Ch.4. Traditional Methods - Part 2

Greedy Algorithm

■ 전체 문제를 여러 개의 단계 (step) 로 분할하여,
각 단계에서 가장 '우수한' 해를 구하고 이를 연결하여 전체 해 구성

greedy ← look for the best 'profit' at each step

■ 알고리즘
- 단순
- 각 단계의 가장 우수한 해를 연결하여 구한 해가 **최적 해는 아니다**.

(1) SAT

<방법 1>

- $x_1, \cdots, x_n$ 순으로 변수를 하나씩 결정 (n 개의 step 으로 문제 분할)
- 각 step 에서 변수 결정 방법 (greedy strategy)

  현재의 unsatisfied clause 들을 가장 많이 만족시키는 값(TRUE/FALSE) 할당

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

(step 1) x_1 = TRUE

∵ TRUE -> 3 개 clause 만족

FALSE -> 1 개 clause 만족

(step 2) x_2 = ?

(step 3) x_3 = ?

(step 4) x_4 = ?

<방법 2>

- 모든 clause 에 대한 출현 빈도를 기준으로 변수 $x_1, \cdots, x_n$ 정렬 (smallest -> largest 순)
- 정렬 순으로 변수를 하나씩 결정 (n 개의 step 으로 문제 분할)
- 각 step 에서 변수 결정 방법 (greedy strategy)

  현재의 unsatisfied clause 들을 가장 많이 만족시키는 값(TRUE/FALSE) 할당

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

(step 1) x_2 = TRUE

(step 2) x_3 = TRUE

(step 3) x_4 = TRUE

(step 4) x_1 = FALSE

(ex)

$(x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (\overline{x_1} \vee x_4) \wedge (\overline{x_2} \vee \overline{x_4}) \wedge (x_2 \vee x_5) \wedge (x_2 \vee x_6) \wedge F(x_3, x_4, x_5, x_6)$

빈도 수: $x_1 \rightarrow x_2 \rightarrow \cdots$

x_1 : TRUE

x_2 : TRUE

⋮

x_4 : ? $(\overline{x_1} \vee x_4) \wedge (\overline{x_2} \vee \overline{x_4})$

※ 모든 SAT 문제에 적용 가능한 greedy algorithm 은 없다!

(2) TSP

- 방문 도시를 1 개씩 추가: n 개의 step 으로 문제 분할

  1st city -> 2nd city -> … n-th city

- 각 방문 도시 추가 시 적절한 **greedy strategy** 적용

<방법 1> nearest-neighbor heuristic

- 방법

  임의의 시작 노드에서 시작하여, 가장 가까운 노드를 tour 에 추가
  마지막 노드를 시작노드에 연결하여 tour 완성

- 알고리즘

  procedure nearest_neighbor

  (1) Select an arbitrary node j, set $l=j$ and $T=\{1,2,\cdots,n\}-\{j\}$

  (2) As long as $T\neq\emptyset$ do the following

  (2.1) Let $j\in T$ such that $c_{lj}=\min\{\, c_{li} \mid i\in T \,\}$

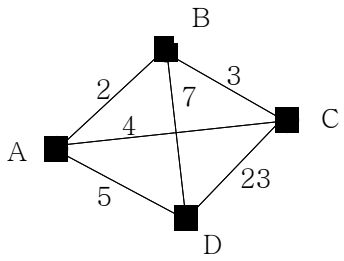
  (2.2) Connect l to j and set $T=T-\{j\}$

  (3) Connect l to the first node to form a tour

  end of nearest_neighbor

- running time = $O(n^2)$

(ex) Fig.4.1



A - B - C - D - A : 33 (greedy solution)

A - C - B - D -A : 19 (optimum solution)

