Page No.
Date

MD YASEEN AHMED
1BM19CS404

02 Write a program for distance vector algorithm to find suitable path for transmission.

```
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)
    nexthop ={node:node}
    for other_node in self.nodes:
        if other_node != node:
            dist[other_node]=10000000


    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()

def print_routing_table(self, node, dist,
                next_hop):
```

Page No.
Date

```
print(f' Routing table for{node} : ')
print (' Destination \t cost \t Next Hop')
for dest, cost in dest.items():
    print(f '{dest} \t\t {cost} \t \t
          {next_hop[dest]}')


nodes = ['A', 'B', 'C', 'D', 'E']

t = Topology(nodes)

t.add_direct_connection('A', 'B', 1)
t.add_direct_connection ('A', 'C', 5)
t.add_direct_connection ('B', 'C', 3)
t.add_direct_connection('B', 'E', 9)
t.add_direct_connection ('C', 'D', 4)
t.add_direct_connection ('D', 'E', 2)

t.distance_vector_routing()
```