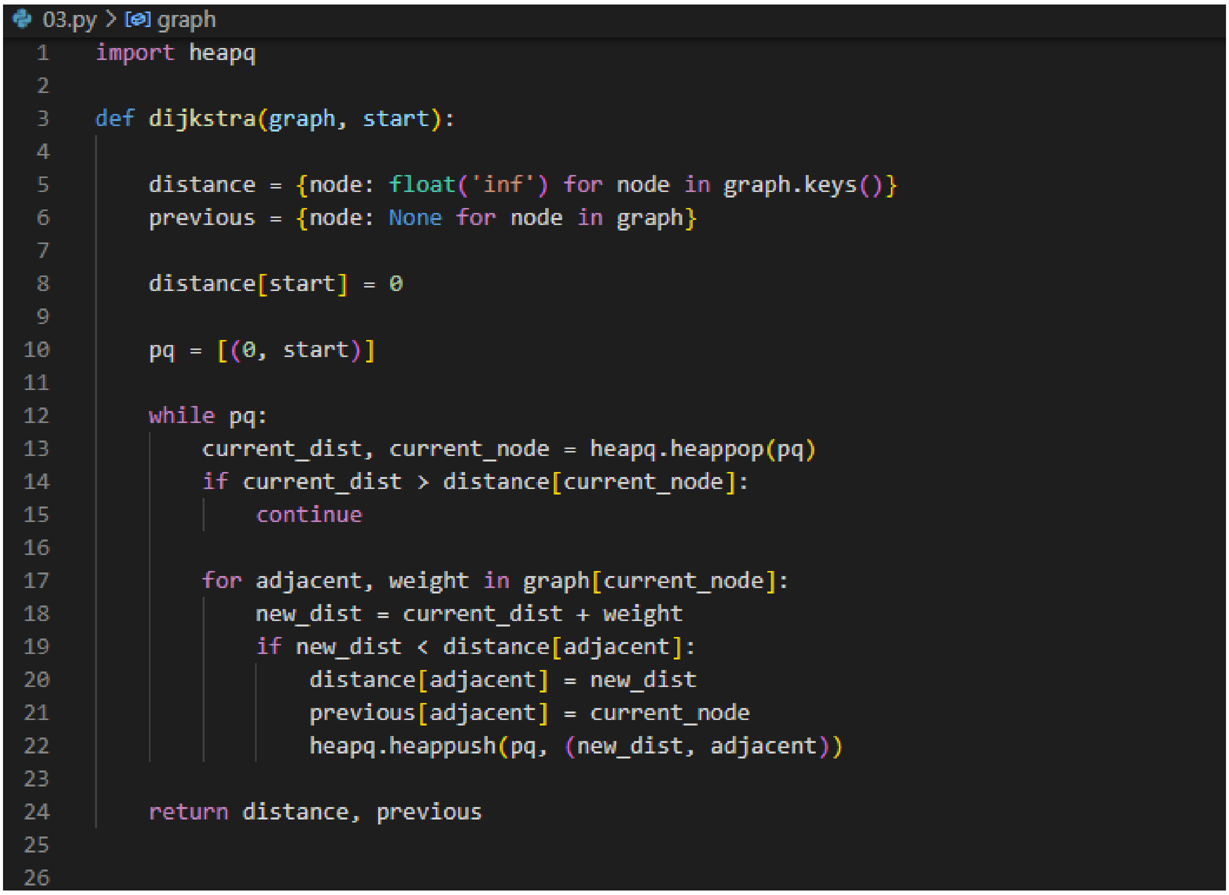
Aryan Sharma

Section – B3

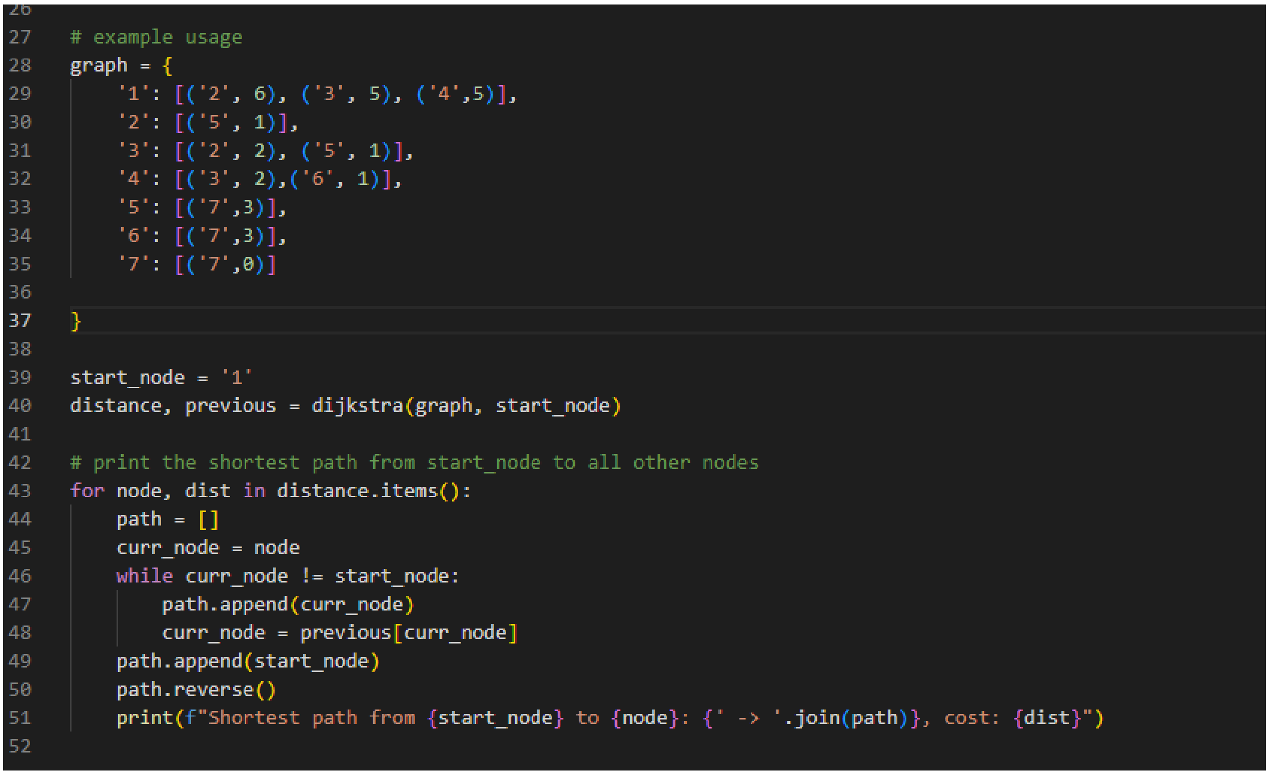
Assignment -09

Question – WAP Dijkstra algorithm

Code –

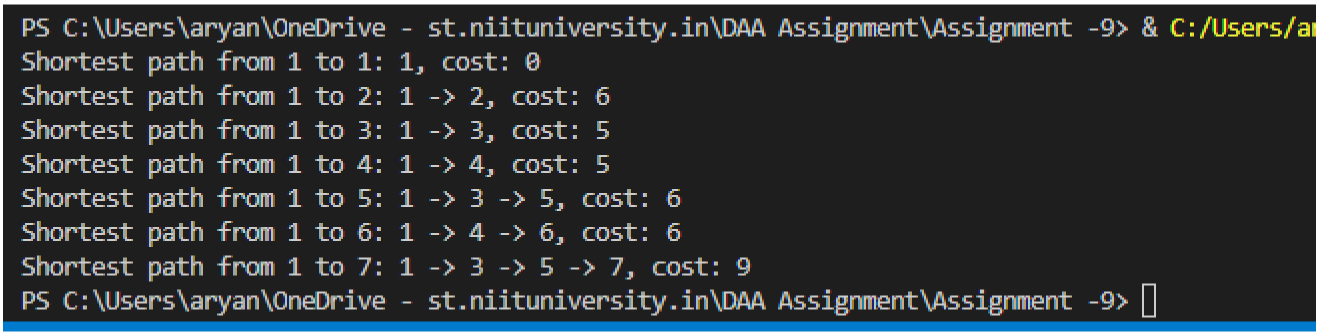


```python
import heapq

def dijkstra(graph, start):

    distance = {node: float('inf') for node in graph.keys()}
    previous = {node: None for node in graph}

    distance[start] = 0

    pq = [(0, start)]

    while pq:
        current_dist, current_node = heapq.heappop(pq)
        if current_dist > distance[current_node]:
            continue

        for adjacent, weight in graph[current_node]:
            new_dist = current_dist + weight
            if new_dist < distance[adjacent]:
                distance[adjacent] = new_dist
                previous[adjacent] = current_node
                heapq.heappush(pq, (new_dist, adjacent))

    return distance, previous


```



```python
# example usage
graph = {
    '1': [('2', 6), ('3', 5), ('4',5)],
    '2': [('5', 1)],
    '3': [('2', 2), ('5', 1)],
    '4': [('3', 2),('6', 1)],
    '5': [('7',3)],
    '6': [('7',3)],