<방법 2> insertion heuristic

- 방법

몇 개의 노드만을 갖는 작은 tour에서 시작

나머지 노드를 tour 에 삽입하여 tour 확장

- 알고리즘

procedure insertion

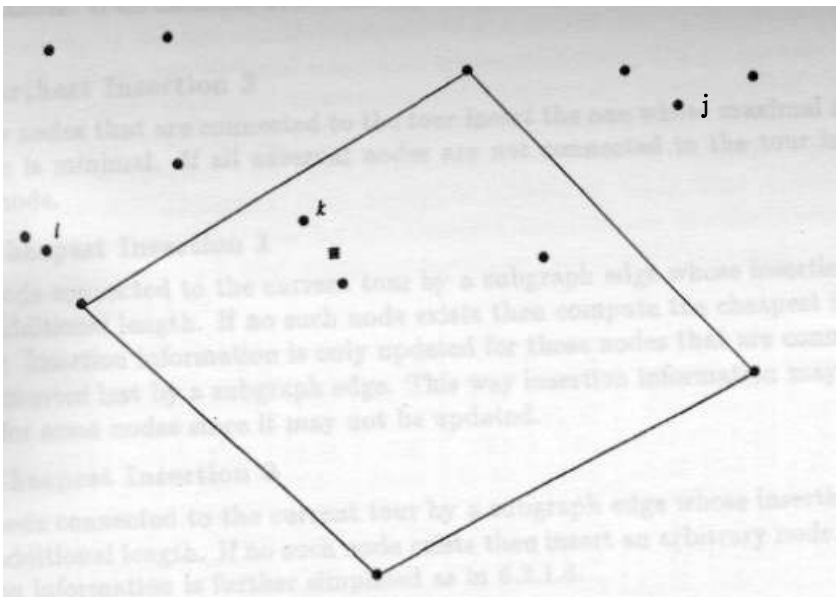
(1) Select a starting tour through k nodes $v_1, v_2, \cdots, v_k (k \geq 1)$, and set $W = V - \{v_1, v_2, \cdots, v_k\}$

(2) As long as $W \neq \emptyset$ do the following

(2.1) Select a node $j \in W$ according to some criterion.

(2.2) Insert j at some position in the tour and set $W = W - \{j\}$

end of insertion



- Insertion Criteria

$d_{\min}(j) = \min\{c_{ij} \mid i \in V - W\}$, $d_{\max}(j) = \max\{c_{ij} \mid i \in V - W\}$ (단 $j \in W$)

