Keerthi. R
1BM19CS045

DATE PAGE

# distance Algorithm

```
class Network:
    def __init__(self, n):
        self.matrix = []
        self.n = n
    def addlink(self, u, v, w):
        self.matrix.append((u, v, w))

    def printable(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('A')+i, dist[i])

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

        path = []
        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.printable(dist, src)
```

Kenish
1BM18CS045

DATE: PAGE:

```
def main():
    matrix = []
    print("enter No. of Nodes")
    n = int(input())
    print("Enter Adjacency Matrix:")
    for i 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.algor()

main()
```