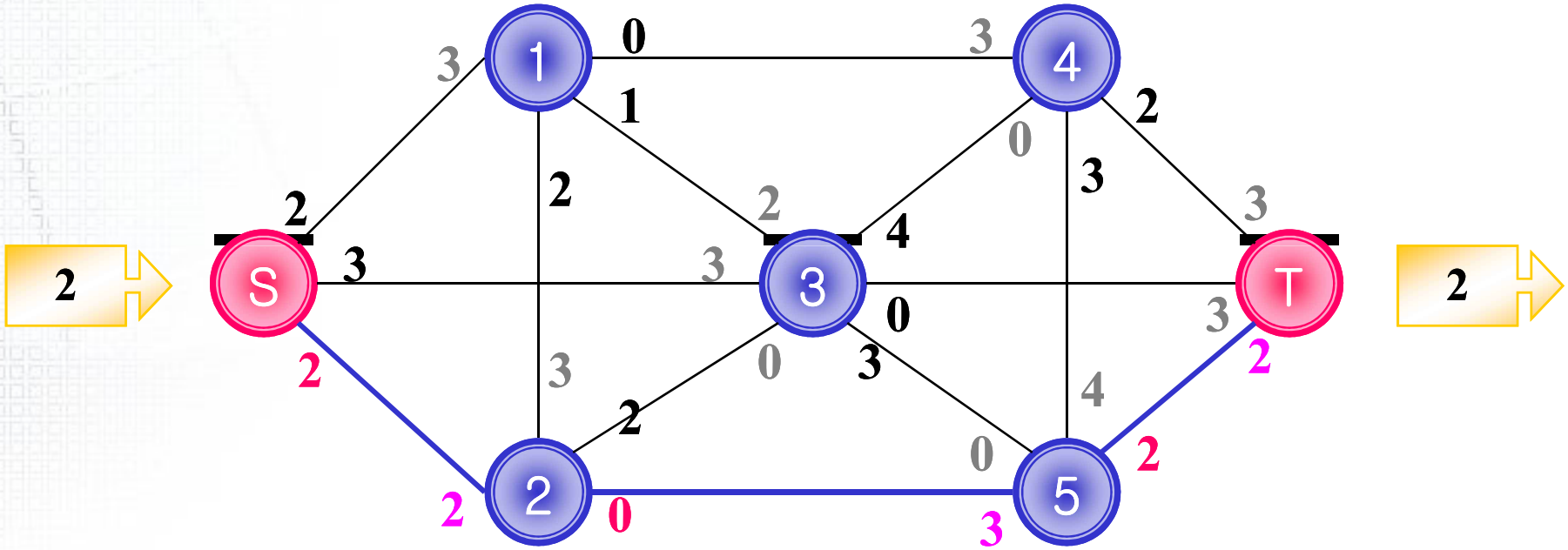
두 번째 배정 : 경로 S→③→T를 선택, 흐름량 min{6, 3} = 3을 배정



두 번째 배정 후의 용량 변화

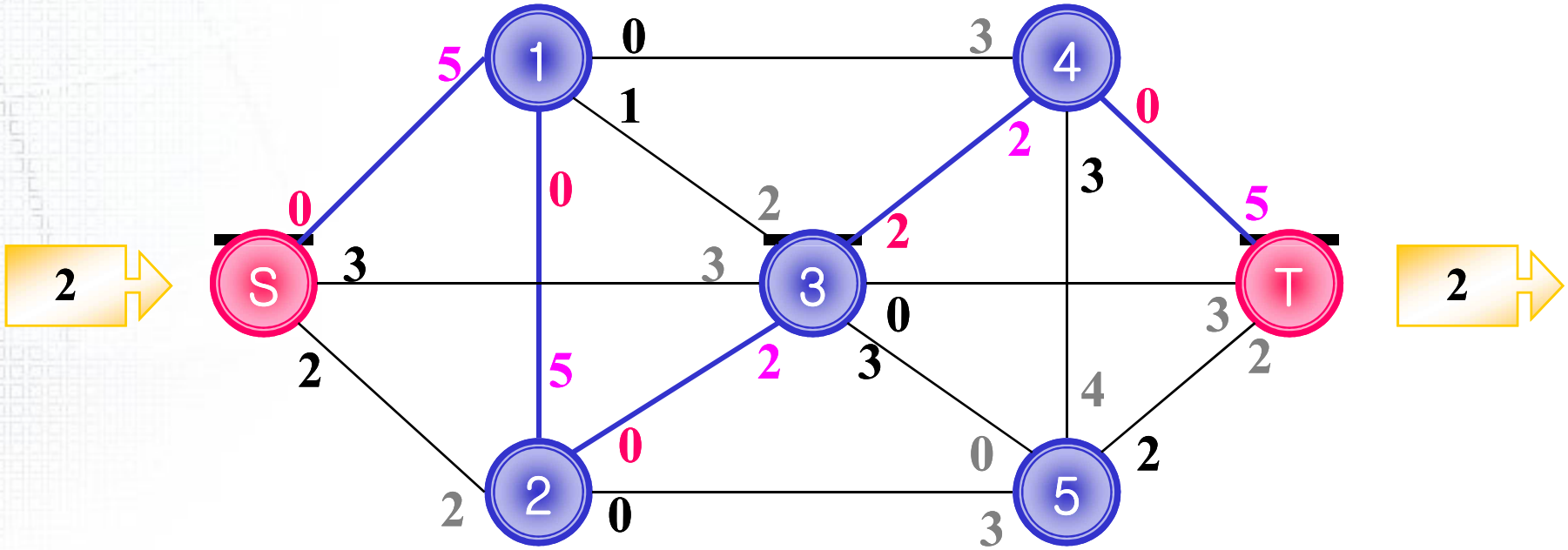
최대 흐름 문제

세 번째 배정 : S→②→⑤→T 경로에 2 배정



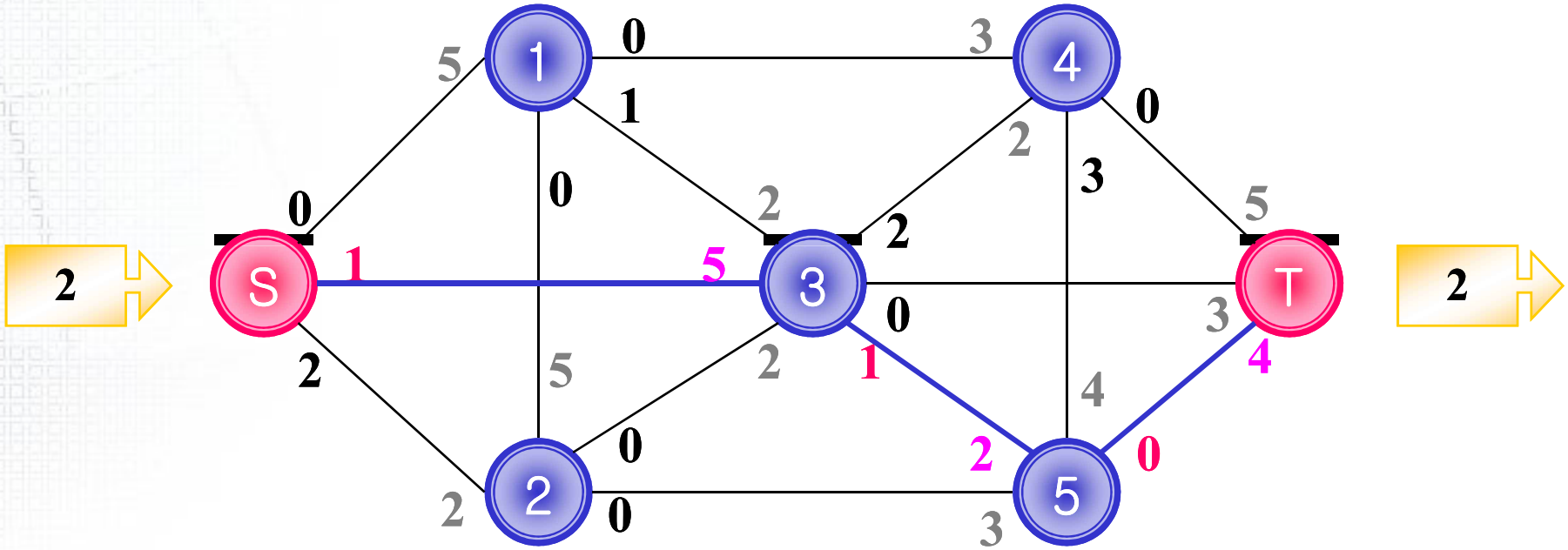
최대 흐름 문제

네 번째 배정 : S→①→②→③→④→T 경로에 2 배정



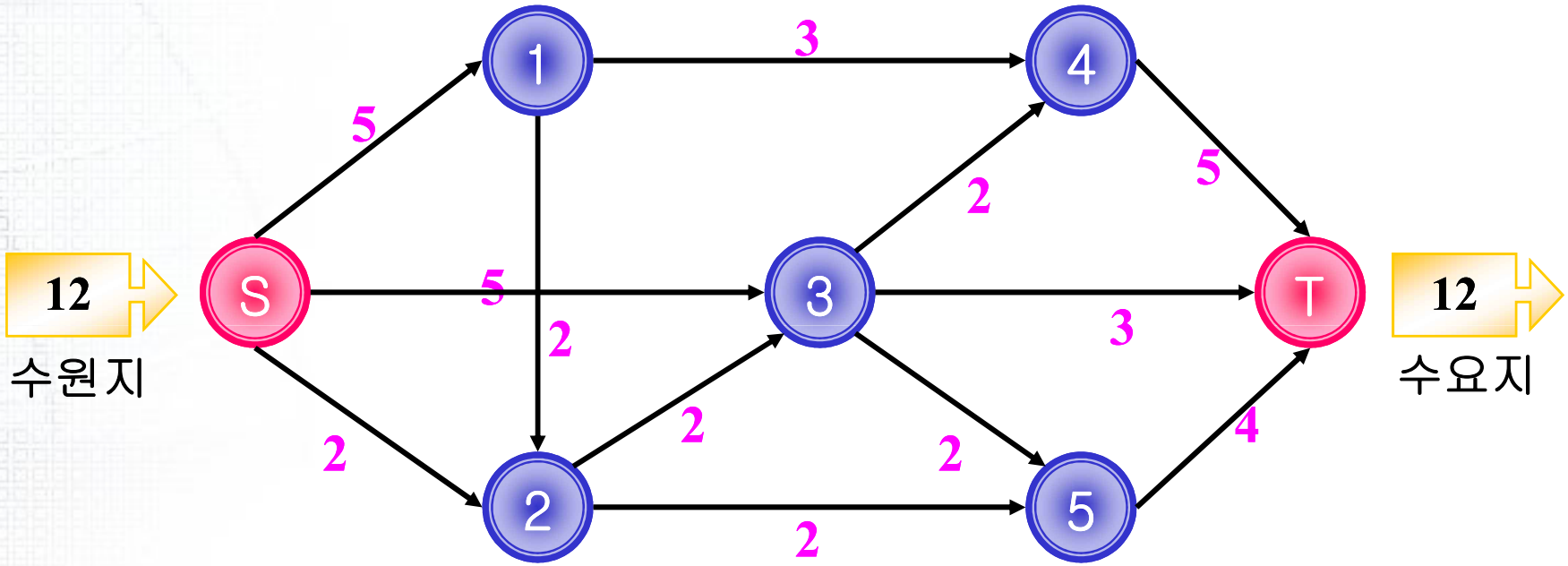
최대 흐름 문제

다섯 번째 배정 : S→③→⑤→T 경로에 2 배정



최대 흐름 문제

최종 배정 결과



Integer Programming(정수 계획법)
- 헝가리법[1](Hungarian method)

Advanced
Ship
Design
Automation
Laboratory

[1]헝가리법(Hungarian method) : 헝가리 수학자 쾨히니에 의하여 제안된 배치 테이블을 이용하여 할당 문제를 해결하는 기법

정수 계획법의 분류

"Stanley Zionts, Linear and integer programming, Prentice-Hall, 1974, p.336"
"Gerard Sierksma, Linear and integer programming, Marcel Dekker, Inc., 1996"
"박순달, 경영과학, 4정판, 민영사, 2003."
"박재흥, 최신경영과학, 시그마그래프, 2004."
"Laurence A. Wolsey, Integer programming, A Wiley-Interscience Publication, 1998"

수학적(Mathematical) 최적화 방법

Cut Algorithm(절단 평면법)

주어진 선형 계획 문제에 정수해는 만족하고, 정수가 아닌 해는 만족하지 않는 제약조건을 추가하면서 문제를 푸는 방법

Enumeration Algorithm(열거법)

주어진 선형 계획 문제의 가능해 영역(feasible region)내의 정수해를 열거하여 최적해를 찾는 방법

예)분지 한계법(분단 탐색법) - Branch and Bound Method

열거법의 일종으로서, 상한(upper bound)와 하한(lower bound)라는 개념을 사용하여 가능한한 가능해를 적게 열거하여 최적해를 찾는 방법

Constructive Algorithm

선형 계획 문제를 특성에 따라 분류하고 그 특성에 맞는 최적 해법을 적용하는 것

문제의 분류	네트워크 이론(Network theory)			할당 문제	...
	최단 경로 문제	최소 비용 문제	최대 흐름 문제		
해법	다익스트라(Dijkstra) 해법	그리디(Greedy) 해법	최대 흐름 문제 해법	헝가리법(Hungarian method)	...

