Ajay Mittur
IBM18CS006

CN Lab

~~T~~ Dijkstras Algorithm

```
class Topology:
    def __init__(self, nodes):
        self.nodes = nodes
        self.graph = [[0 for column in range
                                    (nodes)]
                      for row in range(nodes)]

    def print_routing_table(self):
        print("Source \tDestination \t Distance")
        for node in range(self.nodes):
            print(f"{self.src} \t {node} \t\t
                              {self.dist[node]}")

    def min_distance_node(self, dist, visited):
        min_distance = 1000000
        for v in ~~self~~ range(self.nodes):
            if dist[v] < min_dist and not visited[v]
                min_distance = dist[v]
                _min_distace_node = v
        return _min_distance_node

    def add_direct_connection(self, src, dest, weight):
        self.graph[src][dest] = self.graph[dest][src]
        = weight

    def dijskra(self, src):
        self.dist = [100000] * self.nodes
        self.dist[src] = 0
        visited = [False] * self.nodes
        for _ in range(self.nodes):
            u = self.min_distance_node(self.dist, visited
            visited[u] = True
            for v in range(self.nodes):
                if self.graph[u][v] > 0 and not
                         visited[v]
```

Ajay Mittur
1BM18CS006

CN - lab

Dijkstra Algorithm

```
and self.dist[v] > self.dist[U] + self.graph[U][V]:
    Self.dist[V] = self.dist[U] + self.graph[U][V]

network = Topology(int(input("Enter number of nodes")))
edges = int(input("Enter number of edges: "))

for i in range(edges)
    network.add_direct_connection(src, dest, cost)

src = int(input("Enter src: "))
network.dijkstra(src)
network.print_routing_table()
```