Name	Parth Gandhi
UID No.	2021300033
Class & Division	S.E. COMPS A (BATCH B)
Experiment No.	6

Aim: To implement Single source shortest path using Djikstra’s algorithm.

Theory:
Single Source Shortest Path (SSSP) is a classic problem in computer science, which aims to find the shortest path from a source node to all other nodes in a given graph. Dijkstra's algorithm is a well-known algorithm for solving this problem. It is an efficient algorithm that uses a greedy approach to find the shortest path from the source node to all other nodes in the graph.
Dijkstra's algorithm works by starting at the source node and then iteratively expanding the frontier of visited nodes by selecting the unvisited node with the shortest distance from the source. The algorithm maintains a set of visited nodes, a set of unvisited nodes, and a set of tentative distances from the source node to each node in the graph. At each iteration, the algorithm selects the unvisited node with the smallest tentative distance from the source, and updates the tentative distances of its neighboring nodes.

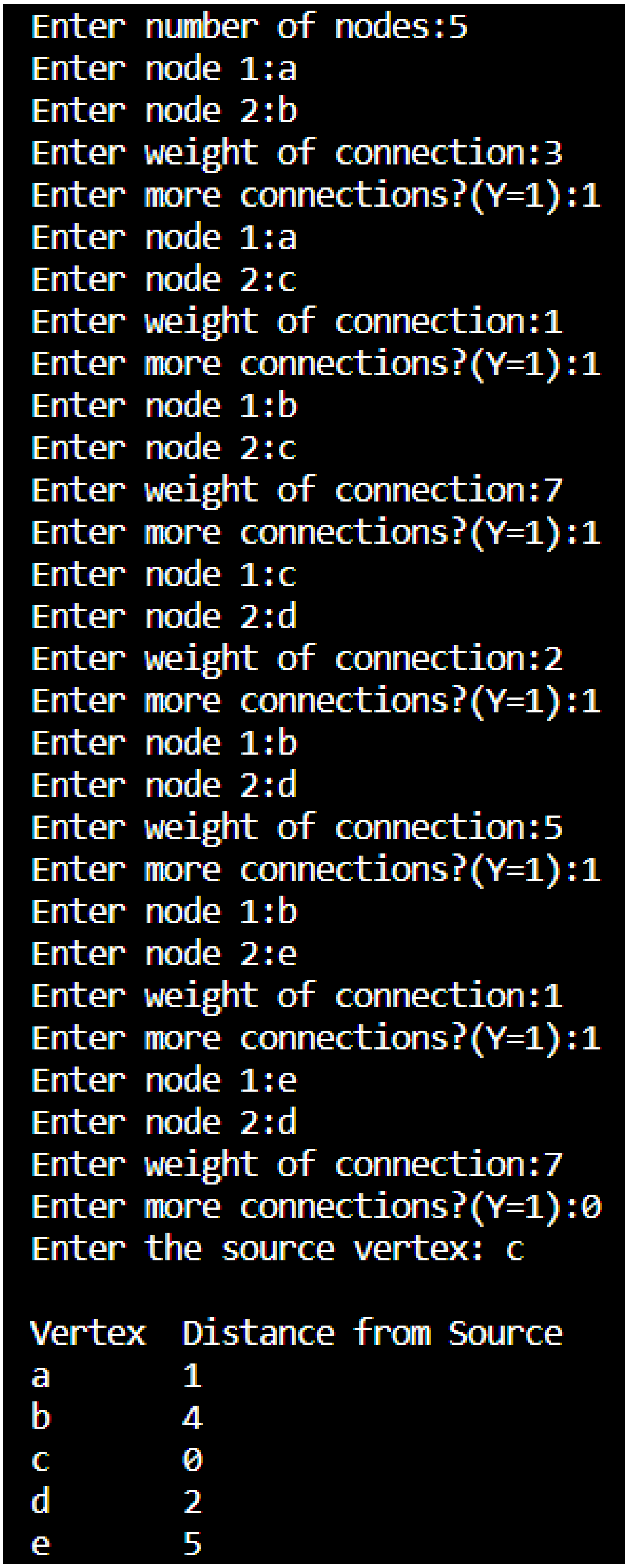
Algorithm:
1. Create a set of visited nodes, initially empty.
2. Create a set of unvisited nodes, initially containing all nodes in the graph.
3. Create a map of tentative distances from the source node to each node in the graph, initially set to infinity for all nodes except the source node, which is set to zero.
4. While the unvisited set is not empty: a. Select the unvisited node with the smallest tentative distance from the source, and mark it as visited. b. For each neighbour of the selected node that is still unvisited: i. Calculate the tentative distance from the source to the neighbour through the selected node. ii. If this tentative distance is smaller than the current tentative distance to the neighbour, update the tentative distance.
5. When all nodes have been visited, the map of tentative distances contains the shortest path from the source node to all other nodes in the graph.

Code:
```
#include <bits/stdc++.h>
using namespace std;

int minDistance(int dist[], bool sptSet[], int n){
    int min = INT_MAX, min_index;
    for (int v = 0; v < n; v++){
        if (sptSet[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;
    }
    return min_index;
}
```

```
int main(){
    int n;
    cout << "Enter number of node:";
    cin >> n;
    int graph[n][n];
    for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++)
            graph[i][j] = 0;
    }
    int ch = 1;
    while (ch == 1){
        char a, b;
        int weight;
        cout << "Enter node 1:";
        cin >> a;
        cout << "Enter node 2:";
        cin >> b;
        cout << "Enter weight of connection:";
        cin >> weight;
        graph[int(a) - 97][int(b) - 97] = weight;
        graph[int(b) - 97][int(a) - 97] = weight;
        cout << "Enter more connections?(Y=1):";
        cin >> ch;
    }
    char source;
    cout << "Enter the source vertex: ";
    cin >> source;
    int src = int(source) - 97;
    int dist[n];
    bool sptSet[n];
    for (int i = 0; i < n; i++){
        dist[i] = INT_MAX;
        sptSet[i] = false;
    }
    dist[src] = 0;
    for (int count = 0; count < n - 1; count++){
        int u = minDistance(dist, sptSet, n);
        sptSet[u] = true;
        for (int v = 0; v < n; v++){
            if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX && dist[u] + graph[u][v] < dist[v])
                dist[v] = dist[u] + graph[u][v];
        }
    }
    cout << "\nVertex\tDistance from Source" << endl;
    for (int i = 0; i < n; i++)
        cout << char(97 + i) << "\t" << dist[i] << endl;
   return 0;
}
```

Output:



```
Enter number of nodes:5
Enter node 1:a
Enter node 2:b
Enter weight of connection:3
Enter more connections?(Y=1):1
Enter node 1:a
Enter node 2:c
Enter weight of connection:1
Enter more connections?(Y=1):1
Enter node 1:b
Enter node 2:c
Enter weight of connection:7
Enter more connections?(Y=1):1
Enter node 1:c
Enter node 2:d
Enter weight of connection:2
Enter more connections?(Y=1):1
Enter node 1:b
Enter node 2:d
Enter weight of connection:5
Enter more connections?(Y=1):1
Enter node 1:b
Enter node 2:e
Enter weight of connection:1
Enter more connections?(Y=1):1
Enter node 1:e
Enter node 2:d
Enter weight of connection:7
Enter more connections?(Y=1):0
Enter the source vertex: c

Vertex  Distance from Source
a       1
b       4
c       0
d       2
e       5
```

Conclusion: Successfully wrote a program to implement Single source shortest path using Djikstra’s algorithm.