    '7': [('7',0)]

}

start_node = '1'
distance, previous = dijkstra(graph, start_node)

# print the shortest path from start_node to all other nodes
for node, dist in distance.items():
    path = []
    curr_node = node
    while curr_node != start_node:
        path.append(curr_node)
        curr_node = previous[curr_node]
    path.append(start_node)
    path.reverse()
    print(f"Shortest path from {start_node} to {node}: {' -> '.join(path)}, cost: {dist}")

```

Output –



```
PS C:\Users\aryan\OneDrive - st.niituniversity.in\DAA Assignment\Assignment -9> & C:/Users/ar
Shortest path from 1 to 1: 1, cost: 0
Shortest path from 1 to 2: 1 -> 2, cost: 6
Shortest path from 1 to 3: 1 -> 3, cost: 5
Shortest path from 1 to 4: 1 -> 4, cost: 5
Shortest path from 1 to 5: 1 -> 3 -> 5, cost: 6
Shortest path from 1 to 6: 1 -> 4 -> 6, cost: 6
Shortest path from 1 to 7: 1 -> 3 -> 5 -> 7, cost: 9
PS C:\Users\aryan\OneDrive - st.niituniversity.in\DAA Assignment\Assignment -9> 
```

Analysis –

Time Complexity – O (E logV)

Space Complexity – O(V)

Where, E – no. of Edge, V – no. of Vertices