ALL-PAIR SHORTEST PATH ALGORITHM.

The all pair shortest paths algorithm is also known as Floyd-Warshall algorithm is used to find all pair shortest path problem from a given weighted graph. As a result of this algorithm, it will generate a matrix, which represent the minimum distance from any node to all other nodes in the graph.

At first the output matrix is same as given cost matrix of the graph. After that the output matrix will be updated with all vertices k as the intermediate vertex.

The time complexity of this algorithm is $O(V^3)$, where V is the number of vertices in the graph.

Algorithm

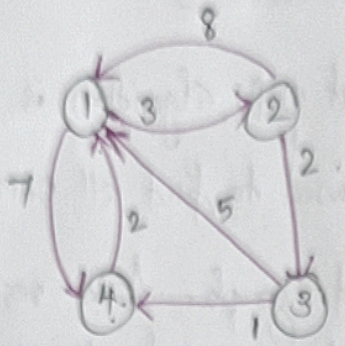
Floyd Warshal (cost)

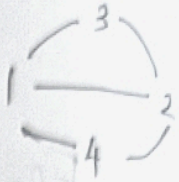
Input – The cost matrix of the given graphs.

Output – Matrix for shortest path between any vertex to any other vertex.

```
Begin
    for k:=0 to n, do
        for i:=0. to n, do
            for j:=0 to n, do
                if cost[i,k] + cost[k,j] < cost[i,j], then
                    cost[i,j] := cost[i,k] + cost[k,j]
            done
        done
    done
    display the current cost matrix
End.
```

Eg:





$$A^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & \infty \\ 5 & \infty & 0 & 1 \\ 2 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

$$A^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 7 \\ 8 & 0 & 2 & 15 \\ 5 & 8 & 0 & 1 \\ 2 & 5 & \infty & 0 \end{bmatrix} \end{array}$$

$A^0[2,3]$ $\quad A^0[2,1] + A^0[1,3]$

$2 \quad < \quad 8 + \infty$

$A^0[2,4]$ $\quad A^0[2,1] + A^0[1,4]$

$\infty \quad > \quad 8 + 7 = 15$

$A^0[3,2]$ $\quad A^0[3,1] + A^0[1,2]$

$\infty \quad > \quad 5 + 3 = 8$

$A^0[3,4]$ $\quad A^0[3,1] + A^0[1,4]$

$1 \quad < \quad 5 + 7 = 12$

$A^0[4,2]$ $\quad A^0[4,1] + A^0[1,2]$

$\infty \quad 2 + 3 = 5$

$A^0[4,3]$ $\quad A^0[4,1] + A^0[1,3]$

$\infty \quad 2 + \infty$

$$A^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & 5 & 7 \\ 8 & 0 & 2 & 15 \\ 5 & 8 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix} \end{array}$$

$$A^3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & 5 & 6 \\ 7 & 0 & 2 & 3 \\ 5 & 8 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix} \end{array}$$

$$A^4 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & 5 & 6 \\ 5 & 0 & 2 & 3 \\ 3 & 6 & 0 & 1 \\ 2 & 5 & 7 & 0 \end{bmatrix} \end{array}$$

Travelling Salesman Problem.

Problem Statement.

A traveler needs to visit all the cities from a list, where distances between all the cities are known and each city should be visited just once. What is the shortest possible route that he visits each city exactly once and returns to the origin city?

Solution:

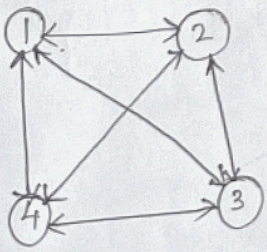
If we use brute-force approach to evaluate every possible tour and select the best one; For n number of vertices in a graph, there are $(n-1)!$ number of possibilities.

Instead of brute-force, using dynamic approach, the solution can be obtained in lesser time, though there is no polynomial time algorithm.

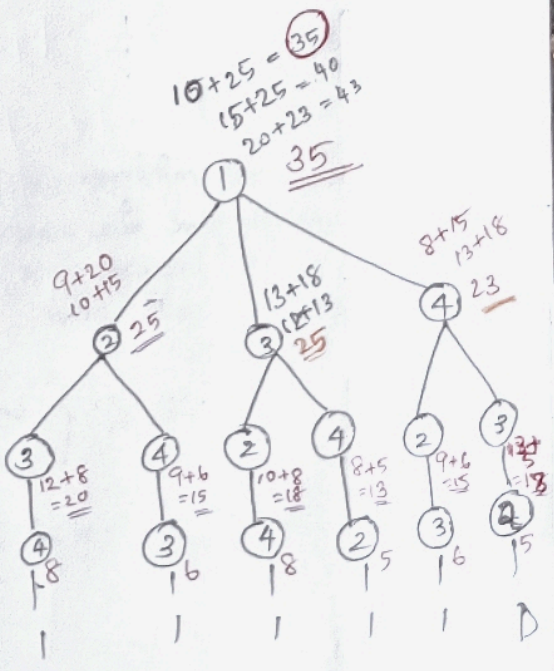
Eg:-

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

→ Cost adjacency matrix.



→



Formula. (For starting vertex 1)

$$g(1,\{2,3,4\}) = \min_{k \in \{2,3,4\}} \{ c_{1k} + g(k, \{2,3,4\} - \{k\}) \}.$$

$$\boxed{g(i,S) = \min_{k \in S} \{ c_{ik} + g(k, S - \{k\}) \}.}$$

	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

$g(1,\{2,3,4\}) = \min_{k \in \{2,3,4\}} \{ c_{1k} + g(k, \{2,3,4\} - \{k\}) \}$.