휴리스틱(Heuristic) 최적화 방법

비록 최적해는 구하지 못하지만 최적해에 가까운 좋은 해를 짧은 시간 내에(즉, 적은 계산량으로) 찾으려는 방법

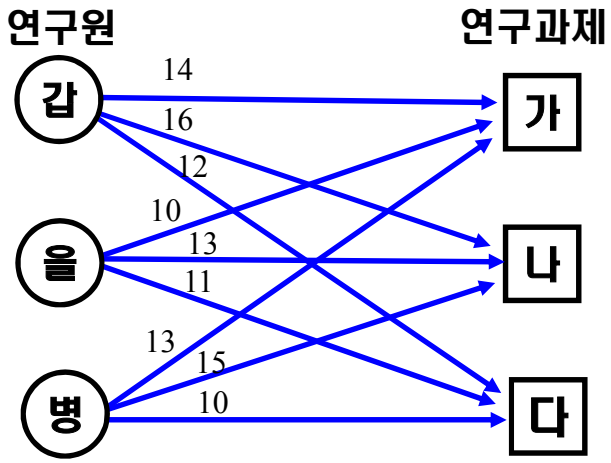
개미 알고리즘	유전 알고리즘	최근거리 인접점 (nearest-neighbor) 알고리즘	교점교환 알고리즘	k-optimal 알고리즘	...

"박재흥, 최신경영과학, 시그마그래프, 2004, p.151 ~ p.156"

할당문제(Assignment Problem)
- 헝가리법(Hungarian method)[1)]

어떤 연구소에서 앞으로 수행해야 할 연구과제가 3건(가, 나, 다)있는데, 이 과제들을 3명의 책임연구원(갑, 을, 병)에게 하나씩 할당하려고 한다.
각 과제의 연구원별 예상수행기간은 표와 같다.

연구원	연구과제		
	가	나	다
갑	14	16	12
을	10	13	11
병	13	15	10



Minimize

$$F = 14x_{11} + 16x_{12} + 12x_{13} + 10x_{21} + 13x_{22} + 11x_{23} + 13x_{31} + 15x_{32} + 10x_{33}$$

s.t.

$$\begin{aligned}
x_{11} + x_{12} + x_{13} &= 1 \\
x_{21} + x_{22} + x_{23} &= 1 \\
x_{31} + x_{32} + x_{33} &= 1 \\
x_{11} + x_{21} + x_{31} &= 1 \\
x_{12} + x_{22} + x_{32} &= 1 \\
x_{13} + x_{23} + x_{33} &= 1 \\
x_{ij} &\geq 0 \ (i = 1,2,3; j = 1,2,3)
\end{aligned}$$

설계변수 $x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}$

[1)]헝가리법(Hungarian Method) : 헝가리 수학자 쾨히니에 의하여 제안된 할당 문제를 해결하는 기법.

할당문제(Assignment Problem)
- 헝가리법(Hungarian method)

연구원	연구과제		
	가	나	다
갑	14	16	12
을	10	13	11
병	13	15	10

Minimize

$$F = 14x_{11} + 16x_{12} + 12x_{13} + 10x_{21} + 13x_{22} + 11x_{23} + 13x_{31} + 15x_{32} + 10x_{33}$$

s.t.

$$\begin{aligned}
x_{11} + x_{12} + x_{13} &= 1 \\
x_{21} + x_{22} + x_{23} &= 1 \\
x_{31} + x_{32} + x_{33} &= 1 \\
x_{11} + x_{21} + x_{31} &= 1 \\
x_{12} + x_{22} + x_{32} &= 1 \\
x_{13} + x_{23} + x_{33} &= 1 \\
x_{ij} &\geq 0 \ (i = 1,2,3;\ j = 1,2,3)
\end{aligned}$$

설계변수 $x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}$

각 열의 가장 작은 비용 선택하여 해당열의 비용에서 차감

연구원	연구과제		
	가	나	다
갑	14	16	12
을	(10)	(13)	11
병	13	15	(10)

0이 없는 행은 그 행의 가장 작은 비용을 해당 행의 모든 비용에서 차감

연구원	연구과제		
	가	나	다
갑	4	3	(2)
을	0	0	1
병	3	2	0

할당문제(Assignment Problem)
- 헝가리법(Hungarian method)

연구원	연구과제		
	가	나	다
갑	14	16	12
을	10	13	11
병	13	15	10

Minimize

$$F = 14x_{11} + 16x_{12} + 12x_{13} + 10x_{21} + 13x_{22} + 11x_{23} + 13x_{31} + 15x_{32} + 10x_{33}$$

s.t.

$$\begin{aligned}
x_{11} + x_{12} + x_{13} &= 1 \\
x_{21} + x_{22} + x_{23} &= 1 \\
x_{31} + x_{32} + x_{33} &= 1 \\
x_{11} + x_{21} + x_{31} &= 1 \\
x_{12} + x_{22} + x_{32} &= 1 \\
x_{13} + x_{23} + x_{33} &= 1 \\
x_{ij} &\geq 0 \ (i = 1,2,3;\ j = 1,2,3)
\end{aligned}$$

설계변수 $x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{33}$

가장 적은 수의 직선으로 모든 0을 지움

직선 수와 행의 수보다 적으므로 할당이 되지 않음
직선으로 지워지지 않은 가장 작은 비용만큼 지워지지 않은 모든 비용에서 차감, 직선으로 두번 지워지는 비용에는 더함

연구원	연구과제		
	가	나	다
갑	2	1	0
을	0	0	1
병	3	2	0

가장 적은 수의 직선으로 모든 0을 지움

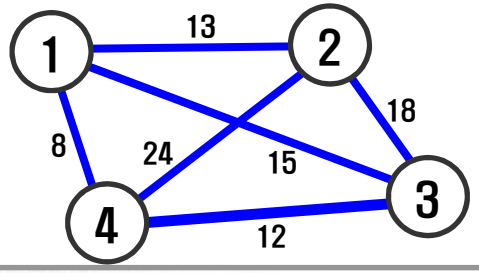
직선 수와 행의 수가 같으므로 할당 할 수 있음

연구원	연구과제		
	가	나	다
갑	1	0	0
을	0	0	2
병	2	1	0

※'병→다' , '갑→나' , '을→가' 연구과제 할당 됨

외판원문제 수학적 최적화 모델 정식화

외판원 문제[1)]



목적함수

Minimize 외판원의 총 이동거리 최소화

$$\begin{aligned} F = \; & 13x_{12} + 15x_{13} + 8x_{14} \\ & + 13x_{21} + 18x_{23} + 24x_{24} \\ & + 15x_{31} + 18x_{32} + 12x_{34} \\ & + 8x_{41} + 24x_{42} + 12x_{43} \end{aligned}$$

$$F = \sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}x_{ij}$$
$$(i, j = 1, 2, \cdots N)$$

설계변수

$$x_{12}, x_{13}, x_{14}, x_{21}, x_{23}, x_{24}, x_{31}, x_{32}, x_{34}, x_{41}, x_{42}, x_{43}$$

제약조건

x_{ij} : 노드 i에서 노드 j로 이동하면 1, 그렇지않으면 0

$x_{ij} = 0$ 또는 1

외판원은 각 노드에서 출발하여 단 한 노드에만 갈 수 있다.

$$x_{12}+x_{13}+x_{14}=1 \qquad x_{21}+x_{23}+x_{24}=1$$
$$x_{31}+x_{32}+x_{34}=1 \qquad x_{41}+x_{42}+x_{43}=1$$

외판원은 도착하는 노드를 기준으로 볼 때 단 한 노드에서부터 출발하여 온다.

$$x_{21}+x_{31}+x_{41}=1 \qquad x_{12}+x_{32}+x_{42}=1$$
$$x_{13}+x_{23}+x_{43}=1 \qquad x_{14}+x_{24}+x_{34}=1$$

외판원은 모든 노드를 방문하여 처음 출발지로 돌아와야 한다.

$$x_{12}+x_{21}\le 1 \qquad x_{13}+x_{31}\le 1 \qquad x_{12}+x_{23}+x_{31}\le 2 \qquad x_{12}+x_{24}+x_{41}\le 2$$
$$x_{23}+x_{32}\le 1 \qquad x_{24}+x_{42}\le 1 \qquad x_{13}+x_{32}+x_{21}\le 2 \qquad x_{13}+x_{34}+x_{41}\le 2$$
$$x_{14}+x_{41}\le 1 \qquad x_{34}+x_{43}\le 1 \qquad x_{14}+x_{42}+x_{21}\le 2 \qquad x_{14}+x_{43}+x_{31}\le 2$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad \sum_{i=1}^{N} x_{ij} = 1$$

i노드에서 단 한번만 출발 j노드에 단 한번만 도착

x_{ij}는 하나의 Hamiltion순환로[2)]를 형성

[1)] **외판원 문제** (TSP : Traveling Salesman Problem) : 외판원 문제는 한 명의 외판원이 최단시간에 주어진 노드(고객)들을 정확하게 한번씩 방문하고 다시 출발지점으로 돌아오는 외판원 경로를 찾는 문제이다.

[2)] Hamilton 순환로 : 모든 노드를 정확히 한번만 지나서 처음 출발한 노드로 가는 path.

외판원문제
- 헝가리법(Hungarian method)

각 열의 거리를 그 열의 가장 짧은 거리로 차감

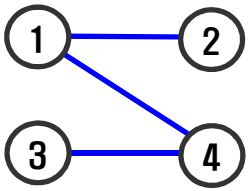
	1	2	3	4
1	∞	13	15	8
2	13	∞	18	24
3	15	18	∞	12
4	8	24	12	∞

0이 없는 행은 그 행의 가장 짧은 거리를 해당 행의 모든 거리에서 차감

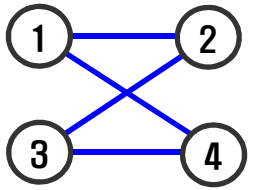
	1	2	3	4
1	∞	0	3	0
2	5	∞	6	16
3	7	5	∞	4
4	0	11	0	∞

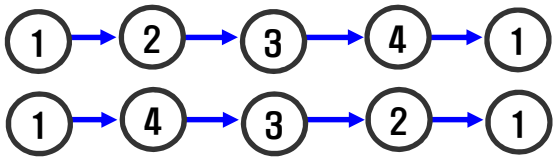
가장 적은 수의 직선으로 모든 0을 지움
순서결정이 불가능하면 지워지지 않은 가장 짧은 거리만큼 지워지지 않은 모든 거리에서 차감

	1	2	3	4
1	∞	0	3	0
2	0	∞	1	11
3	3	1	∞	0
4	0	11	0	∞



	1	2	3	4
1	∞	0	3	0
2	0	∞	0	11
3	3	0	∞	0
4	0	11	0	∞





8+12+18+13=51

13+18+12+8=51

Integer Programming(정수 계획법)

- Heuristic Algorithm(휴리스틱 알고리즘)

Advanced
Ship
Design
Automation
Laboratory

정수 계획법의 분류

"Stanley Zionts, Linear and integer programming, Prentice-Hall, 1974, p.336"
"Gerard Sierksma, Linear and integer programming, Marcel Dekker, Inc., 1996"
"박순달, 경영과학, 4정판, 민영사, 2003."
"박재흥, 최신경영과학, 시그마그래프, 2004."
"Laurence A. Wolsey, Integer programming, A Wiley-Interscience Publication, 1998"

수학적(Mathematical) 최적화 방법

Cut Algorithm(절단 평면법)

주어진 선형 계획 문제에 정수해는 만족하고, 정수가 아닌 해는 만족하지 않는 제약조건을 추가하면서 문제를 푸는 방법

Enumeration Algorithm(열거법)

주어진 선형 계획 문제의 가능해 영역(feasible region)내의 정수해를 열거하여 최적해를 찾는 방법

예)분단 탐색법(분지 한계법) - Branch and Bound Method

열거법의 일종으로서, 상한(upper bound)와 하한(lower bound)라는 개념을 사용하여 가능한한 가능해를 적게 열거하여 최적해를 찾는 방법

Constructive Algorithm

선형 계획 문제를 특성에 따라 분류하고 그 특성에 맞는 최적 해법을 적용하는 것

문제의 분류	네트워크 이론(Network theory)			할당 문제	●●●
	최단 경로 문제	최소 비용 문제	최대 흐름 문제		
해법	다익스트라(Dijkstra) 해법	그리디(Greedy) 해법	최대 흐름 문제 해법	헝가리법(Hungarian method)	●●●

휴리스틱(Heuristic) 최적화 방법

비록 최적해는 구하지 못하지만 최적해에 가까운 좋은 해를 짧은 시간 내에(즉, 적은 계산량으로) 찾으려는 방법

유전 알고리즘 | 최근거리 인접점(nearest-neighbor) 알고리즘 | 교점교환 알고리즘 | k-optimal 알고리즘 | 개미 알고리즘 | ●●●

외판원 문제를 위한 휴리스틱(heuristic) 알고리즘

- 정수계획법의 경우에는 변수의 수와 제약조건의 수가 많아지면 선형계획법을 이용하여 문제의 해를 구하기 어렵다.

