Analysis and Design of Algorithms Lab

Assignment -7

Dated: 17-03-2021

Sub Code: CSE-228

Vivek Kumar Ahirwar

191112419

CSE - 3

Department of
Computer Science and Engineering

Under Guidance:
Prof. Manish Pandey

Maulana Azad

National Institute of Technology,

BHOPAL – 462 003 (INDIA)

Contents

Dijkstra’s shortest path algorithm

Code

```
// Keep Changing....@Vi

// Dijkstra's single source shortest path algorithm.

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

#define V 9

int minDistance(int dist[], bool sptSet[])
{
    // Initialize min value
    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[], int parent[])
{
    cout<<"Vertex \t Dist. from Source \t Parent\n";
    for (int i = 0; i < V; i++)
        cout << i << "\t\t" << dist[i] << "\t\t" << parent[i] << endl;
}

void dijkstra(int graph[V][V], int src)
{

    // dist[i] will hold the parent of i
    int parent[V];

    // dist[i] will hold the shortest distance from src to i
    int dist[V];

    // sptSet[i] will be true if vertex i is included in shortest path tree
    bool sptSet[V];

    // Initialize all distances as INT_MAX (infinite) and stpSet[] as false
    for (int i = 0; i < V; i++)
    {
        dist[i] = INT_MAX, sptSet[i] = false;
```

```
    }

    dist[src] = 0;
    parent[src] = -1;

    for (int count = 0; count < V - 1; count++)
    {
        // Pick the minimum distance vertex
        int u = minDistance(dist, sptSet);

        sptSet[u] = true;

        // Update dist value of the adjacent vertices
        for (int v = 0; v < V; v++)

            // Update dist[v] only if is not in sptSet
            if (!sptSet[v] && graph[u][v] && dist[u] != INT_MAX
                && dist[u] + graph[u][v] < dist[v])
            {
                dist[v] = dist[u] + graph[u][v];
                parent[v] = u;
            }
    }

    printSolution(dist, parent);
}

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;
}
```

Output

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL                                                         1: C/C+

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS F:\MANIT-Online class\Semester-4\CSE 228 ADA Lab> cd "f:\MANIT-Online class\Semester-4\CSE 228 ADA Lab\Lab-7 17-03-21"
PS F:\MANIT-Online class\Semester-4\CSE 228 ADA Lab\Lab-7 17-03-21> & .\"Dijkstra's shortest path algorithm.exe"
Vertex   Dist. from Source       Parent
0                0              -1
1                4               0
2                12              1
3                19              2
4                21              5
5                11              6
6                9               7
7                8               0
8                14              2
PS F:\MANIT-Online class\Semester-4\CSE 228 ADA Lab\Lab-7 17-03-21> 
```

Analysis

- Time Complexity of the implementation is *O(V^2)*.

- If the input graph is represented using adjacency list, it can be reduced to *O(E log V)* with the help of binary heap.

- Dijkstra's algorithm doesn't work for graphs with negative weight cycles, it may give correct results for a graph with negative edges. For graphs with negative weight edges and cycles, *Bellman–Ford algorithm* can be used.