Nearest Insertion

insert the node that has the shortest distance to a node

select $j \in W$ such that $d_{\min}(j) = \min\{d_{\min}(l) \mid l \in W\}$

Farthest Insertion 1

insert the node whose minimal distance to a node is maximal

select $j \in W$ such that $d_{\min}(j) = \max\{d_{\min}(l) \mid l \in W\}$

Farthest Insertion 2

insert the node that has the shortest distance to a node

select $j \in W$ such that $d_{\max}(j) = \max\{d_{\min}(l) \mid l \in W\}$

Farthest Insertion 3

insert the node whose maximal distance to a node is minimal

select $j \in W$ such that $d_{\max}(j) = \max\{d_{\max}(l) \mid l \in W\}$

; running time = $O(n^2)$ - $O(n^2 \log n)$

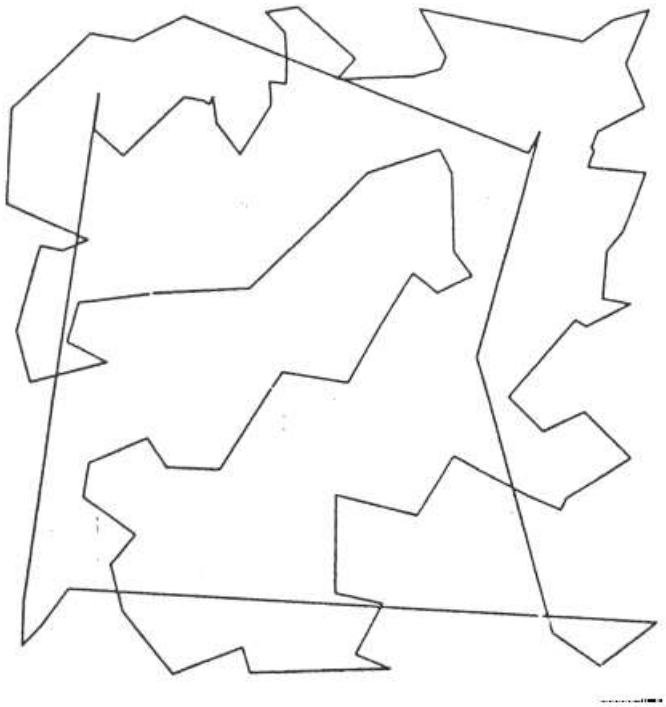
※ Convex Hull

- 초기 tour ($v_1, v_2, \cdots, v_k$) 를 convex hull 로 구성 : $\Theta(n \log n)$

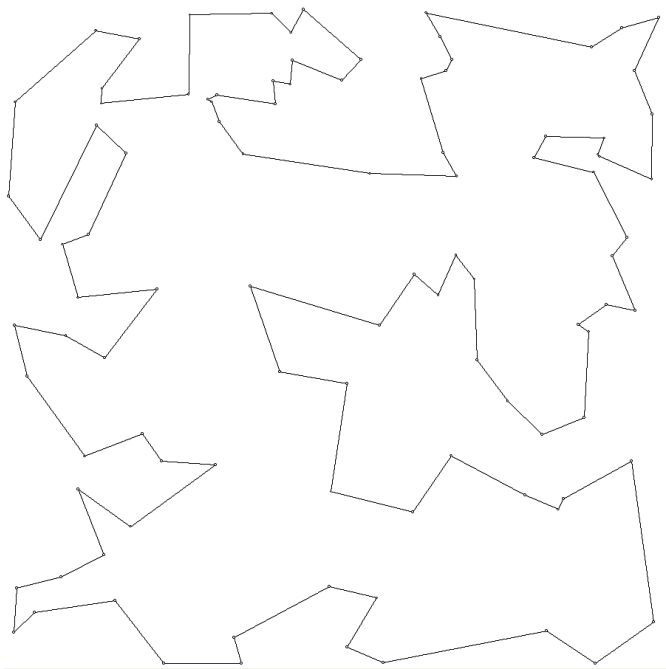
  ; convex hull = the smallest convex set containing all elements of a set

- insertion heuristics를 사용하여 tour 완성

=> cross-edge를 줄이기 위한 방법



(nearest neighbor)



(farthest insertion 1)

(3) NLP

- 변수 $x_1, \cdots, x_n$ 를 하나씩 결정 (n 개의 step 으로 문제 분할)
- 각 step 에서 변수 결정 방법 (greedy strategy)
  다른 변수들의 값은 constant 로 설정하고, 해당 변수를 변화시키며 목적 함수 값을 최대화시키는 값 결정 : *line search*

※ 목적함수 $f(x_1, \cdots, x_n)$ 에 대하여 각 변수 사이의 interaction 이 큰 경우, 해의 성능은 크게 낮아질 수 있음

■ Greedy Algorithm 특징

- Simple
- 변수들 간의 interaction 이 큰 복잡한 문제에 적용 시, 좋은 해를 구하지 못할 수 있다.
- Local search 를 위한 초기해 설정에 주로 사용됨.