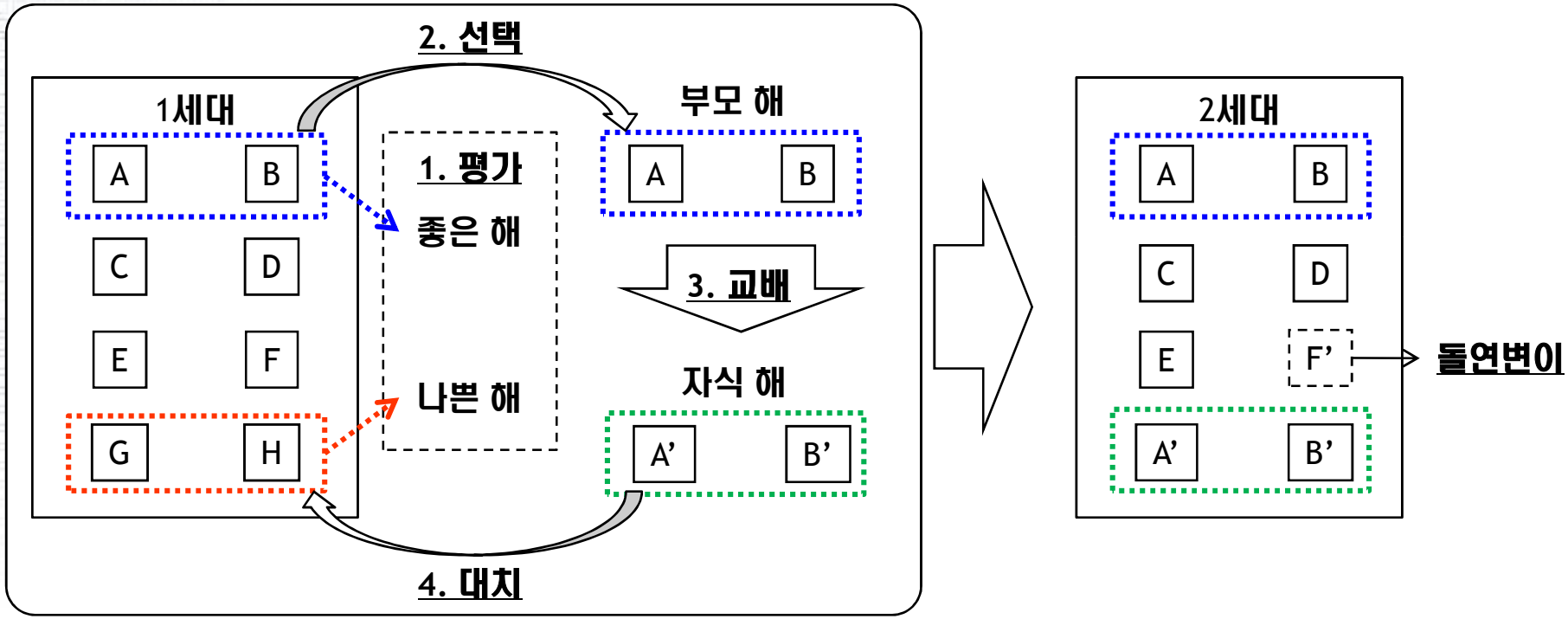
- 한 예로 분단 탐색법을 이용하여 해를 구할 때 최악의 경우 모든 경우를 다 고려하여 계산해야 한다.
  변수와 제약조건의 수가 많아지면 계산시간이 지수적으로 증가하여 해를 구하기 어렵게 된다.

- 휴리스틱(heuristic)알고리즘은 비록 최적해는 구하지 못하지만 최적해에 가까운 좋은 해를 짧은 시간 내에(즉, 적은 계산량으로) 찾으려는 방법이다.
  - 최근거리인접점(nearest-neighbor) 알고리즘
  - k-optimal 알고리즘 (2-optimal, 3-optimal)
  - 교점교환 알고리즘
  - 유전 알고리즘
  - 개미 알고리즘

"문병로, 유전알고리즘, 다성출판사, 2001."

유전 알고리즘 설명

- 생물의 진화 메커니즘을 이용하여 다양한 문제를 해결 하고자 하는 방법이다.
- 적자 생존과 자연 도태의 진화 원리를 도입하여 초기해로부터 선택(selection) 연산, 교배(crossover) 연산, 돌연변이(mutation) 연산과 같은 일련의 과정을 반복하여 해를 구한다.



1. **평가:** 1세대의 각 해 품질을 평가한다.
2. **선택(Selection):** 품질이 좋은 해를 부모 해로 선택 한다.
3. **교배(Crossover):** 부모 해로부터 자식 해를 생성한다.
4. **대치:** 1세대 내의 나쁜 해를 자식 해로 교체한다.

돌연변이(Mutation): 부모 해에 없는 속성을 도입하여 새로운 해를 만드는 연산이다.

평가, 선택, 교배, 대치의 순서를 반복하며 최적 해를 찾는다.

유전 알고리즘 예시(1/5)

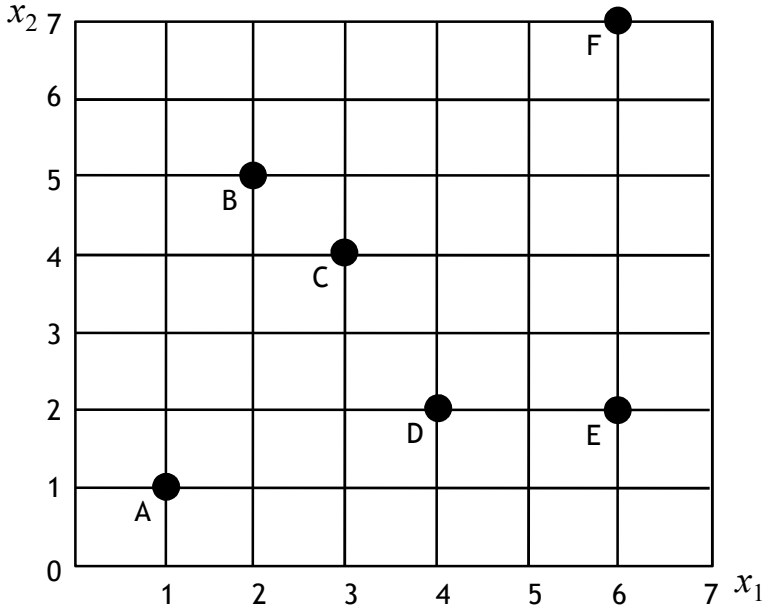
Minimize $f(\mathbf{x}) = x_1^2 + x_2^2 - 8x_1 - 8x_2$

	1세대	평가
A	(1 , 1)	-14
B	(2 , 5)	-27
C	(3 , 4)	-31
D	(4 , 2)	-28
E	(6 , 2)	-24
F	(6 , 7)	-19

부모 해

(3 , 4)

(4 , 2)



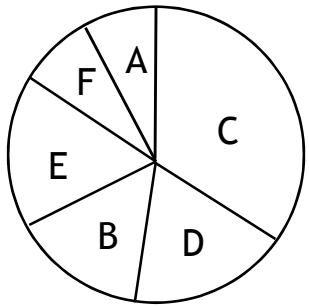
1. 평가: 1세대의 각 해 품질을 평가한다.

2. 선택(Selection): 품질이 좋은 해를 부모해로 선택 한다.

- 본 예제에서는 품질이 가장 좋은 2개의 해를 부모 해로 선택 하였다.
- 일반적으로는 확률적으로 해를 선택하며, 품질이 좋은 해가 선택될 확률을 높인다.

예) 품질 비례 룰렛 휠 선택

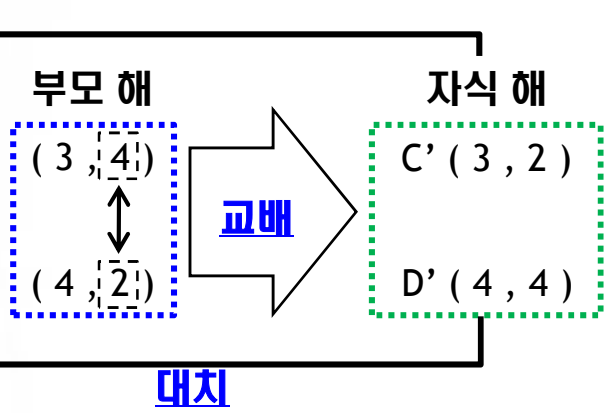
각 해의 품질에 비례하여
룰렛 휠에서 차지하는 넓이를
배정함으로서, 품질에 비례하여
선택 받을 확률을 높이는 방법

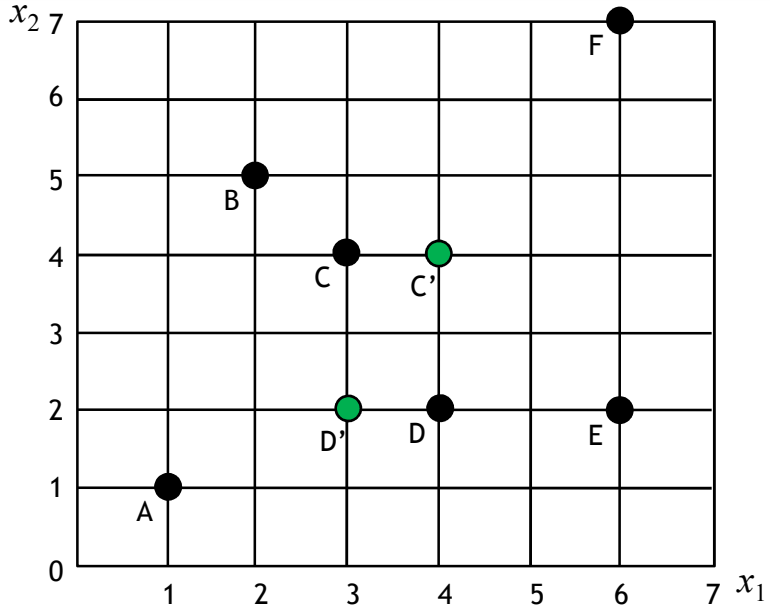


유전 알고리즘 예시(2/5)

Minimize $f(\mathbf{x}) = x_1^2 + x_2^2 - 8x_1 - 8x_2$

	1세대	평가
A	(1 , 1)	-14
B	(2 , 5)	-27
C	(3 , 4)	-31
D	(4 , 2)	-28
E	(6 , 2)	-24
F	(6 , 7)	-19





3. 교배(Crossover): 부모 해로부터 자식 해를 생성한다.

- 본 예제에서는 x_2를 서로 교환하는 방법을 사용 하였음

일반적인 교배(Crossover) 방법 설명

4. 대치: 생성된 자식 해를 평가가 좋지 않은 해와 교환한다.

- 본 예제에서는 품질이 가장 나쁜 2개의 해를 자식 해로 대치 하였다.

해를 대치하는 다른 방법의 예

+ 예시 1)자식 해의 품질을 즉시 평가하여, 부모 해보다 품질이 좋으면 부모 해와 대치, 그렇지 않으면 대치하지 않음
+ 예시 2)자식 해의 품질을 즉시 평가하여, 부모 해보다 품질이 좋으면 부모 해와 대치, 그렇지 않으면 품질이 나쁜 해와 대치

유전 알고리즘 예시(3/5)

Minimize $f(\mathbf{x}) = x_1^2 + x_2^2 - 8x_1 - 8x_2$

	2세대	평가
A	(3 , 2)	-27
B	(2 , 5)	-27
C	(3 , 4)	-31
D	(4 , 2)	-28
E	(6 , 2)	-24
F	(4 , 4)	-32

대치

돌연변이

부모 해

(3 , 4)

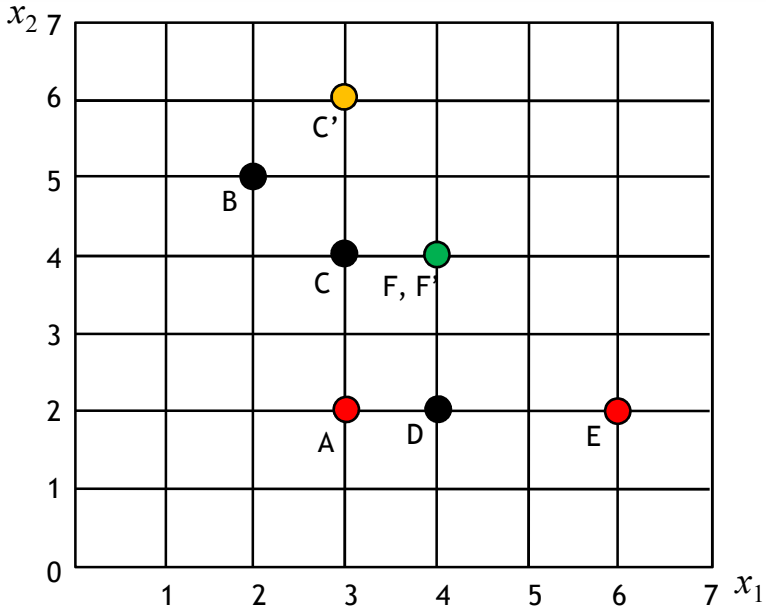
(4 , 4)

교배

자식 해

C' (3 , 6)

F' (4 , 4)



	3세대	평가
A	(3 , 6)	
B	(2 , 5)	
C	(3 , 4)	
D	(4 , 2)	
E	(4 , 4)	
F	(4 , 4)	

다시 평가, 선택, 교배, 대치의 순서를 반복하며 최적 해를 찾는다.

돌연변이: 부모 해에 없는 속성을 도입하여 새로운 해를 만드는 연산이다.

