CSPC 42 - Algo.

106119100

Cycle Test-2

21-04-2021

1

Rajneesh Pandey

Question 1

we have graph $G=(V,E)$,
V: set of cities, $e(u,v)$: edge
E: weighted edges (sets) $d(u,v)$: distance

$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 10 & 15 & 20 \\ 2 & 5 & 0 & 9 & 10 \\ 3 & 6 & 13 & 0 & 12 \\ 4 & 8 & 8 & 9 & 0 \end{array}$$

start from city 1 after visiting some city now we are in city j.

this is a partial tour.
we certainly know j, since this will determine which city are most convinent to visit next.

For a subset S of cities $S \in \{1,2,---n\}$ that include 1 and $j \in S$.
let $C(S,j)$ be the shortest path length visiting each node in S exactly once, start at 1 and end with j.

when $|S|>1$, we say $C(S,1)=\infty$,

Now, $C(S,j)$ will define as,

$C(S,j) = \min C(S-\{j\},i) + d(i,j)$ where $i \in S$ and $i \neq j$ $C(S,j) = \min C(S-\{j\},i) + d(i,j)$ where $i \in S$ and $i \neq j$

Now

$\boxed{S = \phi}$

cost $(2, \phi, 1) = d(2,1) = 5$ Cost $(2, \phi, 1) = 5$
cost $(3, \phi, 1) = d(3,1) = 6$ Cost $(3, \phi, 1) = 6$
cost $(4, \phi, 1) = d(4,1) = 8$ Cost $(4, \phi, 1) = 8$

$\boxed{S = 1}$

cost $(i, s) = \min \{ \text{cost}(j, S-j) + d[i,j] \}$ Cost(i, s)
$= \min \{ \text{Cost}(j, S) - j) + d[i,j]$

cos $(2, \{3\}, 1) = 9 + 6 = 15$
cost $(2, \{4\}, 1) = 10$
cost $(3, \{2\}, 1) = 18$
cost $(3, \{4\}, 1) = 12 + 8 = 20.$
cost $(4, \{3\}, 1) = 9 + 6 = 15$
cost $(4, \{2\}, 1) = 8 + 5 = 13$

$\boxed{S = 2}$ cost $(2, \{3,4\}, 1) = 25$
Cost $(3, \{2,4\}, 1) = 25$
cost $(4, \{2,3\}, 1) = 23$

$\boxed{S = 3}$

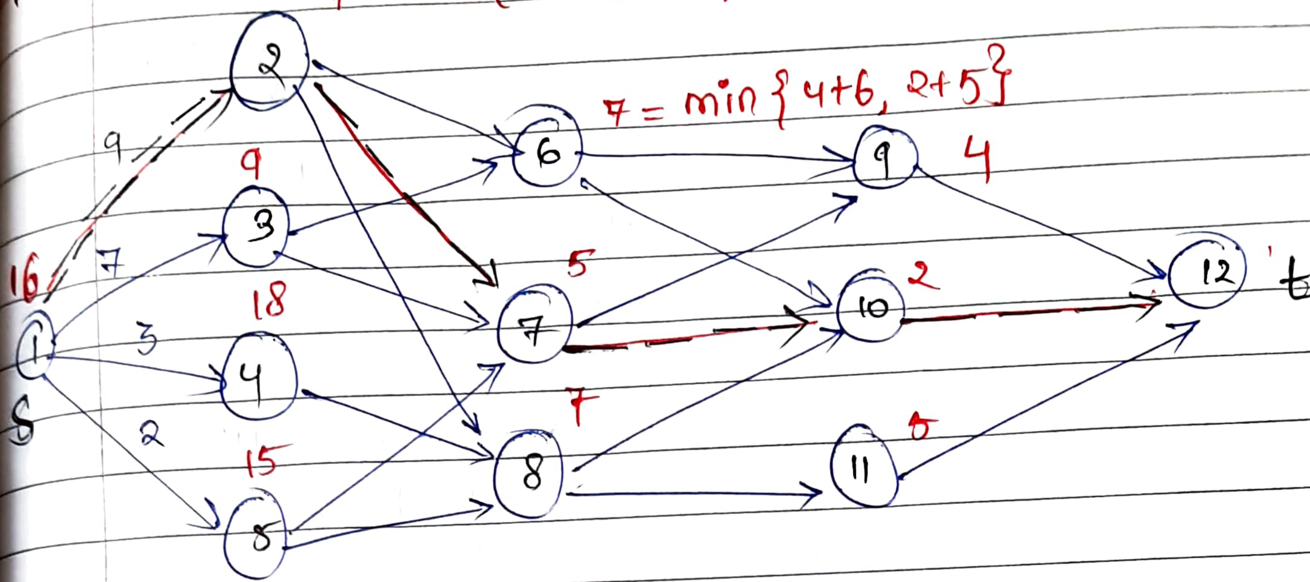
cost $(1, \{2,3,4\}, 1) = 35$

hence, the minimum cost path 35.

