MD YASEEN AHMED
1BM19CS404

03. Implement Dijkstra's algorithm to find suitable path i.e., shortest path for a given topology.

```
import sys

class Graph:

    def __init__(self, vertices):
        self.v = vertices
        self.graph = [[0 for column in
          range(vertices)] for row in
          range(vertices)]

    def printSolution(self, dist):
        print("vertex \t Distance from
          source vertex")
        for node in range(self.v):
            print(node, "\t", dist[node])
```

```
def minDistance(self, dist, sptSet):
    min = sys.maxsize

    for v in range(self.v):
        if dist[v] < min and sptSet[v] ==
                                False:
            min = dist[v]
            min_index = v
    return min_index

def dijkstra(self, src):
    dist = [sys.maxsize] * self.v
    dist[src] = 0
    sptSet = [False] * self.v

    for cout in range(self.v):

        u = self.minDistance(dist, sptSet)
        sptSet[u] = True

    for v in range(self.v):
        if self.graph[u][v] > 0 and
sptSet[v] == False and dist[r] > dist[u]+
```

```
self.graph[u][v]:
    dist[v] = dist[u] + self.graph
                        [u][v]

self.printSolution(dist)


g = Graph(9)

g.graph = [ [0, 4, 0, 0, 0, 0, 0, 8, 0],
            [4, 0, 8, 0, 0, 0, 0, 11, 0]
            [0, 8, 0, 7, 0, 4, 0, 0, 2]
            [0, 0, 7, 0, 9, 14, 0, 0, 0]
            [0, 0, 0, 9, 0, 10, 0, 0, 0]
            [0, 0, 4, 14, 10, 0, 2, 0, 0]
            [0, 0, 0, 0, 0, 2, 0, 1, 6]
            [8, 11, 0, 0, 0, 0, 1, 0, 7]
            [0, 0, 2, 0, 0, 0, 6, 7, 0]
          ]

g.dijkstra(0)
```