(ex) Knapsack Problem

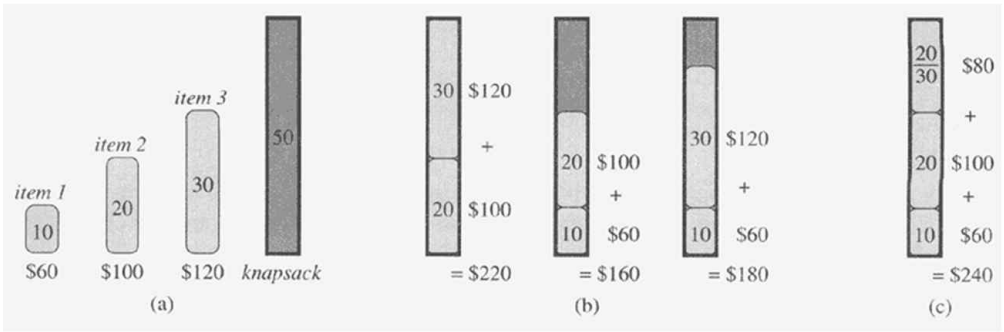
- Definition
  - Given
    - Items 1, 2, ..., n
    - i-th item: v_i dollars, w_i pounds
    - Max. pounds of knapsack: W
  - Store items to knapsack to maximize dollars
- Greedy strategy
  - Select item which has greatest value per pound

(ex) W = 50

i	1	2	3
v_i	60	100	120
w_i	10	20	30
v_i / w_i	6	5	4

(ex) 0-1 knapsack problem
- Allows only binary (0, 1) choice for each item
- Not solvable by greedy strategy

(ex) Fractional knapsack problem
- Allows fractional choice for each item
- Solvable by greedy strategy



0-1 knapsack fractional knapsack

Dynamic Programming

■ 전체 문제를 N x M 의 stage-state 구조로 분할

■ Principle of optimality

$$f_n^*(s) = \min_x \left\{ c_{s,x} + f_{n+1}^*(x) \right\} \ , n = 1,..., N,\ s = 1, \cdots M$$

$f_n^*(s)$: stage n 의 state s 에서 최종 stage 까지의 최소비용

$f_{n+1}^*(x)$: stage $(n+1)$ 의 state x 에서 최종 stage 까지의 최소비용

$c_{s,x}$: stage n 의 state s 에서 stage $n+1$ 의 state x 까지의 비용

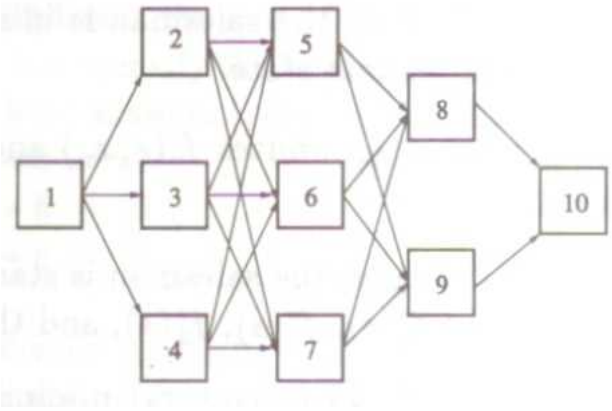
■ Recursive procedure

$$f_N^*(s) \rightarrow f_{N-1}^*(s) \rightarrow \cdots \rightarrow f_1^*(s)$$

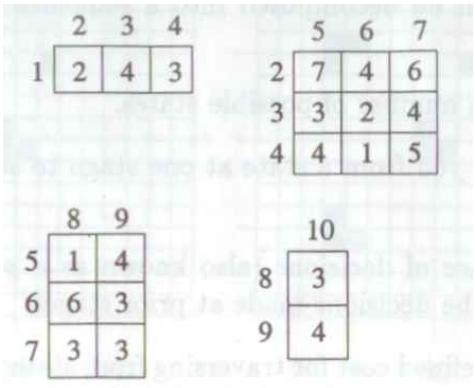
: backward search

■ Global optimum solution

(ex1) stagecoach problem



- cost table



	2	3	4
1	2	4	3

	5	6	7
2	7	4	6
3	3	2	4
4	4	1	5

	8	9
5	1	4
6	6	3
7	3	3

	10
8	3
9	4

- problem: [1] -> [10] 까지의 최소비용 경로 ?

