ARAVINDH R
IBM18CS145

CN Lab 8

```
class Network():
    def __init__(self, N):
        self.matrix = []
        self.N = N

    def addlink(self, u, v, w):
        self.matrix.append((u, v, w))

    def table(self, dist, src):
        print("Table of {chr(ord('A')+src)}")
        print("Dest It Cost")
        for i in range(self.N):
            print(f"{chr(ord('A')+i)} It {dist[i]}")

    def algorithm(self, src):
        dist = [99] * self.N ; dist[src] = 0
        for _ in range(self.N-1):
            for u, v, w in self.matrix:
                if dist[u] != 99 and dist[u]+w < dist[v]:
                    dist[v] = dist[u] + w
        self.table(dist, src)
```

The above code initializes a matrix M, & n being the no. of nodes. Fills the matrix with inputed values. Using the algorithm finds all distances for all nodes & prints the table.

Arvind

ARAVINDH R
1BM18CS145

```
if __name__ == "__main__":
    matrix = []
    n = int(input("Enter no. of nodes"))
    print("Enter adjacency matrix")

    for _ in range(n):
        m = list(map(int, input().split(" ")))
        matrix.append(m)

    g = Network(n)
    for i in range(n):
        for j in range(n):
            if matrix[i][j] == 1:
                g.addlink(i, j, 1)

    for _ in range(n):
        g.algorithm(_)
```

A distance vector routing protocol determines the best route for data packets based on distance. The algorithm usually used to do this is Bellman-Ford algorithm. The router also informs topology changes periodically. This is done by each router maintaining a Distance Vector table, containing destination and cost for all destination nodes.