WORKSHEET 8

Student Name: Harsh Kumar **UID: 22BCS15754**
Branch: CSE **Section/Group: FL_IOT_603'B'**
Semester: 5th **Date of Performance:19/09/24**
Subject Name: Design and Analysis of Algorithms **Subject Code: 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. **Objectives:** Code and analyze to find shortest paths in a graph with positive edge weights using Dijkstra's.

3. **Algorithm:**

- Create a set sptSet (shortest path tree set) that keeps track of vertices included in the shortest- path tree, i.e., whose minimum distance from the source is calculated and finalized. Initially, this set is empty.
- Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign the distance value as 0 for the source vertex so that it is picked first.
- While sptSet doesn't include all vertices
- Pick a vertex u which is not there in sptSet and has a minimum distance value.
- Include u to sptSet.
- Then update distance value of all adjacent vertices of u.
- To update the distance values, iterate through all adjacent vertices.
- For every adjacent vertex v, if the sum of the distance value of u (fromsource) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

4. Implementation/Code:

```
#include <iostream>
using namespace std;
#include <limits.h>
#define V 9

int minDistance(int dist[], bool sptSet[])
{
 int min = INT_MAX, min_index;
 for (int v = 0; v < V; v++)
    if (sptSet[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;
 return min_index;
}
void printSolution(int dist[])
{
 cout << "Vertex \t Distance from Source" << endl;
 for (int i = 0; i < V; i++)
    cout << i << " \t\t\t\t" << dist[i] << endl;
}
void dijkstra(int graph[V][V], int src)
{
 int dist[V];
 bool sptSet[V];
 for (int i = 0; i < V; i++)
    dist[i] = INT_MAX, sptSet[i] = false;
 dist[src] = 0;
 for (int count = 0; count < V - 1; count++) {
    int u = minDistance(dist, sptSet);
    sptSet[u] = true;
    for (int v = 0; v < V; 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];
 }
 printSolution(dist);
}
int main()
{
 int graph[V][V] = { { 0, 4, 0, 0, 0, 0, 0, 8, 0 },
```

```
        { 4, 0, 8, 0, 0, 0, 0, 11, 0 },
        { 0, 8, 0, 7, 0, 4, 0, 0, 2 },
        { 0, 0, 7, 0, 9, 14, 0, 0, 0 },
        { 0, 0, 0, 9, 0, 10, 0, 0, 0 },
        { 0, 0, 4, 14, 10, 0, 2, 0, 0 },
        { 0, 0, 0, 0, 0, 2, 0, 1, 6 },
        { 8, 11, 0, 0, 0, 0, 1, 0, 7 },
        { 0, 0, 2, 0, 0, 0, 6, 7, 0 } };
    dijkstra(graph, 0);
    return 0;
    }
```

5. **Output:**

```
Vertex   Distance from Source
0                               0
1                               4
2                               12
3                               19
4                               21
5                               11
6                               9
7                               8
8                               14
```

6. **Time Complexity:**

$O(V^2)$. where 'V' is the number of vertices..

7. **Learning Outcome:**

1) Learnt how to use Dijkstra's Algorithm.
2) Learnt how to work with Adjacency Matrix.