- notation

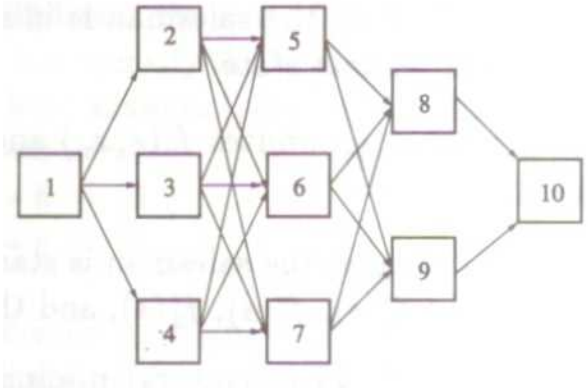
x_n $(n=1,2,3,4)$: stage n 에서 결정한 next state 의 번호

$f_n(s,x_n)$: stage n 의 state s 에서 final 까지의 cost. 단 stage $(n+1)$ 에서의 state 는 x_n

$x_n^*(s)$: $f_n(s,x_n)$ 를 최소화 시키는 x_n

$f_n^*(s)$: $f_n(s,x_n)$ 의 최소치

- solution



(i) stage 4

s	$f_4^*(s)$	$x_4^*(s)$
8	3	10
9	4	10

(ii) stage 3

$$f_3(s,x_3) = c_{s,x_3} + f_4^*(x_3)$$

$$f_3^*(s) = \min_{x_3} f_3(s,x_3)$$

s	$f_3(s,8)$	$f_3(s,9)$	$f_3^*(s)$	$x_3^*(s)$
5	4	8	4	8
6	9	7	7	9
7	6	7	6	8

(iii) stage 2

s	$f_2(s,5)$	$f_2(s,6)$	$f_2(s,7)$	$f_2^*(s)$	$x_2^*(s)$
2	11	11	12	11	5 or 6
3	7	9	10	7	5
4	8	8	11	8	5 or 6

(iv) stage 1

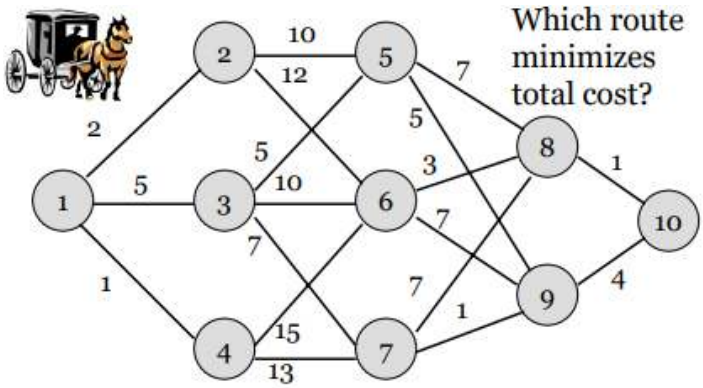
s	$f_1(s,2)$	$f_1(s,3)$	$f_1(s,4)$	$f_1^*(s)$	$x_1^*(s)$
1	13	11	11	11	3 or 4

min. cost = 11

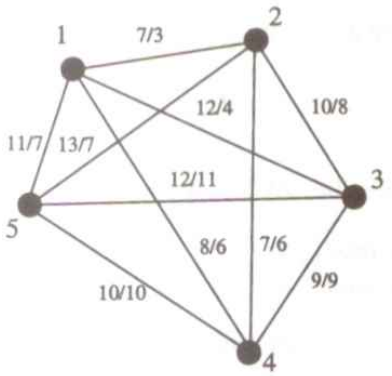
optimal path :

$$1 \to 3 \to 5 \to 8 \to 10$$
$$1 \to 4 \to 5 \to 8 \to 10$$
$$1 \to 4 \to 6 \to 9 \to 10$$

