CN LAB - 8

Abhiram G
1BM18CS127

Djikstras

```
class Graph ():
    def __init__(self, vertices):
        self.V = vertices
        self.graph = [[0 for column in range(vertices)]
        for row in range(vertices)]
    def print_solution(self, dist):
        print("Vertex \t Distance from source")
        for node in range(self.V):
            print(node, "\t", dist[node])
    def min_distance(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, dest, weight):
        self.graph[src][dest] = self.graph[dest][src]
                                = weight
    def djikstra(self, src)
        dist = [9999] * self.V
        dist[src] = 0
        sptSet = [false] * self.V
        for count in range(self.V):
            u = self.min_distance(dist, sptSet)
            sptSet[v] = TRUE
        for v in range(self.V):
            if self.graph[u][v] > 0
```

```
                sptSet[v] : : False and
        dist[v] > dist[u] + self.graph[u][v]:
        dist[v] : dist[u] + self.graph[u][v]
self.print_solution(dist)
g : Graph( int(input("Enter no. of nodes:")))
c : int(input("Enter no. of edges"));
for i in range(c):
    src, dest, cost = [int(_) for _ in input
        ("Enter [src] [Dist] [weight]: ")
            split(' ')]
    g.add_edge(src, dist, cost)
src = int(input("Enter [src] to fil cost"))
g.dijikstra(src)
```