CN Lab

Dhanraj
IBM18CS02

Q) distance vector algorithm to find suitable Path for transmission.

#Graph

```
class Graph:
    def __init__(self, n):
        self.matrix = []
        self.n = n

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

    def print_table(self, dist, src):
        Print("Vector Table of {}".format(chr(
ord('A') + src)))
        for i in range(self.n):
            Print("{0}\t{1}".format(chr(ord(
                dist[i]))
```

#algorithm

```
def solve(self, src):
    dist = [99] * self.n
    dist[src] = 0
```

Dhanraj.K
1BM18CS027

```
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.printArr(dist, src).

def main():
    matrix = []
    print("Enter no of nodes:")
    n = int(input())
    print("Enter the Adjacency matrix:")
    for _ 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 i in range(n):
        g.BellmanFord(_)

main().
```