Experiment:3.1

Student Name: Virat Samdarshi **UID: 22BCS12648**
Branch: B.E CSE **Section/Group:IOT-627-B**
Semester:5th **DateofPerformance:16-10-24**
Subject Name:DAAlab **SubjectCode:22CSH-311**

1. **Aim:** Develop a program and analyze complexity to find shortest paths in a graph with positive edgeweights using Dijkstra‟s algorithm.

2. **Objective:**The objective of the program is to implement Dijkstra's algorithm to find the shortest paths from a source vertex to all other vertices in a graph with positive edge weights. The algorithm uses a greedy approach.

3. **Implementation/Code:**

```
#include <iostream>

#include <vector>

#include <queue>

#include <climits>

using namespace std;

struct Edge {

   int destination, weight;

};

void addEdge(vector<vector<Edge>>& graph, int src, int dest, int weight) {

   graph[src].push_back({dest, weight});

   graph[dest].push_back({src, weight});

}

void dijkstra(const vector<vector<Edge>>& graph, int src) {

   int V = graph.size();
```

```
vector<int> dist(V, INT_MAX);
dist[src] = 0;
priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;
pq.push({0, src});
while (!pq.empty()) {
    int u = pq.top().second;
    pq.pop();
    for (const auto& edge : graph[u]) {
        int v = edge.destination;
        int weight = edge.weight;
        if (dist[u] + weight < dist[v]) {
            dist[v] = dist[u] + weight;
            pq.push({dist[v], v});
        }
    }
}
cout << "Vertex\tDistance from Source " << src << endl;
for (int i = 0; i < V; ++i) {
    if (dist[i] == INT_MAX) {
        cout << i << "\t\t" << "INF" << endl;
    } else {
        cout << i << "\t\t" << dist[i] << endl;
    }
```

```
    }
}
int main() {
    int V, E;
    cout << "Enter the number of vertices: ";
    cin >> V;
    vector<vector<Edge>> graph(V);
    cout << "Enter the number of edges: ";
    cin >> E;
    cout << "Enter edges in the format (source destination weight):" << endl;
    for (int i = 0; i < E; ++i) {
        int src, dest, weight;
        cin >> src >> dest >> weight;
        addEdge(graph, src, dest, weight);
    }
    int source;
    cout << "Enter the source vertex: ";
    cin >> source;
    dijkstra(graph, source);
    return 0;
     }
```

4. **Output:**

```
Enter the number of vertices: 5
Enter the number of edges: 7
Enter edges in the format (source destination weight):
0 1 10
0 4 5
1 2 1
1 4 2
2 3 4
3 4 9
3 0 7
Enter the source vertex: 0
Vertex  Distance from Source 0
0               0
1               7
2               8
3               7
4               5
```

5. **Time Complexity:** O((V+E)logV)