종료조건:

- 루프를 일정 횟수만큼 반복한 후 정지시키는 방법
  + 어느 정도 반복 횟수로서 해가 수렴할 것이라는 경험적 지식이 있어야 함
- 해 집합 내의 해들이 대부분(예: 70%) 똑같은지를 판단하여 종료

유전 알고리즘 예시(4/5)

Minimize $f(\mathbf{x}) = x_1^2 + x_2^2 - 8x_1 - 8x_2$

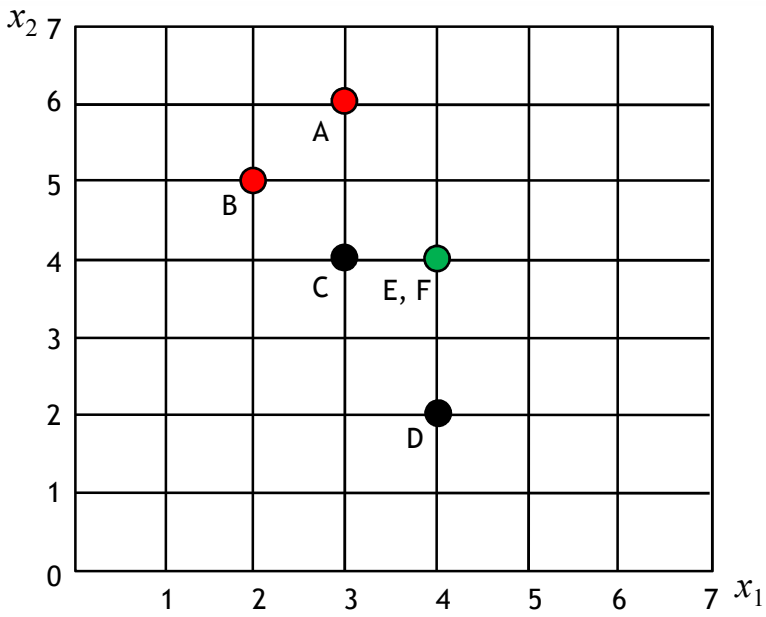
	3세대	평가
A	(3 , 6)	-27
B	(2 , 5)	-27
C	(3 , 4)	-31
D	(4 , 2)	-28
E	(4 , 4)	-32
F	(4 , 4)	-32

대치

부모 해: (4 , 4) ↕ (4 , 4)

교배

자식 해: E' (4 , 4), F' (4 , 4)



	4세대	평가
A	(4 , 4)	
B	(4 , 4)	
C	(3 , 4)	
D	(4 , 2)	
E	(4 , 4)	
F	(4 , 4)	

다시 평가, 선택, 교배, 대치의 순서를 반복하며 최적 해를 찾는다.

종료조건:

- 루프를 일정 횟수만큼 반복한 후 정지시키는 방법
  + 어느 정도 반복 횟수로서 해가 수렴할 것이라는 경험적 지식이 있어야 함
- 해 집합 내의 해들이 대부분(예: 70%) 똑같은지를 판단하여 종료

유전 알고리즘 예시(5/5)

Minimize $f(\mathbf{x}) = x_1^2 + x_2^2 - 8x_1 - 8x_2$

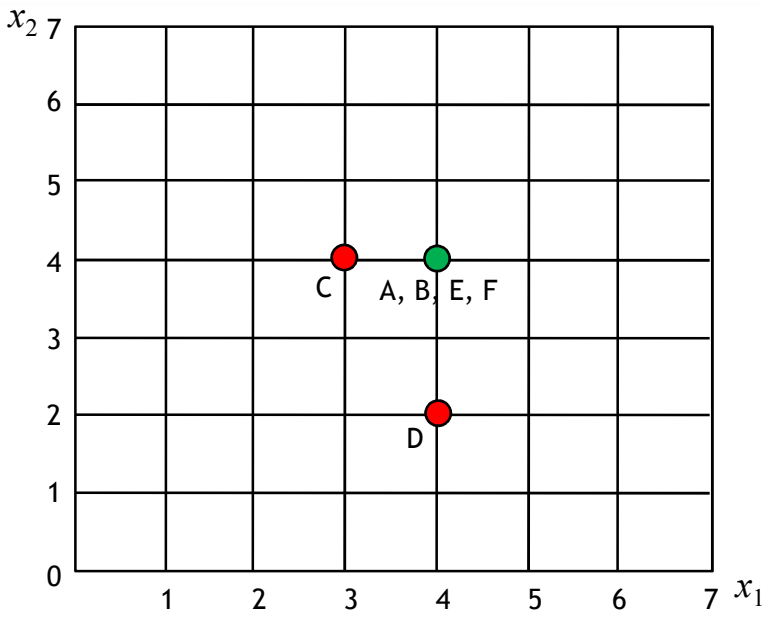
	4세대	평가
A	(4 , 4)	-32
B	(4 , 4)	-32
C	(3 , 4)	-31
D	(4 , 2)	-28
E	(4 , 4)	-32
F	(4 , 4)	-32

대치

부모 해: (4 , 4) / (4 , 4)

교배

자식 해: E' (4 , 4) / F' (4 , 4)



	5세대	평가
A	(4 , 4)	
B	(4 , 4)	
C	(4 , 4)	
D	(4 , 4)	
E	(4 , 4)	
F	(4 , 4)	

종료 → 최적점은 (4,4)

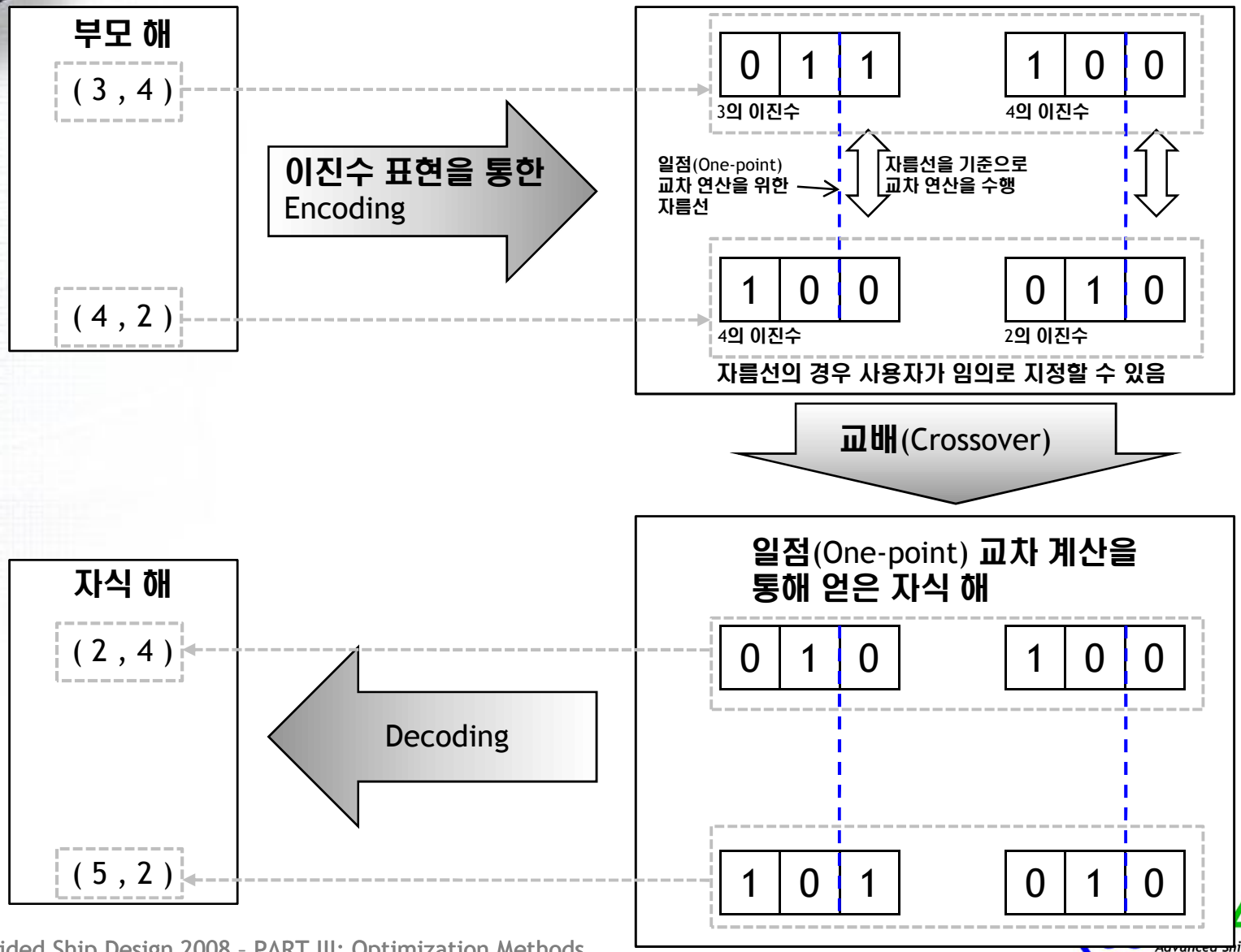
다시 평가, 선택, 교배, 대치의 순서를 반복하며 최적 해를 찾는다.

종료조건:

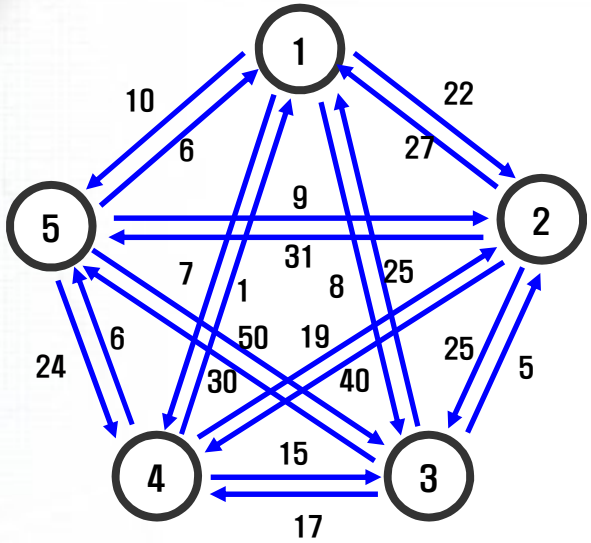
- 루프를 일정 횟수만큼 반복한 후 정지시키는 방법
  + 어느 정도 반복 횟수로서 해가 수렴할 것이라는 경험적 지식이 있어야 함
- 해 집합 내의 해들이 대부분(예: 70%) 똑같은지를 판단하여 종료

유전 알고리즘

- 이진수 표현과 일점(One-point) 교차 연산을 이용한 교배(Crossover)방법의 예시



외판원 문제
-최근거리인접점(nearest-neighbor) 알고리즘



	1	2	3	4	5
1	∞	22	8	7	10
2	27	∞	25	40	31
3	25	5	∞	17	30
4	1	19	15	∞	6
5	6	9	50	24	∞

노드1에서 시작 : 1 → 4 → 5 → 2 → 3 → 1, 총 거리 : 72

노드2에서 시작 : 2 → 3 → 4 → 1 → 5 → 2, 총 거리 : 62

노드3에서 시작 : 3 → 2 → 1 → 4 → 5 → 3, 총 거리 : 95

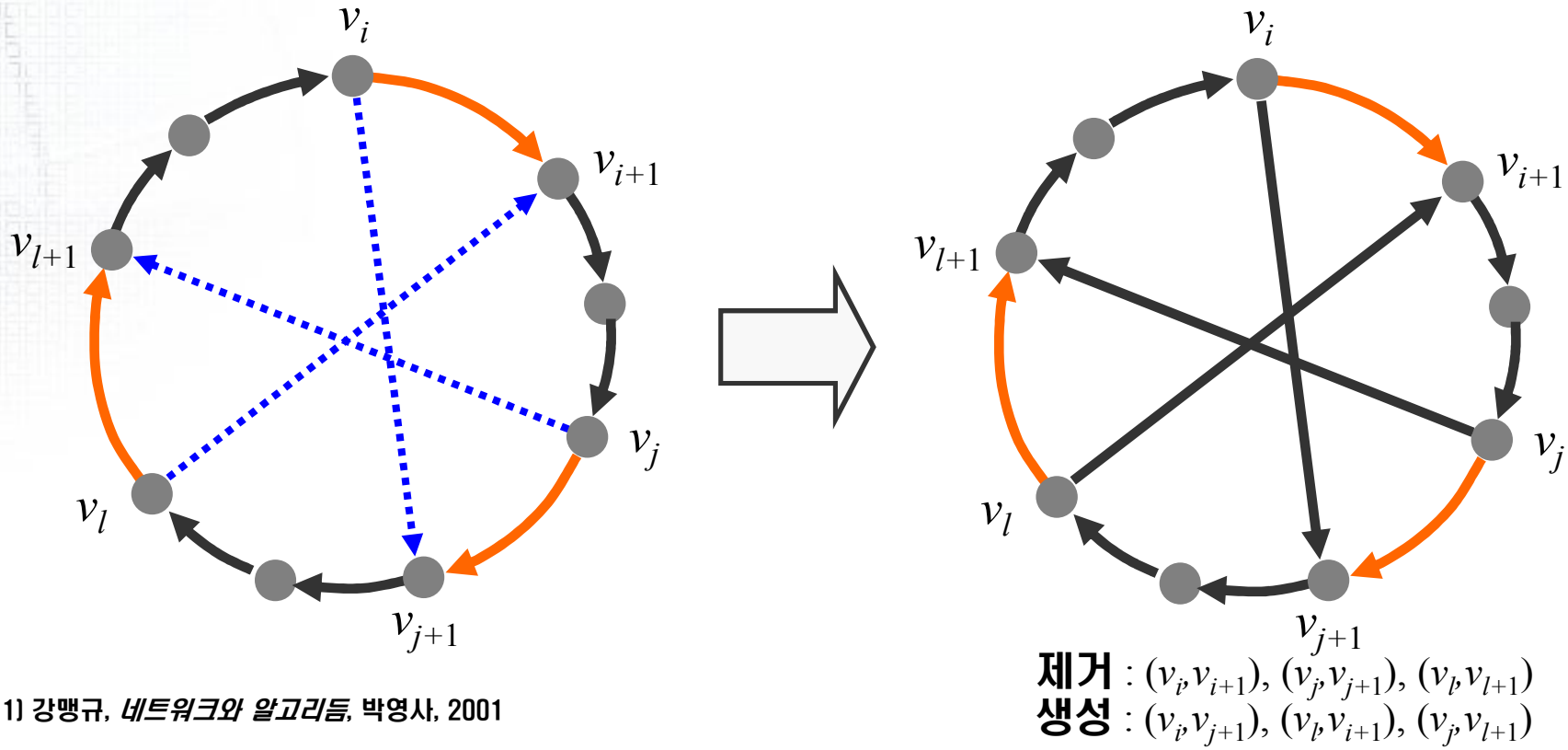
노드4에서 시작 : 4 → 1 → 3 → 2 → 5 → 4, 총 거리 : 69

노드5에서 시작 : 5 → 1 → 4 → 3 → 2 → 5, 총 거리 : 64

1. 한 노드에서 시작하여 그 노드에서 가장 가까운 노드를 방문한다.
2. 방문한 노드에서, 아직 방문되지 않은 노드 중에서 가장 가까운 노드를 방문한다.
3. 모든 노드가 방문될 때까지 방문하고, 마지막으로 방문한 노드와 첫 노드를 연결한다.

