Radhakrishna MP-2 Tutorial-3 190031187

1. graph[][] = $\begin{bmatrix} 0 & 5 & \infty & 10 \\ \infty & 0 & 3 & \infty \\ \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & 0 \end{bmatrix}$

$$D_1 = \begin{bmatrix} 0 & 5 & \infty & 10 \\ \infty & 0 & 3 & \infty \\ \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & 0 \end{bmatrix}$$

$$D_2 = \begin{bmatrix} 0 & 5 & 8 & 10 \\ \infty & 0 & 3 & \infty \\ \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & 0 \end{bmatrix}$$

$$D_3 = \begin{bmatrix} 0 & 5 & 8 & 9 \\ \infty & 0 & 3 & 4 \\ \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & 0 \end{bmatrix}$$

$$D_4 = \begin{bmatrix} 0 & 5 & 8 & 9 \\ \infty & 0 & 3 & 4 \\ \infty & \infty & 0 & 1 \\ \infty & \infty & \infty & 0 \end{bmatrix}$$

2- graph[][] $= \begin{bmatrix} 0 & \infty & -2 & \infty \\ 4 & 0 & 3 & \infty \\ \infty & \infty & 0 & 2 \\ \infty & -1 & \infty & 0 \end{bmatrix}$

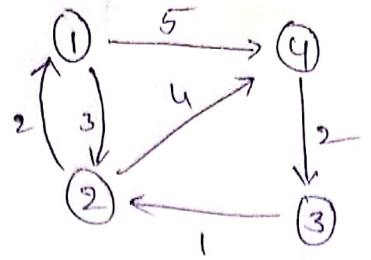
$$D_1 = \begin{array}{cc} & \begin{matrix} 1 & & & \end{matrix} \\ \begin{matrix} 1 \\ \\ \\ \end{matrix} & \begin{bmatrix} 0 & \infty & -2 & \infty \\ 4 & 0 & 2 & \infty \\ \infty & \infty & 0 & 2 \\ \infty & -1 & \infty & 0 \end{bmatrix} \end{array}$$

$$D_2 = \begin{array}{cc} & \begin{matrix} & 2 & & \end{matrix} \\ \begin{matrix} \\ 2 \\ \\ \end{matrix} & \begin{bmatrix} 0 & \infty & -2 & \infty \\ 4 & 0 & 2 & \infty \\ \infty & \infty & 0 & 2 \\ 3 & -1 & 2 & 0 \end{bmatrix} \end{array}$$

$$D_3 = \begin{array}{cc} & \begin{matrix} & & 3 & \end{matrix} \\ \begin{matrix} \\ \\ 3 \\ \end{matrix} & \begin{bmatrix} 0 & \infty & -2 & 0 \\ 4 & 0 & 2 & 4 \\ \infty & \infty & 0 & 2 \\ 3 & -1 & 2 & 0 \end{bmatrix} \end{array}$$

$$D_4 = \begin{array}{cc} & \begin{matrix} & & & 4 \end{matrix} \\ \begin{matrix} \\ \\ \\ 4 \end{matrix} & \begin{bmatrix} 0 & -1 & -2 & 0 \\ 4 & 0 & 2 & 4 \\ 5 & 1 & 0 & 2 \\ 3 & -1 & 2 & 0 \end{bmatrix} \end{array}$$

3.



$$\text{Graph} = \begin{bmatrix} 0 & 3 & \infty & 5 \\ 2 & 0 & \infty & 4 \\ \infty & 1 & 0 & \infty \\ \infty & \infty & 2 & 0 \end{bmatrix}$$

$$D_1 = \begin{array}{cc} & \begin{array}{cccc} 1 & & & \end{array} \\ 1 & \begin{bmatrix} 0 & 3 & \infty & 5 \\ 2 & 0 & \infty & 4 \\ \infty & 1 & 0 & \infty \\ \infty & \infty & 2 & 0 \end{bmatrix} \end{array}$$

