COMPUTER NETWORKS LAB

VIVEK RAJEEV

Date __/__/__
Page ____
SPLASH

PROGRAM : 7A — DISTANCE VECTOR ALGORITHM

CODE

```
class Topology :

    def __init__(self, array_of_points):
        self.nodes = array_of_points
        self.edges = []

    def add_direct_connection (self, p1, p2, cost):
        self.edges.append((p1, p2, cost))
        self.edges.append((p2, p1, cost))

    def distance_vector_routing (self):
        import collections
        for node in self.nodes :
            dist = collections.defaultdict(int)
            next_hop = { node : node }
            for other_node in self.nodes :
                if other_node != node :
                    dist [other_node] = 100000000

            for i in range (len(self.nodes) - 1):
                for edge in self.edges :
                    src, dest, cost = edge
                    if dist[src] + cost < dist[dest]:
                        dist[dest] = dist[src] + cost
                        if src == node :
                            next_hop[dest] = dest
                        elif src in next_hop :
                            next_hop[dest] = next_hop[src]
            self.print_routing_table(node, dist, next_hop
            print()
```

1

Page
SPLASH

```
def print_routing_table (self, node, dist, next_hop):
    print(f'Routing table for {node}:')
    print('Dest \t Cost \t Next Hop')
    for dest, cost in dist.items():
        print(f'{dest} \t {cost} \t
                {next_hop [dest]} ')

def start (self):
    pass
```

PROGRAM 7B: DIJKSTRA'S ALGORITHM

CODE

```
import sys

class graph:

    def_init_(self, vertices):
        self.V = vertices
        self.graph= [[0 for column in
          range (vertices)] for row in range (vertices)]

    def printSolution(self, dist):
        print("Vertex \t Distance from source")
        for node in range (self.V):
            print (node, "\t", dist[node])

    def minDistance (self, dist, sptSet):
        min = sys.maxSize
        for v in range (self.V):
            if dist[v] < min and sptSet[v]=False:
                min = dist [v]
            min_index = V
    return min_index

def dijkstra (self, src) :
    dist = [sys.maxsize] * self.V
    dist [src] = 0
    sptSet = [False] * self.V
    for cout in range (self.V) :
        u = self. minDistance (dist, sptSet)
        sptSet [u] = True
        for v in range (self.V) :
            if self. graph [u][v] > 0 and
             sptSet [v] == False and
             dist [v] > dist [u] + self.graph[u][v]:
                dist [v] = dist [u] + self.graph[u][v]
    self. printSolution ~~(self.dist)~~(dist)
```