Distance Vector algorithm

Shravan Ram P.B
1BM18CS097

```
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 # infinity
# 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]
self.print_routing_table(node, dist, next_hop)
print()
def print_routing_table(self, node, dist, next_hop):
print("Routing table for {}".format(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']
t = Topology(nodes)
t.add_directly_connection('A', 'B', 1)
t.add_directly_connection('A', 'C', 5)
t.add_directly_connection('B', 'C', 3)
t.add_directly_connection('B', 'E', 9)
t.add_directly_connection('C', 'D', 4)
t.add_directly_connection('D', 'E', 2)
t.distance_vector_routing()
nodes = input('Enter the nodes:').split()
t.Topology(nodes)
edges = int(input('Enter the number 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()