외판원 문제
-k-optimal 알고리즘

- 외판원 순환로에서 k개의 edge를 제거하고, 순환로에서 없었던 k개의 edge를 대체하여 새로운 외판원 순환로를 만든다.
- 새로운 외판원 순환로가 기존 해 보다 개선(총 이동 거리가 감소) 되었다면, 이를 초기해로 하여 더 개선된 순환로를 구한다.
- 더 이상 개선된 순환로를 찾을 수 없을 때까지 이 과정을 반복 수행
- k값에 따라 k=2, k=3일 때 그 해를 각각 2-optimal, 3-optimal이라 한다. 3-optimal이 해의 개선과 계산량 면에서 가장 효율적이므로 많이 사용된다.[1)]



제거 : (v_i, v_{i+1}), (v_j, v_{j+1}), (v_l, v_{l+1})
생성 : (v_i, v_{j+1}), (v_l, v_{i+1}), (v_j, v_{l+1})

1) 강맹규, *네트워크와 알고리듬*, 박영사, 2001

외판원 문제
-k-optimal 알고리즘

각 노드 사이의 거리

	1	2	3	4	5	6
1	∞	7	3	12	5	8
2	4	∞	2	10	9	3
3	6	7	∞	11	1	7
4	7	3	1	∞	8	8
5	2	10	2	7	∞	3
6	4	11	7	6	3	∞

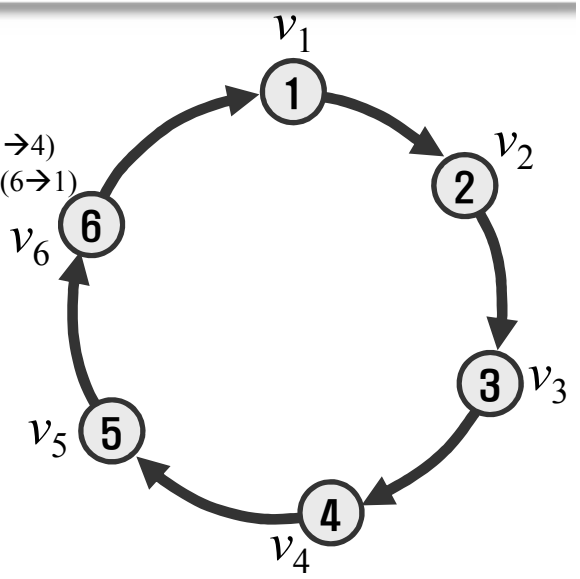
목적함수

Minimize

$$F = 7x_{12}+3x_{13}+12x_{14}+5x_{15}+8x_{16}+4x_{21}+2x_{23}+10x_{24}+9x_{25}+3x_{26}+6x_{31}+7x_{32}+11x_{34}+x_{35}+7x_{36}+7x_{41}+3x_{42}+x_{43}+8x_{45}+8x_{46}+2x_{51}+10x_{52}+2x_{53}+7x_{54}+3x_{56}+4x_{61}+11x_{62}+7x_{63}+6x_{64}+3x_{65}$$

초기순환로

L (총 이동 거리)
= (1→2)+(2→3)+(3→4)
+(4→5)+(5→6)+(6→1)
= 7+2+11+8+3+4
= 35



설계변수

$$x_{12}, x_{13}, x_{14}, x_{15}, x_{16}, x_{21}, x_{23}, x_{24}, x_{25}, x_{26}, x_{31}, x_{32}, x_{34}, x_{35}, x_{36}, x_{41}, x_{42}, x_{43}, x_{45}, x_{46}, x_{51}, x_{52}, x_{53}, x_{54}, x_{56}, x_{61}, x_{62}, x_{63}, x_{64}, x_{65}$$

제거 : $(v_i, v_{i+1}), (v_j, v_{j+1}), (v_l, v_{l+1})$

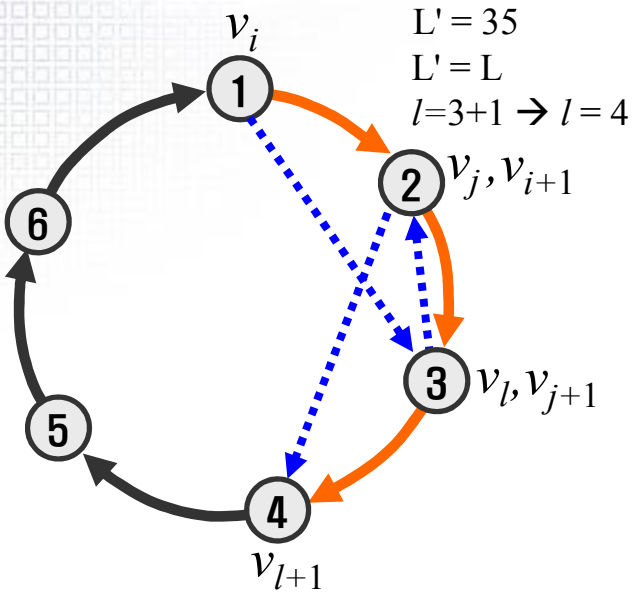
생성 : $(v_i, v_{j+1}), (v_l, v_{i+1}), (v_j, v_{l+1})$

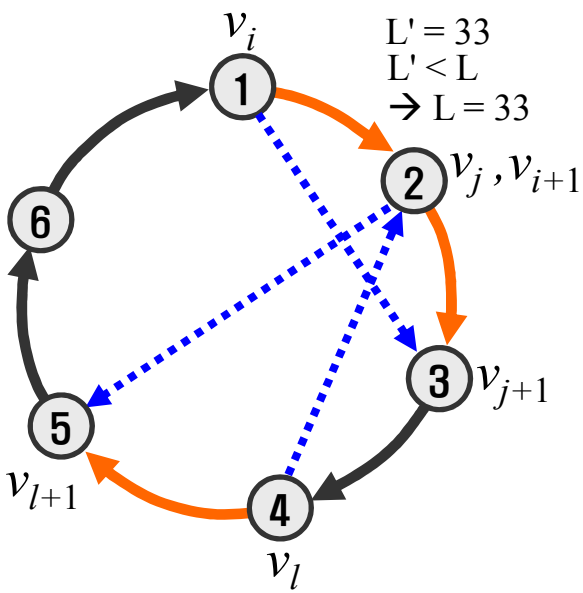
제약조건

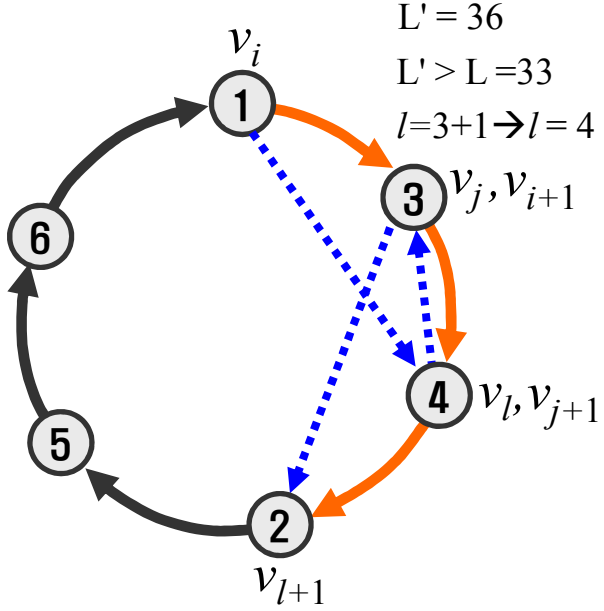
$$\sum_{j=1}^{N} x_{ij} = 1 \qquad \sum_{i=1}^{N} x_{ij} = 1$$

x_{ij} : 노드 i에서 노드 j로 이동하면 1, 그렇지않으면 0

x_{ij}는 하나의 Hamiltion 순환로를 형성



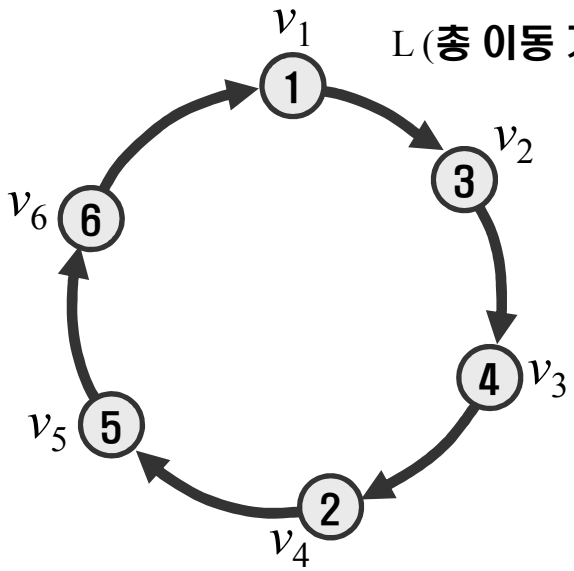




외판원 문제
-k-optimal 알고리즘

각 노드 사이의 거리

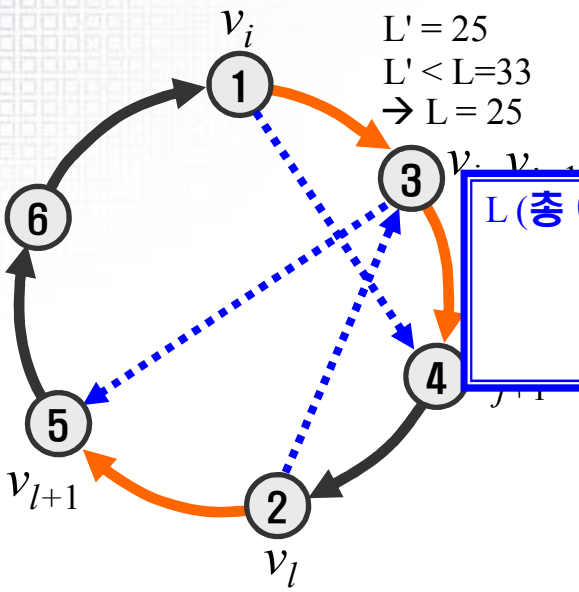
	1	2	3	4	5	6
1	∞	7	3	12	5	8
2	4	∞	2	10	9	3
3	6	7	∞	11	1	7
4	7	3	1	∞	8	8
5	2	10	2	7	∞	3
6	4	11	7	6	3	∞

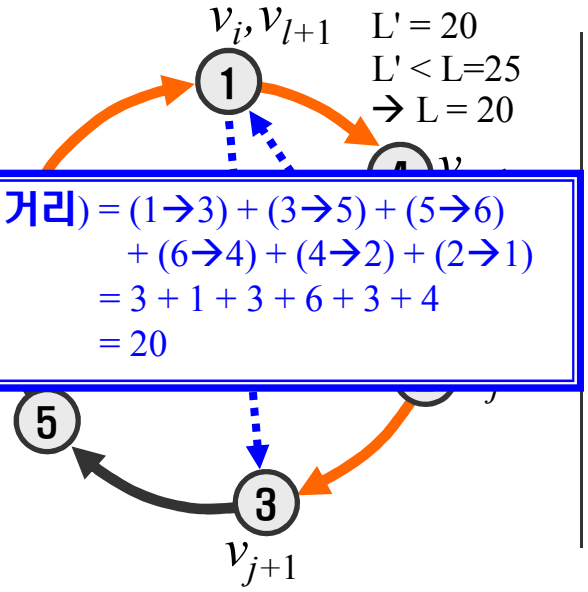


L (**총 이동 거리**) = 33

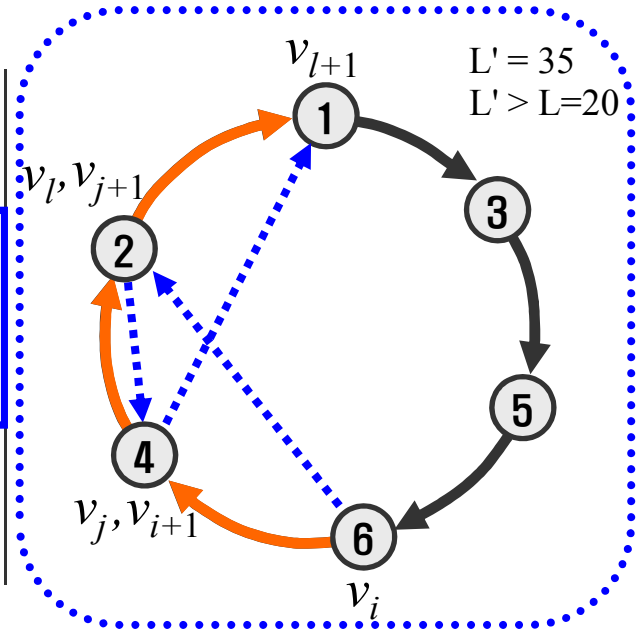
제거 : (v_i, v_{i+1}), (v_j, v_{j+1}), (v_l, v_{l+1})