Quiz: stagecoach problem



(ex2) asymmetric TSP



$$L = \begin{bmatrix} 0 & 7 & 12 & 8 & 11 \\ 3 & 0 & 10 & 7 & 13 \\ 4 & 8 & 0 & 9 & 12 \\ 6 & 6 & 9 & 0 & 10 \\ 7 & 7 & 11 & 10 & 0 \end{bmatrix}$$

start city = 1

- notation

$g(i, S)$: city i 에서 시작하여 집합 S 의 모든 city를 1회씩 방문하고, city 1 에 도착하기 까지 소요되는 최단 거리

ex. $g(4, \{5,2,3\})$

$g(1, \{2,3,4,5\})$ => 문제

- DP formulation

$$g(i,S) = \min_{j \in S} \{L(i,j) + g(j, S - \{j\})\}$$

- solution

(i) $|S| = 0$

$$g(2,\emptyset) = L(2,1) = 3,$$
$$g(3,\emptyset) = L(3,1) = 4,$$
$$g(4,\emptyset) = L(4,1) = 6,$$
$$g(5,\emptyset) = L(5,1) = 7,$$

(ii) $|S| = 1$

$$g(2,\{3\}) = L(2,3) + g(3,\emptyset) = 10 + 4 = 14,$$
$$g(2,\{4\}) = L(2,4) + g(4,\emptyset) = 7 + 6 = 13,$$
$$g(2,\{5\}) = L(2,5) + g(5,\emptyset) = 13 + 7 = 20.$$

$$g(3,\{2\}) = L(3,2) + g(2,\emptyset) = 8 + 3 = 11,$$
$$g(3,\{4\}) = L(3,4) + g(4,\emptyset) = 9 + 6 = 15,$$
$$g(3,\{5\}) = L(3,5) + g(5,\emptyset) = 12 + 7 = 19,$$

$$g(4,\{2\}) = L(4,2) + g(2,\emptyset) = 6 + 3 = 9,$$
$$g(4,\{3\}) = L(4,3) + g(3,\emptyset) = 9 + 4 = 13,$$
$$g(4,\{5\}) = L(4,5) + g(5,\emptyset) = 10 + 7 = 17,$$

$$g(5,\{2\}) = g(5,\{2\}) = L(5,2) + g(2,\emptyset) = 7 + 3 = 10,$$
$$g(5,\{3\}) = g(5,\{3\}) = L(5,3) + g(3,\emptyset) = 11 + 4 = 15,$$
$$g(5,\{4\}) = g(5,\{4\}) = L(5,4) + g(4,\emptyset) = 10 + 6 = 16.$$

(iii) $|S| = 2$

$$\begin{aligned} g(2,\{3,4\}) &= \min\{L(2,3) + g(3,\{4\}), L(2,4) + g(4,\{3\})\} \\ &= \min\{10 + 15, 7 + 13\} = \min\{25, 20\} = 20, \\ g(2,\{3,5\}) &= \min\{L(2,3) + g(3,\{5\}), L(2,5) + g(5,\{3\})\} \\ &= \min\{10 + 19, 13 + 15\} = \min\{29, 28\} = 28, \\ g(2,\{4,5\}) &= \min\{L(2,4) + g(4,\{5\}), L(2,5) + g(5,\{4\})\} \\ &= \min\{7 + 17, 13 + 16\} = \min\{24, 29\} = 24. \end{aligned}$$

$$\begin{aligned} g(3,\{2,5\}) &= \min\{L(3,2) + g(2,\{5\}), L(3,5) + g(5,\{2\})\} \\ &= \min\{8 + 20, 12 + 10\} = \min\{28, 22\} = 22, \\ g(3,\{2,4\}) &= \min\{L(3,2) + g(2,\{4\}), L(3,4) + g(4,\{2\})\} \\ &= \min\{8 + 13, 9 + 9\} = \min\{21, 18\} = 18, \\ g(3,\{4,5\}) &= \min\{L(3,4) + g(4,\{5\}), L(3,5) + g(5,\{4\})\} \\ &= \min\{9 + 17, 12 + 16\} = \min\{26, 28\} = 26, \end{aligned}$$

