Date: 10/03/2021

Practical 5:

2CSDE56 - Graph Theory

Name: Shrey Viradiya

Roll No: 18BCE259

Aim:

Write a program to implement single source shortest path algorithm.

Code:

Prac5_Djikstra.cpp

```cpp
// Write a program to implement single source shortest path algorithm.

#include <iostream>
#include "Djikstra.h"

int main(){
    using namespace std;

    int V, src;



    cout << "Enter the number of vertices: " << endl;
    cin >> V;

    cout << "Enter the adjecency matrix of the graph: " << endl;
    int **graphA = new int *[V];

    int error = 0;
    for (int i = 0; i < V; i++)
    {
        graphA[i] = new int[V];
        for (int j = 0; j < V; j++)
        {
            cin >> graphA[i][j];
            if (graphA[i][j] < 0) error = 1;
        }
    }

    if(error){
        cout << "Djikstra can not perform with negative weights! " << endl;
        // Releasing Memory
        for (int i = 0; i < V; i++)
        {
            delete [] graphA[i];
        }
        delete [] graphA;
        return 0;
    }

    cout << "Enter the Source Vertice: " << endl;
    do
    {
        cin >> src;
        if (src >= V){
            cout << "Invalid Source.. must be between 0 to" << (V-1) << endl;
```

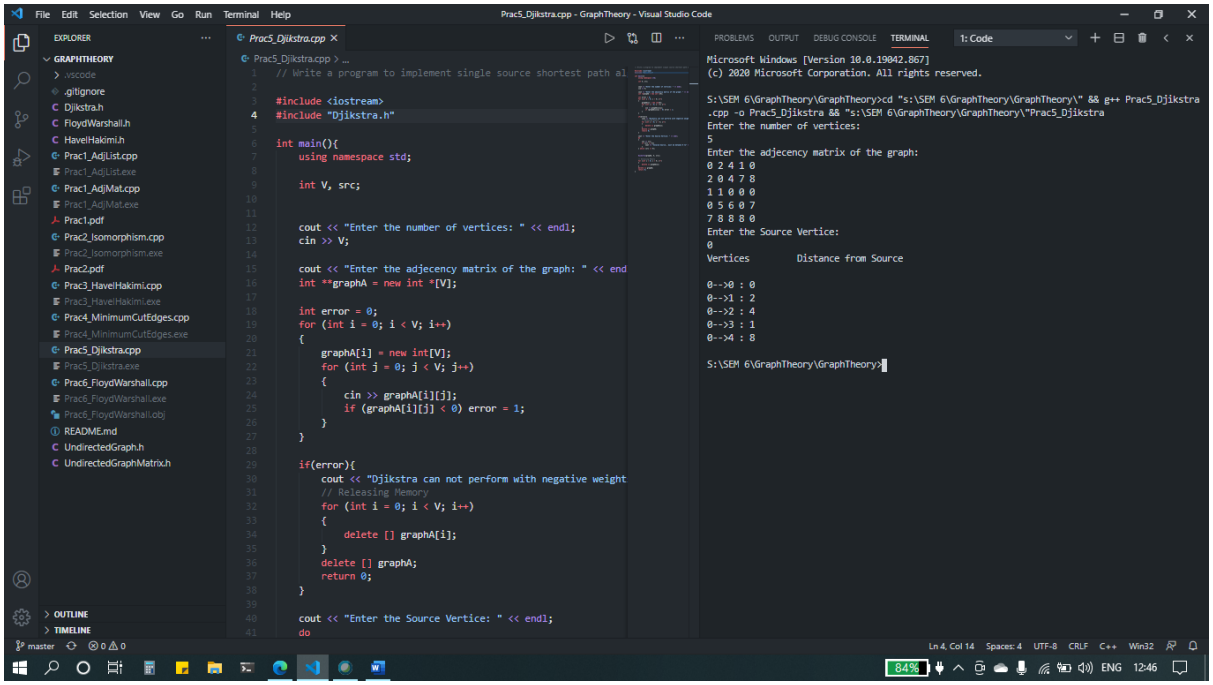
```
        }
    } while (src >= V);



    Djikstra(graphA, V, src);

    // Releasing Memory
    for (int i = 0; i < V; i++)
    {
        delete [] graphA[i];
    }
    delete [] graphA;
    return 0;
}
```

Djikstra.h

```
#pragma once

#include <limits.h>
#include <iostream>

void Djikstra(int **graph, int V, int src){
    int *dist = new int[V];
    bool *visited = new bool[V];

    for (int i = 0; i < V; i++)
        dist[i] = INT_MAX, visited[i] = false;

    dist[src] = 0;

    for (int count = 0; count < V - 1; count++)
    {
        int min = INT_MAX, min_index;

        // Finding Minimum Distance
        for (int v = 0; v < V; v++){
            if (visited[v] == false && dist[v] <= min)
                min = dist[v], min_index = v;
        }

        visited[min_index] = true;
        for (int v = 0; v < V; v++){
            if (!visited[v] && graph[min_index][v] && dist[min_index] != INT_MAX && dist[mi
n_index] + graph[min_index][v] < dist[v]){
                dist[v] = dist[min_index] + graph[min_index][v];
            }
```

```
        }
    }

    using namespace std;
    cout << "Vertices \t Distance from Source\n" << endl;
    for (int i = 0; i < V; i++)
        cout << src << "-->" << i << " : " << dist[i] << endl;
}
```

Snapshot of the output:



Conclusion:

Using Djikstra's Algorithm, we can find single source shortest path. Negative edges are not allowed.