생성 : (v_i, v_{j+1}), (v_l, v_{i+1}), (v_j, v_{l+1})



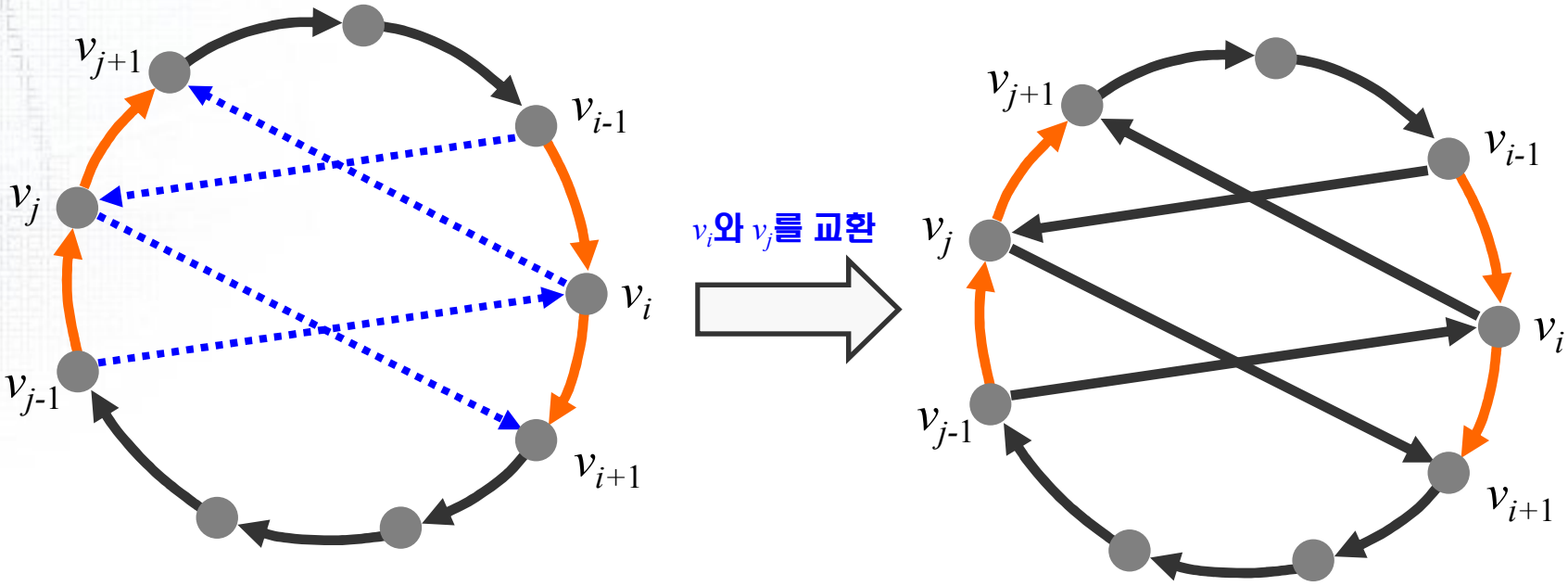


L (**총 이동 거리**) = (1→3) + (3→5) + (5→6)
+ (6→4) + (4→2) + (2→1)
= 3 + 1 + 3 + 6 + 3 + 4
= 20



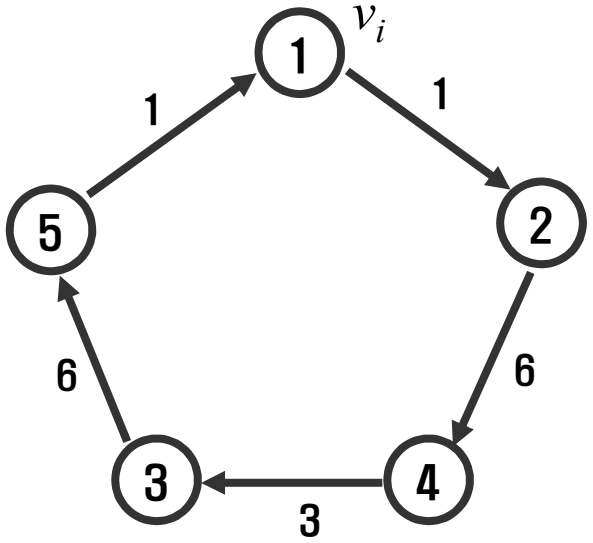
외판원 문제
- 교점교환 알고리즘

- k-optimal 알고리즘은 edge를 제거하고 새로운 edge로 대체하지만,
  교점교환 알고리즘은 edge를 대체하는 대신 교점을 교환(switch)한다.



외판원 문제
- 교점교환 알고리즘

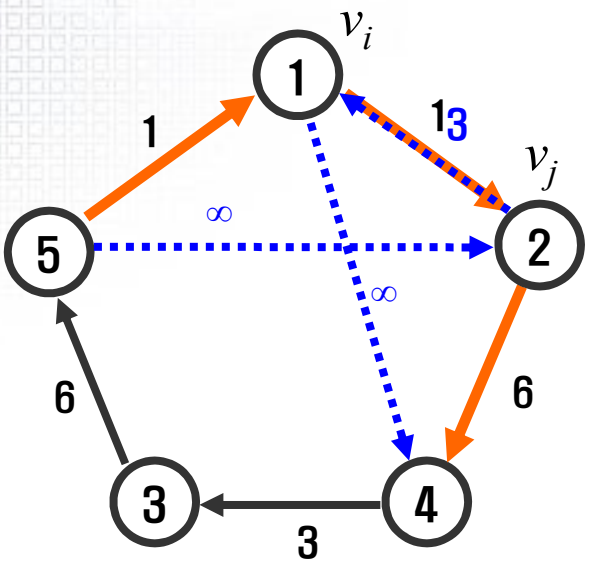
	1	2	3	4	5
1	∞	1	∞	∞	3
2	3	∞	1	6	∞
3	∞	3	∞	1	6
4	∞	∞	3	∞	1
5	1	∞	∞	3	∞

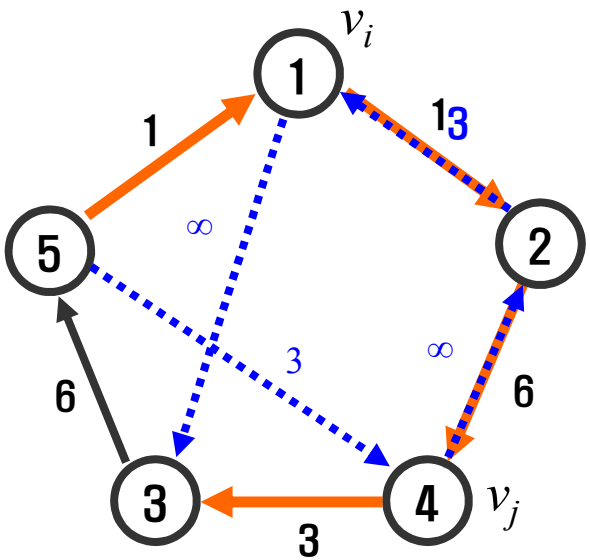


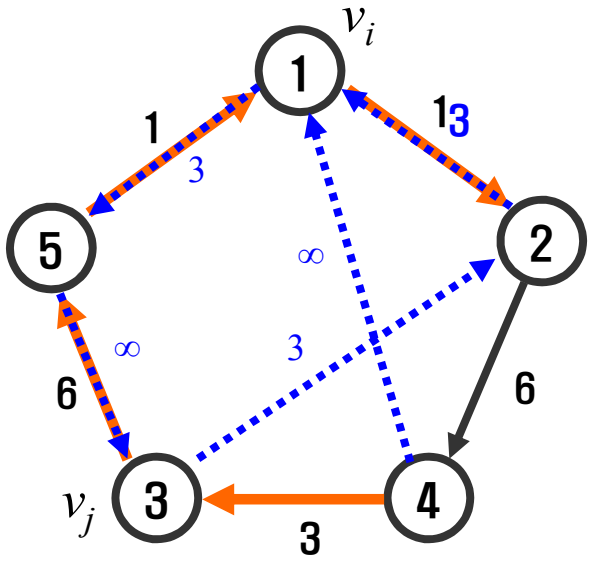
초기순환로

L (**총 이동 거리**) = (1→2) + (2→4) + (4→3)
+ (3→5) + (5→1)
= 1 + 6 + 3 + 6 + 1
= 17

v_j**와** v_j**를 교환**

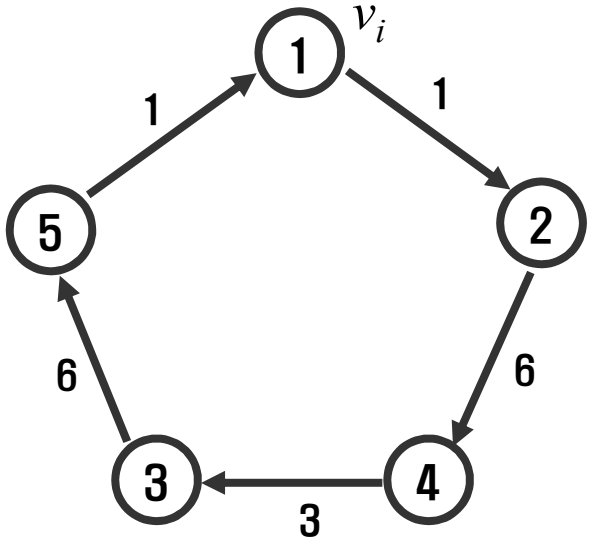






외판원 문제
- 교점교환 알고리즘

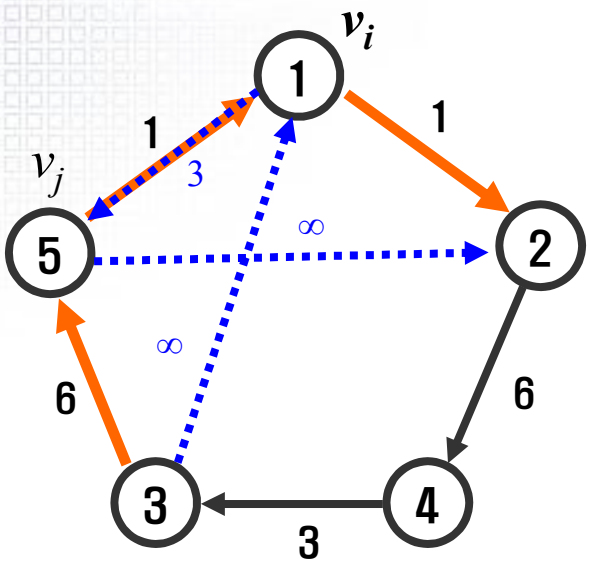
	1	2	3	4	5
1	∞	1	∞	∞	3
2	3	∞	1	6	∞
3	∞	3	∞	1	6
4	∞	∞	3	∞	1
5	1	∞	∞	3	∞

