Date: 17/03/2021

Practical 6:

2CSDE56 - Graph Theory

Name: Shrey Viradiya

Roll No: 18BCE259

Aim:

Write a program to implement all pairs shortest path algorithm

Code:

Prac6_FloydWarshall.cpp

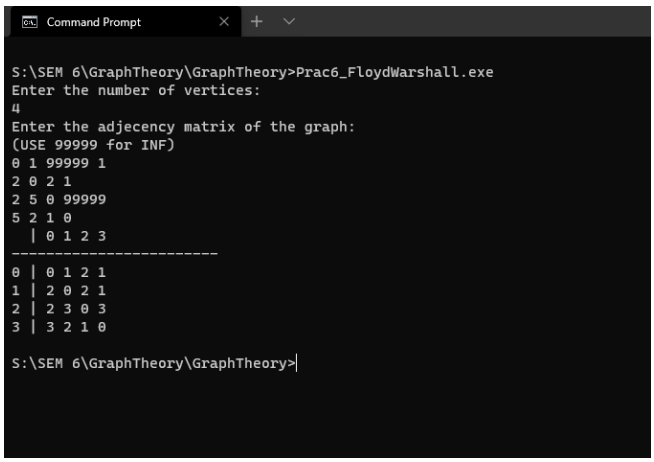
```cpp
// Write a program to implement All source shortest path algorithm.

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

int main(){
    using namespace std;

    int V, src, INF=99999;


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

    cout << "Enter the adjecency matrix of the graph: \n(USE 99999 for INF)" << endl;
    int **graphA = new int *[V];

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

    FloydWarshall(graphA, V);

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

FloydWarshall.h

```cpp
#pragma once

#include <iostream>

void FloydWarshall(int **graphA, int V){
```

```
    using namespace std;

    int **dist = new int *[V];

    // Copy the dist matrix

    for (int i = 0; i < V; i++)
    {
        dist[i] = new int[V];
        for (int j = 0; j < V; j++)
            dist[i][j] = graphA[i][j];
    }

    for (int k = 0; k < V; k++) // for all vertices update matrix by
        for (int i = 0; i < V; i++)     // picking the pair of source
            for (int j = 0; j < V; j++)     // and destination vertices
                if (dist[i][k] + dist[k][j] < dist[i][j]) dist[i][j] = dist[i][k] + dist[k]
[j]; // update is necessary


    cout << "Floyd Warshal" << endl;
    cout << "   | " ;
    for (int i = 0; i < V; i++) cout << i << " ";
    cout << endl;
    cout << "------------------------" << endl;
    for (int i = 0; i < V; i++){
        cout << i << " | " ;
        for (int j = 0; j < V; j++)  cout << dist[i][j] << " ";
        cout << endl;
    }
}
```

Snapshot of the output:



Conclusion:

With Floyd Warshall Algorithm, we can find all source shortest distance.