Distance Vector Algorithm (Python)

1BM18CS002
Abhijnya K G

```
class vector:
    def _init_(self, n):
        self.matrix = [ ]
        self.n = n

    def newedge(self, u, v, w):
        self.matrix.append([u, v, w])

    def display(self, dist, src):
        print " vector ~~or~~ routing table of {}"
                .format(chr(ord('A') + src)))
        for i in range(self, n):
            print ("{0}\t{1}", ~~format(chod~~
                .format(chr(ord('A' +i), dist[i])

    ~~def path~~
    def BellmanFord(self, src):
        dis = [99]* self.n
        dis [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 · ~~print~~ display ( dist , src )

matrix = [ ]
print (" Enter no of router")
n = int (input ())
print (" Enter adj matrix" )
for i in range (n) :
        g = list (map ( int, input () · split (" ")))
        matrix · append ( g )
  g = ~~graph~~ vector (n)
for i in range (n) :
    for j in range (n) :
        if matrix [i][j]==1 :
            g · newedge (i, j, 1)

    for _ in range (n) :
        g · Bellmanford (_)
```