Worksheet 1.4

Student Name: Rydem
UID: 25MCI10027
Branch: MCA (AI & ML)
Section/Group: MAM-1 (A)
Semester: I
Date of Performance: 23/09/25
Subject Name: Design and Analysis of Algorithms Lab
Subject Code: 25CAP-612

1. Aim/Overview of the practical:

Experiment-1:
To implement Dijkstra's Algorithm in Python using a priority queue (min-heap) to find the shortest path from a given source node to all other nodes in a weighted, directed graph with non-negative edge weights.

2. Coding:

```
import heapq
def dijkstra(graph, start):
    dist = {node: float("inf") for node in graph}
    dist[start] = 0
    pq = [(0, start)]
    while pq:
        curr_dist, u = heapq.heappop(pq)
        if curr_dist > dist[u]:
            continue
        for v, weight in graph[u]:
            if dist[v] > dist[u] + weight:
                dist[v] = dist[u] + weight
                heapq.heappush(pq, (dist[v], v))
    return dist
```

```
# Example Graph (undirected, weighted):
graph = {
'P': [('Q', 4), ('R', 2), ('F', 9)],
'Q': [('P', 4), ('R', 5), ('S', 10)],
'R': [('P', 2), ('Q', 5), ('S', 3), ('E', 4)],
'S': [('Q', 10), ('R', 3), ('E', 1)],
'E': [('R', 4), ('S', 1)],
'F': [('P', 9)]
}
start_node = 'P'
distances = dijkstra(graph, start_node)
print("Shortest distances from node", start_node, ":")
for node, d in distances.items():
    print(f"{start_node} → {node} = {d}")
```

OUTPUT:-

```
PS E:\CLASS CODING> & C:/Users/Dell/AppData/Local/Programs/Python/Python310/python.exe "e:/CLASS CODING/DAA/dij.py"
Shortest distances from node P :
P → P = 0
P → Q = 4
P → R = 2
P → S = 5
P → E = 6
P → F = 9
```

3. Learning outcomes (What I have learnt):

- Understand the concept of shortest path in a weighted graph
- Learn how Dijkstra's Algorithm works step-by-step
- Implement priority queue using Python's heapq module
- Represent graphs using adjacency lists (dictionary format)
- Apply logic to update and track minimum distances
- Strengthen Python coding and algorithmic thinking skills