$$D_2 = \begin{array}{cc} & \begin{array}{cccc} & 2 & & \end{array} \\ 2 & \begin{bmatrix} 0 & 3 & \infty & 5 \\ 2 & 0 & \infty & 4 \\ 3 & 1 & 0 & 5 \\ \infty & \infty & 2 & 0 \end{bmatrix} \end{array}$$

$$D_3 = \begin{array}{cc} & \begin{array}{cccc} & & 3 & \end{array} \\ 3 & \begin{bmatrix} 0 & 3 & \infty & 5 \\ 2 & 0 & \infty & 4 \\ 3 & 1 & 0 & 5 \\ 5 & 3 & 2 & 0 \end{bmatrix} \end{array}$$

$$D_4 = \begin{array}{cc} & \begin{array}{cccc} & & & 4 \end{array} \\ 4 & \begin{bmatrix} 0 & 3 & 7 & 5 \\ 2 & 0 & 6 & 4 \\ 3 & 1 & 0 & 5 \\ 5 & 3 & 2 & 0 \end{bmatrix} \end{array}$$

$\therefore$ Final Answer is

$$\begin{bmatrix} 0 & 3 & 7 & 5 \\ 2 & 0 & 6 & 4 \\ 3 & 1 & 0 & 5 \\ 5 & 3 & 2 & 0 \end{bmatrix}$$

NAME: NERELLA VENKATA RADHAKRISHNA

ID: 190031187

TUTORIAL-3

In [1]:

```
# Floyd-warshall algorithm

import sys
INF = sys.maxsize

def floydWarshall(graph):
    # number of vertices in the graph
    n = len(graph)

    # dist will be the output matrix that will have the shortest distances between every pair
    dist = [[] for i in range(n)]


    # Initialize the dist matrix as same as the input graph matrix.
    for i in range(n):
        for j in range(n):
            dist[i].append(graph[i][j])

    # Taking all vertices one by one and setting them as intermediate vertices
    for k in range(n):
        # Pick all vertices as source one by one.
        for i in range(n):
            # Pick all vertices as the destination for the above choosen source vertex.
            for j in range(n):
                # Update the value of dist[i][j] if k provides a shortest path from i to j
                dist[i][j] = min(dist[i][j],dist[i][k]+dist[k][j])

    # Shortest distance for every pair of vertex.
    print('Shortest Distance between every pair of vertex:-')
    for i in range(n):
        for j in range(n):
            if dist[i][j]==INF:
                print ("%7s" % ("INF"),end=' ')
            else:
                print ("%7s" % (dist[i][j]),end=' ')
        print()
```

In [2]:

```
graph = [[0,5,INF,10],[INF,0,3,INF],[INF,INF,0,1],[INF,INF,INF,0]]

floydWarshall(graph)
```

```
Shortest Distance between every pair of vertex:-
      0       5       8       9
    INF       0       3       4
    INF     INF       0       1
    INF     INF     INF       0
```

In [3]:

```
#Example-3

# Floyd Warshall Algorithm in python
# The number of vertices
nV = 4
INF = 999
# Algorithm implementation
def floyd_warshall(G):
    distance = list(map(lambda i: list(map(lambda j: j, i)), G))
    # Adding vertices individually
    for k in range(nV):
        for i in range(nV):
            for j in range(nV):
                distance[i][j] = min(distance[i][j], distance[i][k] + distance[k][j])
    print_solution(distance)
# Printing the solution
def print_solution(distance):
    for i in range(nV):
        for j in range(nV):
            if(distance[i][j] == INF):
                print("INF", end=" ")
            else:
                print(distance[i][j], end="  ")
        print(" ")
G = [[0, 3, INF, 5],
         [2, 0, INF, 4],
         [INF, 1, 0, INF],
         [INF, INF, 2, 0]]
floyd_warshall(G)
```

```
0  3  7  5
2  0  6  4
3  1  0  5
5  3  2  0
```

In [4]:

```
# Problem 2
graph = [[0,INF,-2,INF],[4,0,3,INF],[INF,INF,0,2],[INF,-1,INF,0]]

floydWarshall(graph)
```

```
Shortest Distance between every pair of vertex:-
      0      -1      -2       0
      4       0       2       4
      5       1       0       2
      3      -1       1       0
```