Distance Vectors Algorithm

[The n/w is represented as a Directed Graph]

Shikha-N
1BM18CS149

```
class Network:
    def __init__(self, n):
        self.mat = []
        self.n = n

    def add_edge(self, s, d, w):
        self.mat.append((s, d, w))

    def print_path(self, dist, src):
        print("Router table of {}".format(src))
        for i in range(self.n):
            print("{0}\t{1}".format(i, dist[i]))

    def DV_algo(self, src):
        dist = [99] * self.n
        dist[src] = 0
        for _ in range(self.n - 1):
            for s, d, w in self.mat:
                if dist[s] != 99 and dist[s] + w
                        < dist[d]:
                    dist[d] = dist[s] + w
        self.print_path(dist, src)

def main():
    matrix = []
    n = int(input("Enter No. of Routers: "))
    print("Enter the Adjacency matrix:")
    for i in range(n):
        row = list(map(int, input().split(" ")))
        matrix.append(row)
```


Shikha

```
g = Network (n)
for i in range(n):
    for j in range(n):
        if matrix[i][j] == 1:
            g.add_edge(i,j,1)

for i in range(n):
    g.DV_algo(i)
```

* Distance-vector routing protocol determines the best/shortest route for packets based on the no. of routers it has to pass (one router counts as one hop) using Bellman-Ford Algorithm which determines the minimum distance from source to destination node directly or by passing through intermediate nodes.

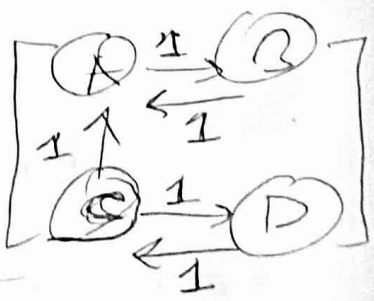
$$[\ D_{xy} = \min(D_{xy}, (c_{xa} + D_{ay}))\ ]$$

Sample case:

n = 4

matrix =

	A	B	C	D
A	0	1	99	99
B	1	0	99	99
C	1	99	0	1
D	99	99	1	0



Routing table for A

	A	B	C	D
A	0	1	99	99
B	1	0	99	99
C	1	2	0	1
D	2	3	1	0