**NAME:** Aditya Jethani

**ROLL NO:** 21BCP166

**DIV-**3 **G-**5

**DAA assignment 7**

**AIM:**

Implement the Floyd WarshallAlgorithm for All Pair Shortest Path Problem.You are given a weighted diagraph 𝐺=(𝑉,𝐸), with arbitrary edge weights or costs 𝒄𝒗𝒘between any node 𝒗and node 𝒘.Findthe cheapest path from every node to every other node. Edges may have negative weights.Consider the following test caseto check your algorithm:

Edge (0, 1) has weight -1

Edge (0, 2) has weight 4

Edge (1, 2) has weight 3

Edge (1, 3) has weight 2

Edge (1, 4) has weight 2

Edge (3, 2) has weight 5

Edge (4, 2) has weight -3

**THEORY:**

The Floyd-Warshall algorithm is an algorithm for finding the shortest paths between all pairs of vertices in a weighted graph, with positive or negative edge weights (but with no negative cycles). The algorithm has a time complexity of O(V^3), where V is the number of vertices in the graph.

The algorithm maintains a 2D array dist[][] where dist[i][j] is the shortest path from vertex i to vertex j. Initially, dist[][] is initialized to the weights of the edges in the graph. Then, the algorithm considers all pairs of vertices (i, j) and all intermediate vertices k, and checks if going from i to k and then from k to j produces a shorter path than the current shortest path between i and j. If so, it updates the dist[i][j] to the shorter path.

**CODE:**

#include<bits/stdc++.h>

using namespace std;

const int INF = 1e9;

int main() {

    int n, m;

    cout << "Enter the number of vertices and edges: ";

    cin >> n >> m;

    vector<vector<int>> dist(n, vector<int>(n, INF));

    // initialize the distance matrix

    cout << "Enter the edges and their weights: " << endl;

    for(int i=0; i<m; i++) {

        int a, b, w;

        cin >> a >> b >> w;

        dist[a][b] = w;

    }

    for(int i=0; i<n; i++) {

        dist[i][i] = 0;

    }

    // apply Floyd Warshall Algorithm

    for(int k=0; k<n; k++) {

        for(int i=0; i<n; i++) {

            for(int j=0; j<n; j++) {

                if(dist[i][k] < INF && dist[k][j] < INF) {

                    dist[i][j] = min(dist[i][j], dist[i][k] + dist[k][j]);

                }

            }

        }

    }

    // print the result

    cout << "The all pair shortest path matrix is: " << endl;

    for(int i=0; i<n; i++) {

        for(int j=0; j<n; j++) {

            if(dist[i][j] == INF) {

                cout << "INF ";

            } else {

                cout << dist[i][j] << " ";

            }

        }

        cout << endl;

    }

    return 0;

}

**OUTPUT:**

**Text

Description automatically generated**

**ANALYSIS:**

The Floyd-Warshall algorithm has a time complexity of O(V^3), where V is the number of vertices in the graph. This time complexity arises from the fact that the algorithm uses three nested loops to consider all pairs of vertices and all possible intermediate vertices.

In terms of space complexity, the algorithm requires a 2D array of size V^2 to store the distances between all pairs of vertices. Therefore, the space complexity of the algorithm is O(V^2).