start from cost $\{1, \{2,3,4\}, 1\}$, we get the minimum value for $d[1,2]$, when $S = 3$ select path 1 to 2 (cost 10). $d[4,2]$ cost (10) go back, $d[3,4]$ cost (9) and $d[3,1]$ cost (6)

$$1 \xrightarrow{10} 2 \xrightarrow{10} 4 \xrightarrow{9} 3 \xrightarrow{6} 1$$

Question (2)



$$cost(i,j) = \min_{\substack{l \in V_{i+1} \\ \langle j,l\rangle \in E}} \{ c(j,l) + cost(i+1,l) \}$$

stage (j) → (l) --- (t)

So the shortest path from (s) to (t) is :-

$1 \xrightarrow{9} 2 \xrightarrow{2} 7 \xrightarrow{3} 10 \xrightarrow{2} 12$

s ... t

⇒ Most suitable way to solve the multistag is using dynamic programming.

They can be solved using forward or backward approach.

At every step we using minimum cost path.

Question (3)

→ Two major types of knapsack problems are :

(i) 0-1 Knapsack

in this problem the choice of the object to take is either yes or no.

(ii) fractional knapsack

In this problem the object we can take it partially into the knapsack

→ No, 0-1 knap sack cannot be solved using greedy. as Greedy approaches does not ensure optimal. solution.

In many instances, Greedy approaches may give an optimal solution

Example:

Item	A	B	C	D
Profit	24	18	18	10
weight	24	10	10	7

W= 25

consider the profit per unit weight Pi/wi, if we apply Greedy to solve this problem, first item "A" will selecte, after no item can be selected so, total profit = 24

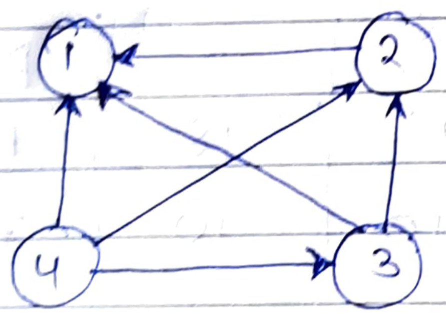
but the optimal solution is B and C which is 18+18 = 36.

we, can solve it using Dynamic Programming approach.

Algorithm :

```
0-1-knapsack (v, w, n W)
    for w = 0 to W do
        c[0, w] = 0
    for i = 1 to n do
        c[i, 0] = 0
    for w = 1 to W do
        if wi ≤ w then
            if vi + c[i-1, w-wi] then
                c[i, w] = vi + c[i-1, w-wi]
        else   c[i, w] = c[i-1, w]
    else
        c[i, w] = c[i-1, w]
```

Question (4)



Transitive closure is the reachability matrix to reach from vertex u to vertex v of a graph.

we can use the floyd warshall algorithm to calculate Transitive closure.

$$G(V,E) = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 \\ 3 & 1 & 1 & 0 & 0 \\ 4 & 1 & 1 & 1 & 0 \end{array}$$

the above matrix is the representation of graph.

So, using the Floy warshal algo

transitive closure will be.

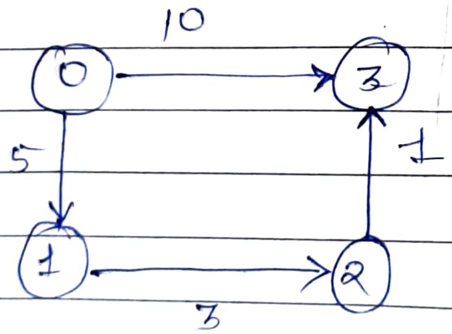
$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 1 & 1 \\ 2 & 0 & 0 & 1 & 1 \\ 3 & 0 & 0 & 0 & 1 \\ 4 & 0 & 0 & 0 & 0 \end{array}$$

Question (5)

Dijkstra's Algorithm: $O(E \log V)$
is one example of a single source shortest algorithm.
given a source vertex it finds shortest path from source to all other vertices
→ won't work on -ve edge

Floyd warshall's Algorithm: $O(V^3)$
Its an example of all pair shortest path algorithm, meaning it computes the shortest path between all pair of nodes
→ works on -ve edge but not on negative cycle

Example:



so, using Floyd warshall Algo, all pair shortest path

	0	1	2	3
0	0	5	8	9
1	INF	0	3	4
2	INF	INF	0	1
3	INF	INF	INF	0

and using Dijkstra's algo. for every node

0 →	1: 5	2: 8	3: 9
1 →	0: ∞	2: 3	3: 4
2 →	0: ∞	1: ∞	3: 1
3 →	0: ∞	1: ∞	2: ∞