$$\begin{aligned} g(4,\{2,3\}) &= \min\{L(4,2) + g(2,\{3\}), L(4,3) + g(3,\{4\})\} \\ &= \min\{6 + 14, 9 + 15\} = \min\{20, 24\} = 20, \\ g(4,\{2,5\}) &= \min\{L(4,2) + g(2,\{5\}), L(4,5) + g(5,\{2\})\} \\ &= \min\{6 + 20, 10 + 10\} = \min\{26, 20\} = 20, \\ g(4,\{3,5\}) &= \min\{L(4,3) + g(3,\{5\}), L(4,5) + g(5,\{3\})\} \\ &= \min\{9 + 19, 10 + 15\} = \min\{28, 25\} = 25. \end{aligned}$$

$$
\begin{aligned}
g(5,\{2,3\}) &= \min\{L(5,2)+g(2,\{3\}), L(5,3)+g(3,\{2\})\} \\
&= \min\{7+14, 11+11\} = \min\{21,22\} = 21, \\
g(5,\{2,4\}) &= \min\{L(5,2)+g(2,\{4\}), L(5,4)+g(4,\{2\})\} \\
&= \min\{7+13, 10+19\} = \min\{20,29\} = 20, \\
g(5,\{3,4\}) &= \min\{L(5,3)+g(3,\{4\}), L(5,4)+g(4,\{3\})\} \\
&= \min\{11+15, 10+13\} = \min\{26,23\} = 23.
\end{aligned}
$$

(iv) $|S| = 3$

$$
\begin{aligned}
&g(2,\{3,4,5\}) \\
&\quad = \min\{L(2,3)+g(3,\{4,5\}), L(2,4)+g(4,\{3,5\}), L(2,5)+g(5,\{3,4\})\} \\
&\quad = \min\{10+26, 7+25, 13+23\} = \min\{36,32,34\} = 32.
\end{aligned}
$$

$$
\begin{aligned}
&g(3,\{2,4,5\}) \\
&\quad = \min\{L(3,2)+g(2,\{4,5\}), L(3,4)+g(4,\{2,5\}), L(3,5)+g(5,\{2,4\})\} \\
&\quad = \min\{8+24, 9+20, 12+20\} = \min\{32,29,32\} = 29, \\
&g(4,\{2,3,5\}) \\
&\quad = \min\{L(4,2)+g(2,\{3,5\}), L(4,3)+g(3,\{2,5\}), L(4,5)+g(5,\{2,3\})\} \\
&\quad = \min\{6+28, 9+22, 10+21\} = \min\{34,31,31\} = 31, \text{ and} \\
&g(5,\{2,3,4\}) \\
&\quad = \min\{L(5,2)+g(2,\{3,4\}), L(5,3)+g(3,\{2,4\}), L(5,4)+g(4,\{2,3\})\} \\
&\quad = \min\{7+20, 11+18, 10+20\} = \min\{27,29,30\} = 27.
\end{aligned}
$$

(v) $|S| = 4$

$$
\begin{aligned}
&g(1,\{2,3,4,5\}) = \min\{L(1,2)+g(2,\{3,4,5\}), L(1,3)+g(3,\{2,4,5\}), \\
&\quad L(1,4)+g(4,\{2,3,5\}), L(1,5)+g(5,\{2,3,4\})\} = \\
&\quad \min\{7+32, 12+29, 8+31, 11+27\} = \min\{39,41,39,38\} = 38.
\end{aligned}
$$

