Dynamic Programming

All pair shortest path Problem:

Let $G(V, E)$ be a directed graph with n vertices.

Let cost be a cost Adjacency matrix A for G such that

$cost(i, i) = 0$, $1 \le i \le n$. and

$cost(i, j) =$ cost of edge $\langle i, j\rangle \in E$ if $(i, j) \in E$

$cost(i, j) = \infty$ if edge $(i, j) \notin E$

Our objective is to compute the length of shortest path between each pair of nodes. (We Assume that there is no cycle of negative length)

For finding the shortest path between each pair of nodes of Graph $G(V, E)$ consisting n vertices requires total n iterations.

⇒ After iteration k, cost matrix A gives the length of shortest path that only uses nodes $\{1, 2 .. k\}$ as an Intermediate nodes

⇒ At iteration k, the Algorithm must check for each pair of nodes (i, j) whether or not there exist a path from i to j passing through Vertex k, that is better than the present optimal path through nodes in $\{1, 2 .. k-1\}$.

⇒ i.e. $A^k[i, j] = \min\left\{A^{k-1}[i, j], A^{k-1}[i, k] + A^{k-1}[k, j]\right\}$

$k \ge 1$

Algorithm All pairpaths (cost, A, n)

// cost[1..n, 1..n] is the cost Adjacency matrix of graph G with n vertices.

// A[i, j] is cost of shortest path from vertex i to j

// cost[i, i] = 0 for 1 ≤ i ≤ n.

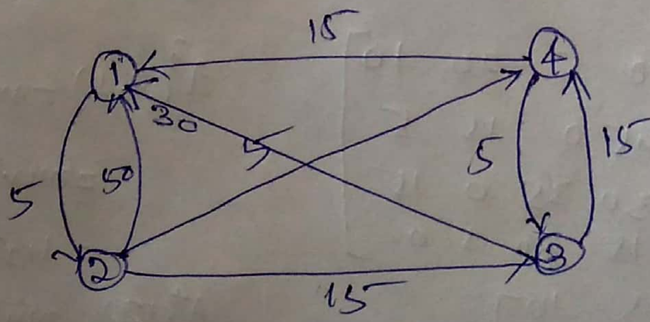
{

```
for i <- 1 to n do
for j <- 1 to n do
    A[i,j] = cost[i,j];   || copy cost into matrix A
end for
end for
  for k <- 1 to n do
   for i <- 1 to n do
   for j <- 1 to n do
        A[i,j] <- min(A[i,j], A[i,k] + A[k,j]);
    end for
   end for
  end for
return (A)   || matrix of shortest path between each
                 pair of vertices.
} || End of Algorithm
```

⇒ this Algorithm is known as Floyd-Warshall's Algorithm.

Time complexity $= O(n^3)$

<u>Example:</u> From the Graph Given below, find the shortest path between each pair of vertices



Find the cost Adjacency matrix of Given graph: (2)

$$A^0 = \begin{array}{c|c} & \begin{matrix}1 & 2 & 3 & 4\end{matrix} \\ \begin{matrix}1\\2\\3\\4\end{matrix} & \begin{bmatrix} 0 & 5 & \infty & \infty \\ 50 & 0 & 15 & 5 \\ 30 & \infty & 0 & 15 \\ 15 & \infty & 5 & 0 \end{bmatrix}\end{array}$$

= cost Adjacency matrix in which
cost(i, i) = 0 i = 1, - n
cost(i, j) = cost of edge(i, j) if (i, j) ∈ E
cost(i, j) = ∞ if (i, j) ∉ E

Find A^1 from A^0 which gives a minimum cost matrix obtained by using {1} as an intermediate node

$$A^1 = \begin{bmatrix} 0 & 5 & \infty & \infty \\ 50 & 0 & 15 & 5 \\ 30 & 35 & 0 & 15 \\ 15 & 20 & 5 & 0 \end{bmatrix}$$

$A^1[3,2] = \min\{A^{k-1}[3,2], A^{k-1}[3,1] + A^{k-1}[1,2]$

$= \min\{\infty, 30+5$

$= 35$

Similarly we obtain A^2 from A^1 which is minimum cost matrix obtained by using {1, 2} as an intermidiate node

$$A^2 = \begin{bmatrix} 0 & 5 & 20 & 10 \\ 50 & 0 & 15 & 5 \\ 30 & 35 & 0 & 15 \\ 15 & 20 & 5 & 0 \end{bmatrix}$$

$A^2[1,3] = \min\{A^1(1,3), A^1(1,2) + A^1(2,3)$

$= \min\{\infty, 5+15$

$= 20$

$A^2[1,4] = \min[A^1(1,4), A^1(1,2) + A^1(2,4)$

Similarly we obtain A^3 from A^2 which is minimum cost matrix obtained by using {1, 2, 3} as an intermidiate vertiey.

$$A^3 = \begin{bmatrix} 0 & 5 & 20 & 10 \\ 45 & 0 & 15 & 5 \\ 30 & 35 & 0 & 15 \\ 15 & 20 & 5 & 0 \end{bmatrix}$$

$A^3(2,1) = \min\{A^2(2,1), A^2(2,3) + A^2(3,1)$

$= \min\{50, 15+30 =$

$= 45$

Similarly

$$A^4 = \begin{bmatrix} 0 & 5 & 15 & 10 \\ 20 & 0 & 10 & 5 \\ 30 & 35 & 0 & 15 \\ 15 & 20 & 5 & 0 \end{bmatrix}$$

→ cost matrix for All pair Shortest paths by using {1, 2, 3, 4} as Intermidiate Vertices.

$A^4(1,3) = \min\{A^3(1,3), A^3(1,4) + A^3(4,3)$

$= \min\{20, 15\}$

$= 15$

Scanned by CamScanner