

NAME: NIHA

USN: IBM18CS060

papergrid

Date: / /

## LAB-1

Write 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)  
        next_hop = {node: node}  
        for other_node in self.nodes:  
            if other_node != node:  
                dist[other_node] = 10000000  
                # infinity
```

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

```
nodes = input('Enter the nodes:').split()
```

```
t = Topology(nodes)
```

```
edges = int(input('Enter no. of connections:'))  
for _ in range(edges):  
    src, dest, cost = input('Enter [src]  
                             [dest] [cost]: ').split()  
    t.add_direct_connection(src, dest, int(cost))  
t.distance_vector_routing()
```