Date 23/11/20

Expt. No. ____ ABHAYA.V 1BM18CS001 Page No. 1

CN Lab

Q) Distance vector algorithm to find suitable path for transmission

```
class Graph:
    def __init__(self, vertices):
        self.V = vertices
        self.matrix = []
    def addedge(self, u, v, w):
        self.matrix.append((u, v, w))
    def printtable(self, dist, src):
        print("Vector table of {}".format(chr(ord('A')+src)))
        for i in range(self.V):
            print("{0}\t{1}".format(chr(ord('A')+i),
                dist[i]))

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

Teacher's Signature : ____

Date 23/11/20

Expt. No. _____ Page No. 2

```
def main():
    matrix = []
    print("Enter no. of Nodes: ")
    n = int(input())
    print("Enter the adjacency matrix : ")
    for i in range(n):
        m = list(map(int, input().split(" ")))
        matrix.append(m)
    g = Graph(n)
    for i in range(n):
        for j in range(n):
            if matrix[i][j] == 1
                g.addEdge(i,j,1)
    for _ in range(n):
        g.BellmanFord(_)
```