23/11/2020
Aniruddh N.S
1BM18CS015

CN Lab

Distance - Vector Algorithm to find suitable path for transmission

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

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

    def printArr(self, dist, src):
        print("Vector Table of {} ".format(chr
                                    (ord('A')
        for i in range(self.n):                 + src)))
            print("{0}\t{1}".format(chr(ord('A')+i),
                                    dist[i]))
    def bellmanFord(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[
                    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 i in range(n):
    m = list(map(int, input().split(" ")))
    matrix.append(m)
g = Graph(n)
for 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(i)
```