S. B. JAIN INSTITUTE OF TECHNOLOGY,

MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 6

Aim: Create a program that solves the all-pairs shortest path problem using the Floyd-Warshall's algorithm.

Name of Student: Shrutika Pradeep Bagdi

Roll No: CS22130

Semester/Year: V/III

Academic Session:2024-2025

Date of Performance:

Date of Submission:

AIM: Create a program that solves the all-pairs shortest path problem using the Floyd-Warshall's algorithm.

OBJECTIVE/EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

- To understand the concepts of all pairs shortest path problem.
- To implement the Floyd Warshall's algorithm for all pairs shortest path.

THEORY:

The Floyd-Warshall algorithm, named after its creators Robert Floyd and Stephen Warshall, is a fundamental algorithm in computer science and graph theory. It is used to find the shortest paths between all pairs of nodes in a weighted graph. This algorithm is highly efficient and can handle graphs with both positive and negative edge weights, making it a versatile tool for solving a wide range of network and connectivity problems.

The **Floyd Warshall Algorithm** is an all pair shortest path algorithm unlike Dijkstra and Bellman Ford which are single source shortest path algorithms. This algorithm works for both the **directed** and **undirected weighted** graphs. It does not work for the graphs with negative cycles (where the sum of the edges in a cycle is negative). It follows Dynamic Programming approach to check every possible path going via every possible node in order to calculate shortest distance between every pair of nodes.

PSEUDO CODE:

For k = 0 to n – 1

For i = 0 to n – 1

For j = 0 to n – 1

Distance[i, j] = min(Distance[i, j], Distance[i, k] + Distance[k, j])

where i = source Node, j = Destination Node, k = Intermediate Node

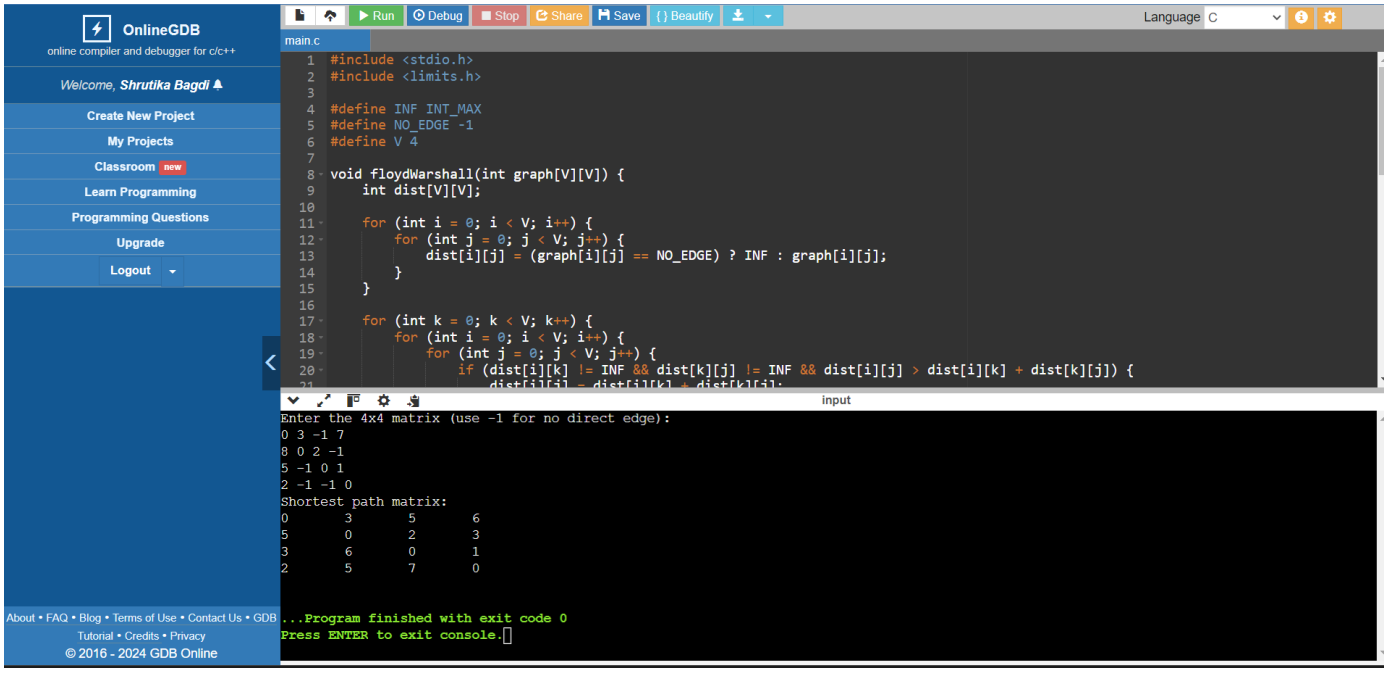
CODE:

```
#include <stdio.h>
#include <limits.h>
#define INF INT_MAX
#define NO_EDGE -1
```

```
#define V 4
void floydWarshall(int graph[V][V]) {
    int dist[V][V];
    for (int i = 0; i < V; i++) {
        for (int j = 0; j < V; j++) {
            dist[i][j] = (graph[i][j] == NO_EDGE) ? INF : graph[i][j];
        }
    }
    for (int k = 0; k < V; k++) {
        for (int i = 0; i < V; i++) {
            for (int j = 0; j < V; j++) {
                if (dist[i][k] != INF && dist[k][j] != INF && dist[i][j] > dist[i][k] + dist[k][j]) {
                    dist[i][j] = dist[i][k] + dist[k][j];
                }
            }
        }
    }
    printf("Shortest path matrix:\n");
    for (int i = 0; i < V; i++) {
        for (int j = 0; j < V; j++) {
            if (dist[i][j] == INF) {
                printf("INF\t");
            } else {
                printf("%d\t", dist[i][j]);
            }
        }
        printf("\n");
    }
}
int main() {
    int graph[V][V];
```

```
    printf("Enter the 4x4 matrix (use -1 for no direct edge):\n");
    for (int i = 0; i < V; i++) {
        for (int j = 0; j < V; j++) {
            scanf("%d", &graph[i][j]);
            if (graph[i][j] < 0 && graph[i][j] != NO_EDGE) {
                printf("Invalid input. Use -1 for no direct edge.\n");
                return 1;
            }
        }
    }
    floydWarshall(graph);
    return 0;
}
```

INPUT & OUTPUT WITH DIFFERENT TEST CASES:



CONCLUSION:

DISCUSSION AND VIVA VOCE:

- Explain the all-pairs shortest path problem using Floyd Warshall's algorithm.
- Discuss the complexity of Floyd Warshall's algorithm.
- What are the application of Floyd Warshall's algorithm?
- Can Floyd Warshall's algorithm detect negative cycles?

REFERENCES:

- *https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/*
- *https://www.javatpoint.com/floyd-warshall-algorithm*
- *https://www.programiz.com/dsa/floyd-warshall-algorithm*
- *https://www.tutorialspoint.com/data_structures_algorithms/floyd_warshall_algorithm.htm*