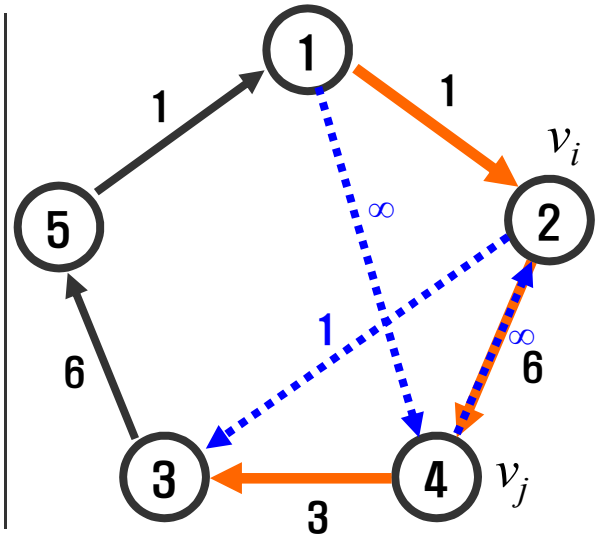


초기순환로

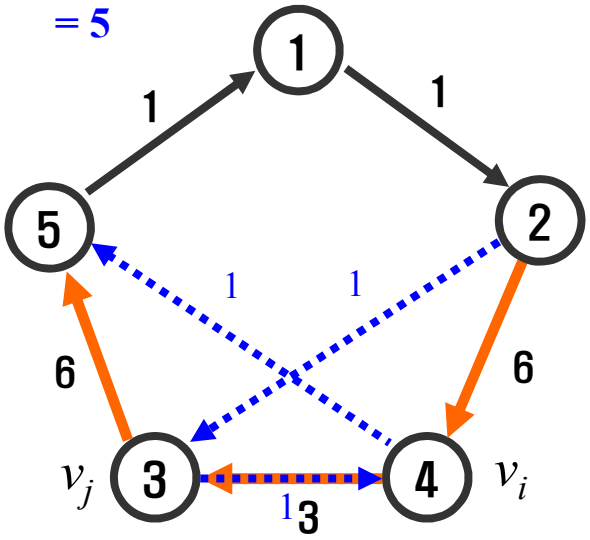
L (**총 이동 거리**) = (1→2) + (2→4) + (4→3)
+ (3→5) + (5→1)
= 1 + 6 + 3 + 6 + 1
= 17

v_j와 v_j를 교환





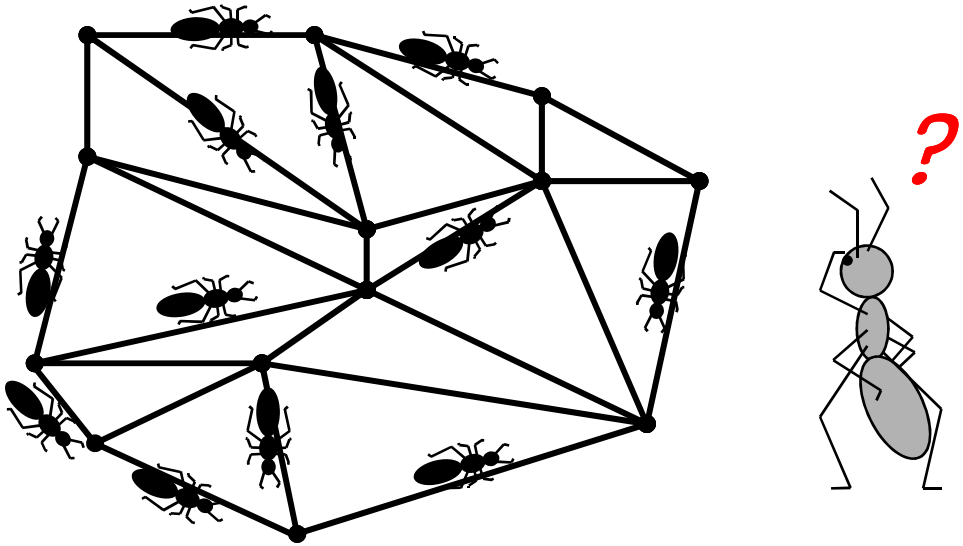
L' = 1 + 1 + 1 + 1 + 1
= 5



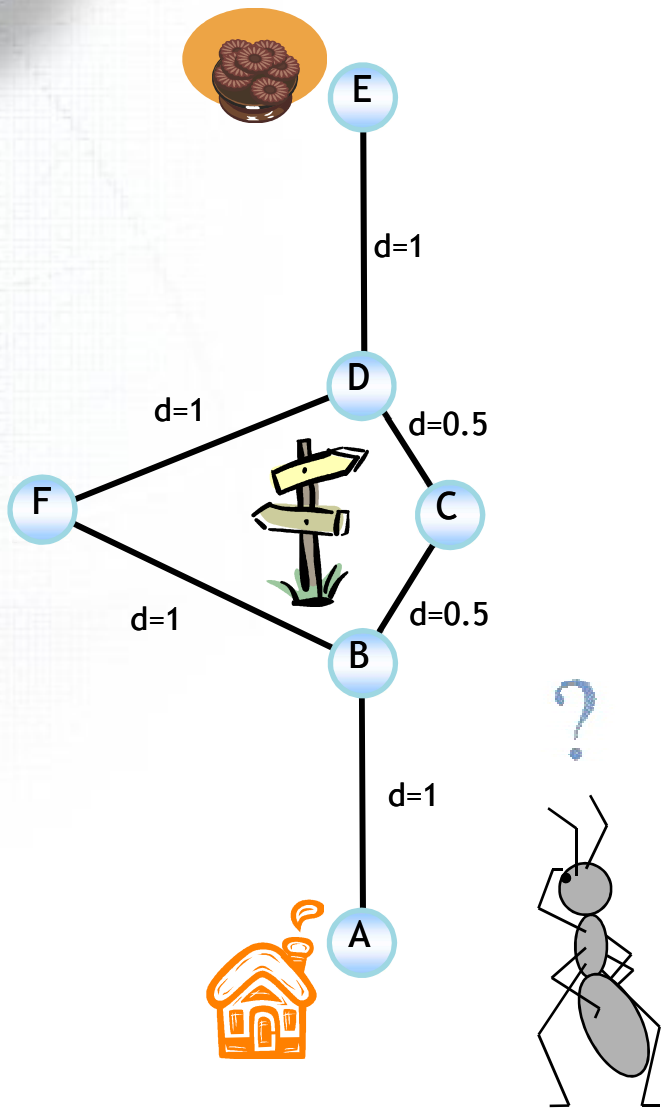
"Dorigo, M., Maniezzo, V. and Colorni, A., "Ant System: An Autocatalytic Optimizaing Process", Technical Report No. 91-016, 1991"
"Dorigo, M., Maniezzo, V. and Colorni, A., "Ant System: Optimization by a colony of cooperating agents", IEEE Transactions on System, Man and Cybernetics-Part B, Vol. 26, No. 1, 1-13, 1996"

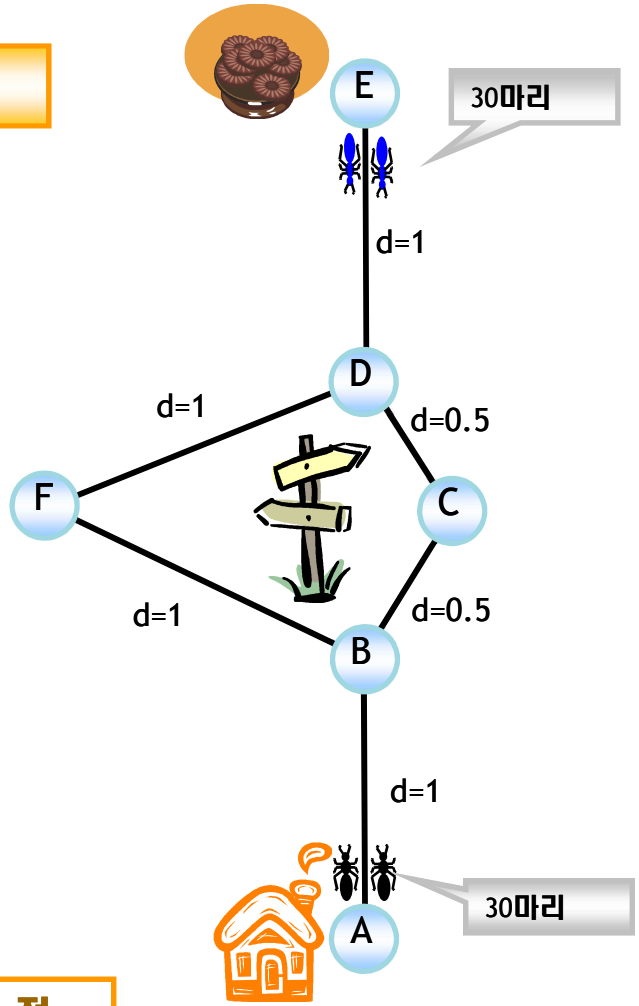
개미 시스템 알고리즘 (Ant System Algorithm)

- 1991, 유럽서 개발된 메타 휴리스틱
- TSP 문제, 할당문제 등에서 우수성 입증
- 유전 알고리즘(GA)과 대등 또는 더 나은 해 생성
- 매개변수에 대한 낮은 민감도
- 개미 무리들의 먹이를 옮기는 이동 행태로부터 착상
  임의의 개미들이 임의의 경로로 이동 (이동경로에 페르몬 흔적 남김) → 시간 경과 후 점차적으로 페르몬 흔적이 많은 곳으로 개미들의 경로 수정 → 궁극적으로 최단 경로를 찾음



개미 시스템 알고리즘 (개미들의 이동형태)

