TANYA MISHRA
16MI8CS117

Camlin Page
Date / /

Cycle 2 Lab 3

```
from collections import defaultdict

class graph():
    def __init__(self):
        self.edges = defaultdict(list)
        self.weights = {}

    def add_edge(self, from_node, to_node, weight):
        self.edges[from_node].append(to_node)
        self.edges[to_node].append(from_node)
        self.weights[(from_node, to_node)] = weight
        self.weights[(to_node, from_node)] = weight

def dijsktra(graph, initial, end):
    shortest_paths = {initial: (None, 0)}
    current_node = initial
    visited = set()
    while current_node != end:
        visited.add(current_node)
        destinations = graph.edges[current_node]
        weight_to_current_node =
                shortest_path[current_node][1]
        for next_node in destinations:
            weight = graph.weights[(current
            _node, next_node)] + weight_to_
                current_node
            if next_node not in shortest_paths:
                shortest_paths[next_node] =
                        (current_node, weight)
            else:
                current_shortest_weight =
                shortest_paths[next_node][1]
```

Scanned with CamScanner

TANYA MISHRA
1BM18CS117

Camlin Page
Date / /

```
if current_shortest_weight > weight:
    shortest_paths[next_node] =
        (current_node, weight)

next_destinations = {node: shortest_paths[node]
for node in shortest_paths if node
not in visited}

if not next_destinations:
    return "Route not possible"
current_node = min(next_destinations,
        key = lambda k: next_destination
                                [k][1])

path=[]

while current_node is not None:
    path.append(current_node)
    next_node = shortest_paths
                [current_node][0]
    current_node = next_node

path = path[::-1]
print("Shortest weight:", current_shortest_
                                weight)
print(path)
```

Scanned with CamScanner