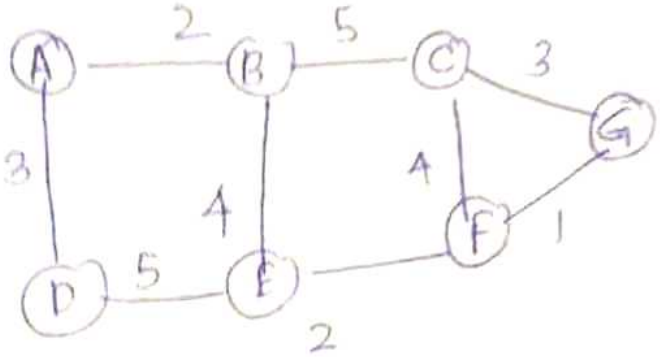
31/2/21

CN - Lab - 2

Neelesh Gangadhaya
1BM18CS058

1. Program : Choose the suitable link state routing algorithm for the following topology & show the following topology & Show the updates up to iteration 3 starting at Node F and implement the same



Program

```
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
```


Next

Nalish Gangopadhyay
1BM18CS058

Bellman Ford algorithm

```
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]

def print_routing_table(nodes, dist, next_hop)
    print()

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]}')

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

d = Topology(nodes)

d.add_direct_connection('A', 'B', 2)
d.add_direct_connection('A', 'D', 3)
d.add_direct_connection('B', 'C', 5)
d.add_direct_connection('B', 'E', 4)
d.add_direct_connection('C', 'G', 3)
d.add_direct_connection('C', 'F', 4)
d.add_direct_connection('D', 'E', 5)
```


Next

```
t. add_direct_connection('E', 'F', 2)
t. add_direct_connection('F', 'G', 1)

d. distance_vector_routing()
```