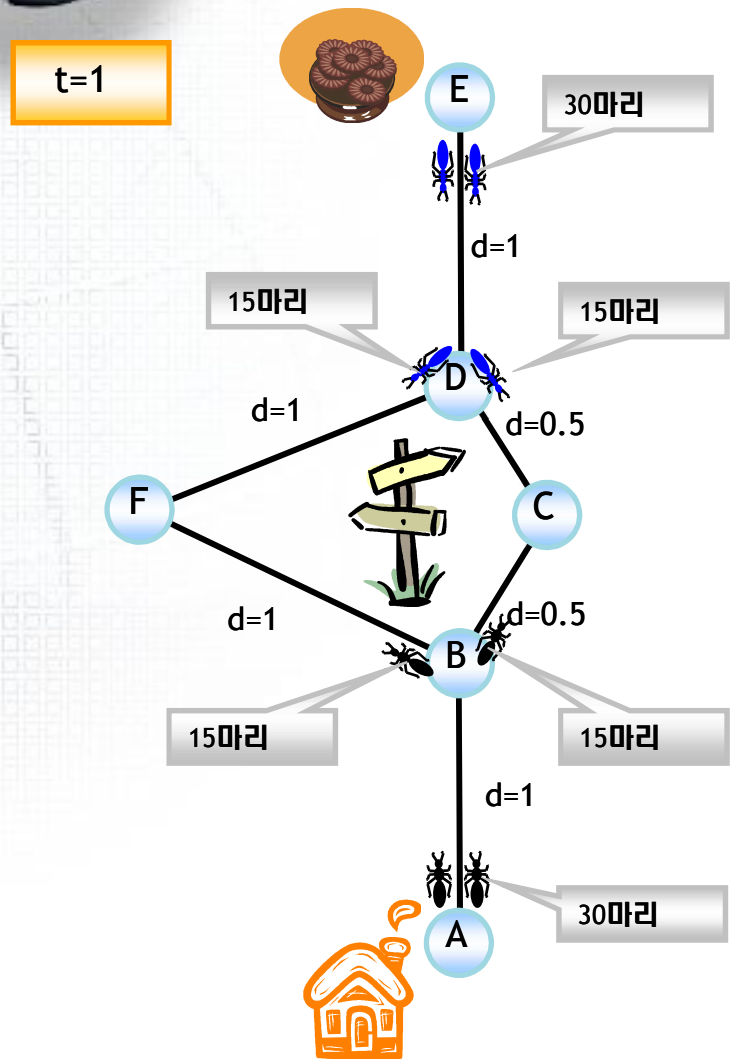


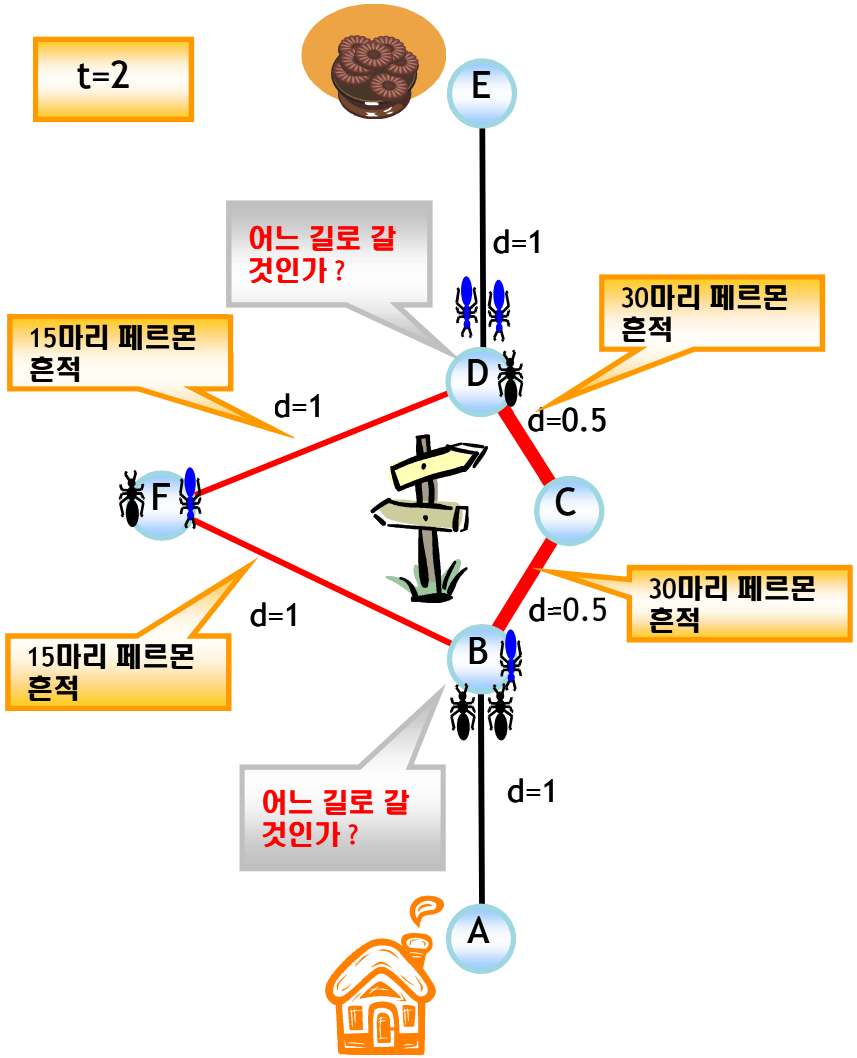


가 정

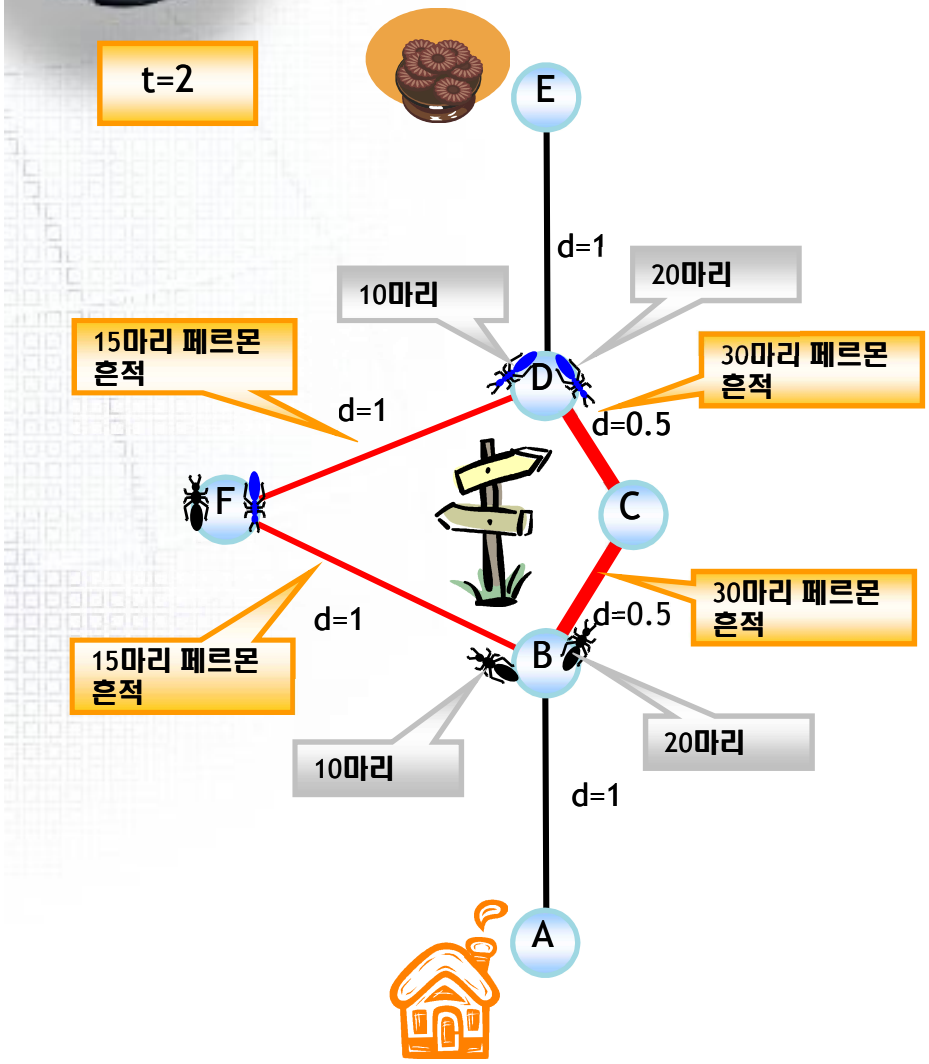
- 매 단위시간 마다 A, E에서 각각 30마리씩 출발
- 모든 개미의 이동속도는 일정, 이산형 시간

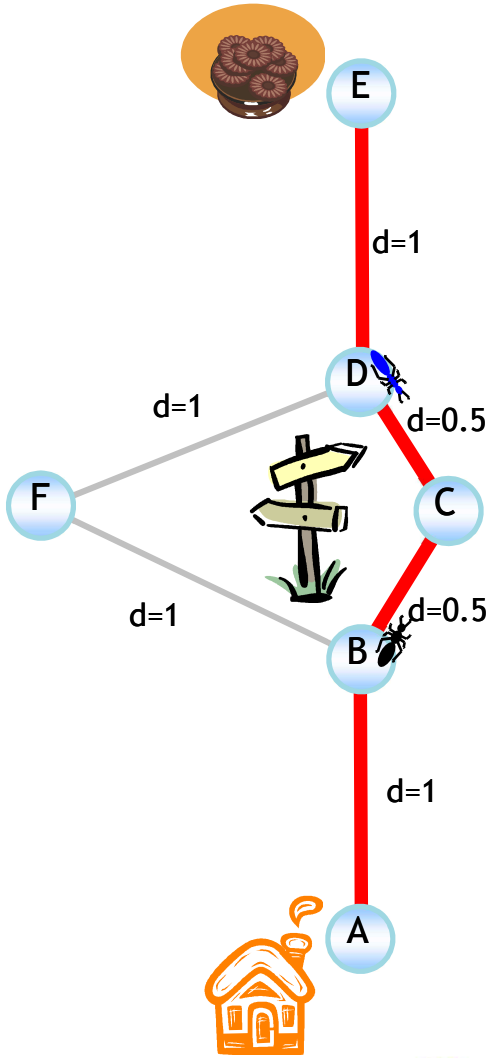
개미 시스템 알고리즘 (개미들의 이동형태)





개미 시스템 알고리즘 (개미들의 이동형태)





Bullnheimer*의 Ant Colony Optimization (ACO) 절차

Step 1 : 초기화
- 유인 물질 흔적 초기 매트릭스 생성.
- 모든 지점에 개미들을 위치 시킴.

Step 2 : 차량 경로 구축 (Savings 기법 응용)
- 각 개미들은 지점간 링크에 남은 유인 물질 흔적과 Saving의 값을 이용하여 차량 경로를 구축.

Step 3 : 유인물질 Local Update
- 2-opt 개선 후 우수한 해를 얻은 개미의 순위에 따라 유인 물질을 차등 추가.

(2-opt 알고리즘은 초기 경로의 두 도시의 위치를 바꾼 후, 더 나은 경로가 되는지를 탐색하되, 모든 경우를 탐색하여 그 중 가장 좋은 경로로 바꿈)

Step 4 : 유인 물질 Global Update 수정 및 증발
- 모든 유인 물질 흔적을 일정량 감소 (증발).

Step 5 : 종료 조건(주어진 반복횟수)

* Bullnheimer at al.(1999), An improved Ant System algorithm for the Vehicle Routing Problem
→ 차량경로문제에 대하여 적용

참고 자료

Advanced
Ship
Design
Automation
Laboratory

[참고] 절단 평면을 계산하는 방법

선형 계획법으로 문제를 푼 결과가 다음과 같다면,

$$
\begin{aligned}
f(\mathbf{x}) &: z \qquad\qquad + \bar{c}_{m+1}x_{m+1} + \cdots + \bar{c}_n x_n = z_0 \\
g_1(\mathbf{x}) &: x_1 \qquad\quad + \bar{a}_{1,\,m+1}x_{m+1} + \cdots + \bar{a}_{1,\,n}x_n = \bar{b}_1 \\
g_2(\mathbf{x}) &: \quad x_2 \qquad + \bar{a}_{2,\,m+1}x_{m+1} + \cdots + \bar{a}_{2,\,n}x_n = \bar{b}_2 \\
&\vdots \\
g_m(\mathbf{x}) &: \qquad\quad x_m + \bar{a}_{2,\,m+1}x_{m+1} + \cdots + \bar{a}_{2,\,n}x_n = \bar{b}_m
\end{aligned}
$$

기저 변수 비기저 변수(=0)

✓ **기저 변수:** $x_1 \sim x_m$

✓ **비기저 변수:** $x_{m+1} \sim x_n$

비기저 변수는 모두 0이므로,

$$x_1 = \bar{b}_1, x_2 = \bar{b}_2, \cdots x_n = \bar{b}_n$$

따라서 $\bar{b}_1, \bar{b}_2, \cdots \bar{b}_m$ **이 정수이면** $x_1 \sim x_n$ **이 정수이며,**
이 문제는 정수해를 가진다.

[참고] 절단 평면을 계산하는 방법

선형 계획법으로 문제를 푼 결과가 다음과 같다면,

$$
\begin{aligned}
f(\mathbf{x}) &: z \qquad\qquad + \bar{c}_{m+1}x_{m+1} + \cdots + \bar{c}_n x_n = z_0 \\
g_1(\mathbf{x}) &: x_1 \qquad\quad + \bar{a}_{1,\,m+1}x_{m+1} + \cdots + \bar{a}_{1,\,n}x_n = \bar{b}_1 \\
g_2(\mathbf{x}) &: \quad x_2 \qquad + \bar{a}_{2,\,m+1}x_{m+1} + \cdots + \bar{a}_{2,\,n}x_n = \bar{b}_2 \\
&\vdots \\
g_m(\mathbf{x}) &: \qquad\quad x_m + \bar{a}_{2,\,m+1}x_{m+1} + \cdots + \bar{a}_{2,\,n}x_n = \bar{b}_m
\end{aligned}
$$

기저 변수 비기저 변수 (=0)

$\bar{b}_1, \bar{b}_2, \cdots \bar{b}_m$ **가 정수가 아니면, 이 문제는 정수해를 가지지 않으므로 다음과 같이 절단 평면을 계산한다.**

좌변(bi)이 정수가 아닌 식을 선택하여 다음과 같이 변형한다.

$$x_i + \sum_{j=m+1}^{n} \bar{a}_{i,j}x_j = \bar{b}_i$$

$$x_i + \sum_{j=m+1}^{n} \left(\left[\bar{a}_{i,j} \right] + \bar{a}_{i,j} - \left[\bar{a}_{i,j} \right] \right) x_j = \left[\bar{b}_i \right] + \bar{b}_i - \left[\bar{b}_i \right]$$

여기서, $\left[\bar{a}_{i,j} \right]$는 $\bar{a}_{i,j}$보다 작은 가장 큰 정수를 뜻한다.

$$x_i + \sum_{j=m+1}^{n} \left(\left[\bar{a}_{i,j} \right] + \bar{a}_{i,j} - \left[\bar{a}_{i,j} \right] \right) x_j = \left[\bar{b}_i \right] + \bar{b}_i - \left[\bar{b}_i \right]$$

$f_{i,j} = \bar{a}_{i,j} - \left[\bar{a}_{i,j} \right] \quad f_i = \bar{b}_i - \left[\bar{b}_i \right]$ **로 치환하면**

$$x_i + \sum_{j=m+1}^{n} \left(\left[\bar{a}_{i,j} \right] + f_{i,j} \right) x_j = \left[\bar{b}_i \right] + f_i$$

$$x_i + \sum_{j=m+1}^{n} \left[\bar{a}_{i,j} \right] x_j + \sum_{j=m+1}^{n} f_{i,j}x_j = \left[\bar{b}_i \right] + f_i$$

$$x_i + \sum_{j=m+1}^{n} \left[\bar{a}_{i,j} \right] x_j - \left[\bar{b}_i \right] = f_i - \sum_{j=m+1}^{n} f_{i,j}x_j \qquad \text{--- ①}$$

$$f_i - \sum_{j=m+1}^{n} f_{i,j}x_j \le f_i < 1$$

[참고] 절단 평면을 계산하는 방법

선형 계획법으로 문제를 푼 결과가 다음과 같다면,

$$
\begin{aligned}
f(\mathbf{x}) &: z \qquad\qquad + \bar{c}_{m+1}x_{m+1} + \cdots + \bar{c}_n x_n = z_0 \\
g_1(\mathbf{x}) &: x_1 \qquad\quad + \bar{a}_{1,\,m+1}x_{m+1} + \cdots + \bar{a}_{1,\,n}x_n = \bar{b}_1 \\
g_2(\mathbf{x}) &: \quad x_2 \qquad + \bar{a}_{2,\,m+1}x_{m+1} + \cdots + \bar{a}_{2,\,n}x_n = \bar{b}_2 \\
&\vdots \\
g_m(\mathbf{x}) &: \qquad\quad x_m + \bar{a}_{2,\,m+1}x_{m+1} + \cdots + \bar{a}_{2,\,n}x_n = \bar{b}_m
\end{aligned}
$$

기저 변수 비기저 변수 (=0)

$\bar{b}_1, \bar{b}_2, \cdots \bar{b}_m$ 가 정수가 아니면, 이 문제는 정수해를 가지지 않으므로 다음과 같이 절단 평면을 계산한다.

좌변(bi)이 정수가 아닌 식을 선택하여 다음과 같이 변형한다.

$$x_i + \sum_{j=m+1}^{n} \bar{a}_{i,j}x_j = \bar{b}_i$$

$$x_i + \sum_{j=m+1}^{n} \left(\left[\bar{a}_{i,j} \right] + \bar{a}_{i,j} - \left[\bar{a}_{i,j} \right] \right) x_j = \left[\bar{b}_i \right] + \bar{b}_i - \left[\bar{b}_i \right]$$

여기서, $\left[\bar{a}_{i,j} \right]$는 $\bar{a}_{i,j}$보다 작은 가장 큰 정수를 뜻한다.

$f_{i,j} = \bar{a}_{i,j} - \left[\bar{a}_{i,j} \right]$ $f_i = \bar{b}_i - \left[\bar{b}_i \right]$ 로 치환하면

$$x_i + \sum_{j=m+1}^{n} \left[\bar{a}_{i,j} \right] x_j - \left[\bar{b}_i \right] = f_i - \sum_{j=m+1}^{n} f_{i,j}x_j \quad \text{--- ①}$$

$$f_i - \sum_{j=m+1}^{n} f_{i,j}x_j \le f_i < 1$$

식 ①의 x_i, x_j에 정수를 대입하면 좌변은 정수이므로

$$f_i - \sum_{j=m+1}^{n} f_{i,j}x_j \le 0 \quad \text{--- ②}$$

식 ②의 x_i, x_j에 (정수해가 아닌)최적해를 대입하면 X_j=0이므로 식 2를 만족하지 못한다.

$$f_i - \sum_{j=m+1}^{n} f_{i,j}x_j = f_i > 0$$

따라서 식 ②는 정수해는 만족하고, 정수해가 아닌 최적해는 만족하지 않는 절단 평면이 된다.