DP - **Floyd-Warshall Algorithm / O1 Knapsack**

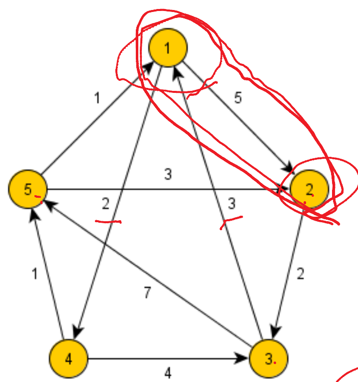
2017년 10월 16일 월요일

오후 12:07

Problem : All-pairs shortest path in Directed Graph.

$$\begin{cases} \text{vertices} : v_1, \ldots, v_n \\ \ell(v_i, v_j) : \text{length of edge from } (v_i, v_j) \end{cases}$$

Determine the cost of shortest path from v_i to v_j for all pairs (i, j)



$1 \sim 5$ $\quad$ $i \neq i$ & $k=0$

$1 \sim 4$ $\quad$ 2

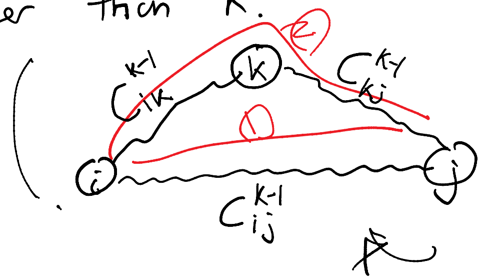
C^0_{14}

D^1_{53} vs $D^1_{52} + D^1_{23}$

$$D_0 = \begin{pmatrix} 0 & 5 & \infty & 2 & \infty \\ \infty & 0 & 2 & \infty & \infty \\ 3 & \infty & 0 & \infty & 7 \\ \infty & \infty & 4 & 0 & 1 \\ 1 & 3 & \infty & \infty & 0 \end{pmatrix} \quad D_1 = \begin{pmatrix} 0 & 5 & \infty & 2 & \infty \\ \infty & 0 & 2 & \infty & \infty \\ 3 & 8 & 0 & 5 & 7 \\ \infty & \infty & 4 & 0 & 1 \\ 1 & 3 & \infty & 3 & 0 \end{pmatrix} \quad D_2 = \begin{pmatrix} 0 & 5 & \boxed{7} & 2 & \infty \\ \infty & 0 & 2 & \infty & \infty \\ 3 & 8 & 0 & 5 & 7 \\ \infty & \infty & 4 & 0 & 1 \\ 1 & 3 & \boxed{5} & 3 & 0 \end{pmatrix}$$

$$D_3 = \begin{pmatrix} 0 & 5 & 7 & 2 & 14 \\ 5 & 0 & 2 & 7 & 9 \\ 3 & 8 & 0 & 5 & 7 \\ 7 & 12 & 4 & 0 & 1 \\ 1 & 3 & 5 & 3 & 0 \end{pmatrix} \quad D_4 = \begin{pmatrix} 0 & 5 & 6 & 2 & 3 \\ 5 & 0 & 2 & 7 & 8 \\ 3 & 8 & 0 & 5 & 6 \\ 7 & 12 & 4 & 0 & 1 \\ 1 & 3 & 5 & 3 & 0 \end{pmatrix} \quad D_5 = \begin{pmatrix} 0 & 5 & 6 & 2 & 3 \\ 5 & 0 & 2 & 7 & 8 \\ 3 & 8 & 0 & 5 & 6 \\ 2 & 4 & 4 & 0 & 1 \\ 1 & 3 & 5 & 3 & 0 \end{pmatrix}$$

- let C^k_{ij} be length of shortest path with no intermediate vertex of index larger than k.



$$C^k_{ij} = \begin{cases} 0 & \text{if } i = j \\ \ell(v_i, v_j) & \text{if } i \neq j, k = 0 \\ \min\{C^{k-1}_{ij},\ C^{k-1}_{ik} + C^{k-1}_{kj}\} & \text{if } i \neq j \text{ \& } k > 0 \end{cases}$$

Floyd-Warshall Algo

```
for i ← 1 to n do
    for j ← 1 to n do
        b_ij ← 0
        if i = j then
            c^0_ij ← 0
        else c^0_ij ← l(v_i, v_j)
        endif
    endfor
endfor
```

time? $\Theta(n^3)$

space $\Theta(n^2)$

```
for k ← 1 to n do
    for i ← 1 to n do
        for j ← 1 to n do
            ~~c^k_ij ← c^(k-1)_ij~~
            value ← c^(k-1)_ik + c^(k-1)_kj
            if value < c^k_ij
            then c^k_ij ← value
                 b_ij ← k
            endif
        endfor
    endfor
endfor
```

$\min\{c^{k-1}_{ij}, c^{k-1}_{ik} + c^{k-1}_{kj}\}$

/* path reconstruction */

Path(i,j)

if $b_{ij} = 0$ (Null) then

```
    return Null
path = i
while i ≠ j
    i ← b_ij
    path.append(i)
return path.
```

DP: Problem 01 Knapsack problem.

Gives a set of items, I, indexed from 1 upto n, where item i has weight $w_i > 0$ and profit p_i, and a knapsack capasity of M,

find the subset of I that maximize $\sum_{i \in I} p_i$

subjected to $\sum_{i \in I} w_i \leq M$

ex) $n = 3$ $M = 30$

	1	2	3
p_i	20	17	16
w_i	16	15	15

Approach I: (Brute-force.

Check each subset of $\{1 \dots n\}$

\# $\{ \overset{1}{0/1} \quad 0/1 \qquad \overset{n}{0/1} \}$

$\underline{2^n}$

let R^i_ℓ be the maximum profit possible using a subset of elements indexed $\{1 \dots i\}$ and yeild weight exactly ℓ $(i \leq n, \ 0 \leq \ell \leq W)$