Akanksha Laddha   Lab 7.

1BM18CS007.

Nov 23, 2020.

Date:
P. No:

write a program for 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 printGraph (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 pathCal (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.printGraph ( dist , src)
```

Scanned by TapScanner

Date:
P. No:

```
matrix = [ ]
print ( " Enter no. of nodes")
n = int (input ())
print (" enter adjacency matrix")
for i in range (n):
        g = list (map (int, input (). split (" ")))
        matrix. append (g)
x = Graph (n)
for i in range (n) :
      for j in range (n) :
              if matrix [i] [j] = = 1 :
                      x. addEdge (i, j, 1 )


for _ in range (n) :
      x. pathCal (_)
```

Akansha.