CN lab
Harsh Shankar Rao
1BM18CS032

Dijikestra's algorithm

```
class Graph():

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

    def print_solution(self, dist):

        for node in range(self.V):
            print(node, "\t", dist[node])

    def min_distage(self, dist, sptSet):
        min = 9999
        for v in range(self.V):
            if dist[v] < min and sptSet[v] == False
                min = dist[v]
                min_index = v
        return min_index

    def add_edge(self, src, dist, weight):

        self.graph[src][dist] = self.graph[dist][src] = weight

    def dijkstra(self, src):
        dist = [9999] *self.v
        dist[src] = 0
        sptSet=[False] *self.v

    for cout in range (self.v):

            u = self.min-distance (dist, sptset)
            sptset[u] =True
            for v in range (self.v):
                if self.graph[u][v] >0 and sptSet[v]==False
                and dist[v] > dist[u] + self.graph[u][v]
                    dist[v]= dist[u] + self.graph[u][v]

    self.print (dist)

g = Graph(int(input()))

e= int (input())
for i in range(e):
    src,dist, cost = [int(_) for _ in input.split()]
    g.addedge (src,dist, cost)

src = int(input())
g.dijeketra(src)
```