path: 1-5-2-4-3-1

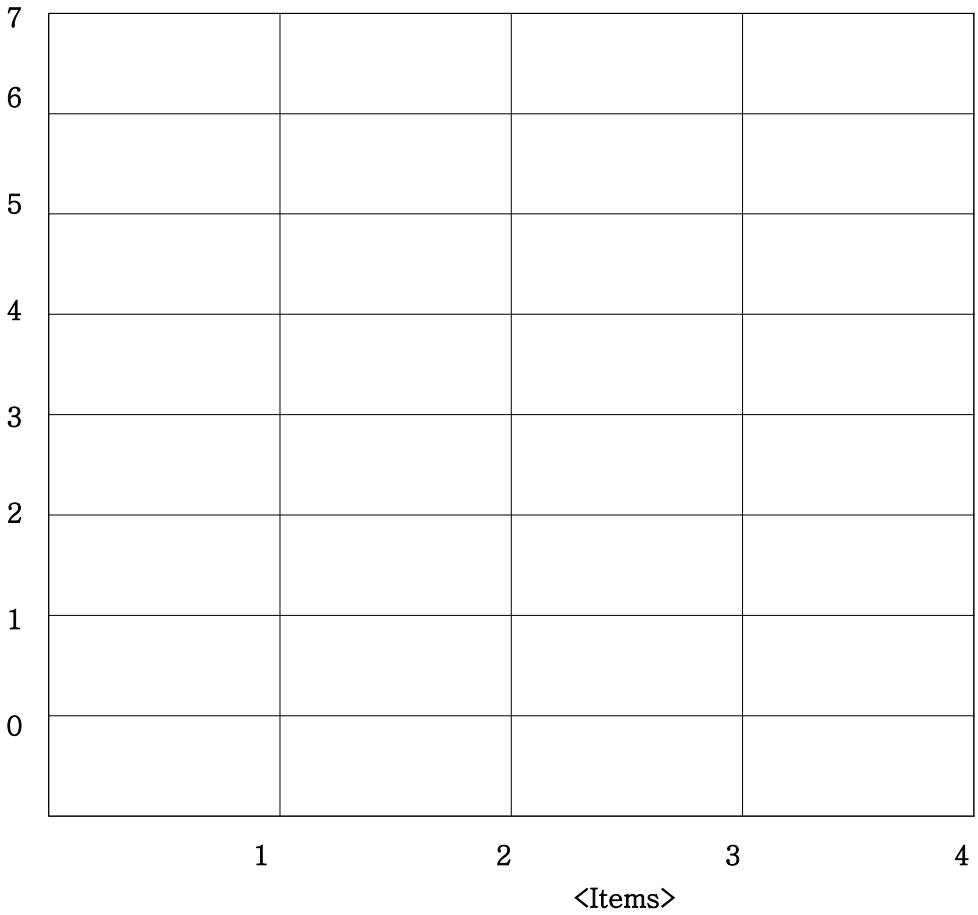
※ TSP 에 DP 적용 시 계산량이 지나치게 증가하므로, 적용 어렵다 !

(ex3) 0-1 Knapsack Problem

Items	1	2	3	4
Weight	1	3	4	5
Value	1	4	5	7

Total Weight = 7

<Remained Weights>




```
if(j < wt[i])
    T[i][j] = T[i-1][j]
else
    T[i][j] = max { val[i] + T[i-1][j-wt[i]]
                  { T[i-1][j]
```

https://www.youtube.com/watch?v=8LusJS5-AGo

(Quiz) item 1: 3kg 7$, item 2: 4kg 11$, item 3: 5kg 12$

Total weight = 12kg

=> Solve fractional and 0-1 knapsack problems

■ DP 적용이 가능한 문제

- The problem can be decomposed into a sequence of decision made at various *stages*.

- Each stage has a number of possible *states*.

- A decision takes you from a state at one stage to some state at the next stage.

- The best sequence of decisions at any stage is *independent* of decisions made at prior stages.

- There is well-defined cost for traversing from state to state across stages. Moreever, there is a *recursive relationship* for choosing the best decisions to make.