## Meet Jain, 23BCP093

**DAA Lab 7: Floyd Warshall (Dynamic)**

**Theory:**

The Floyd Warshall Algorithm is an all-pair shortest path algorithm that uses [Dynamic Programming](https://www.geeksforgeeks.org/introduction-to-dynamic-programming-data-structures-and-algorithm-tutorials/) to find the shortest distances between every pair of vertices in a graph, unlike [Dijkstra](https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/) and [Bellman-Ford](https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/) which are single source shortest path algorithms. This algorithm works for both the directed and undirected weighted graphs and can handle graphs with both positive and negative weight edges.  
  
Note: It does not work for the graphs with negative cycles (where the sum of the edges in a cycle is negative).

**Code:**

#include <bits/stdc++.h>

using namespace std;

#define INF INT\_MAX

void floydWarshall(vector<vector<int>>& graph, int V) {

    vector<vector<int>> dist = graph;

    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][k] + dist[k][j] < dist[i][j]) {

                    dist[i][j] = dist[i][k] + dist[k][j];

                }

            }

        }

    }

    cout << "Shortest distances between every pair of vertices:" << endl;

    for (int i = 0; i < V; i++) {

        for (int j = 0; j < V; j++) {

            if (dist[i][j] == INF) {

                cout << "INF ";

            } else {

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

            }

        }

        cout << endl;

    }

}

int main() {

    int V = 4;

    vector<vector<int>> graph = {

        {0, 3, INF, INF},

        {2, 0, INF, 1},

        {INF, 3, 0, 2},

        {INF, INF, 4, 0}

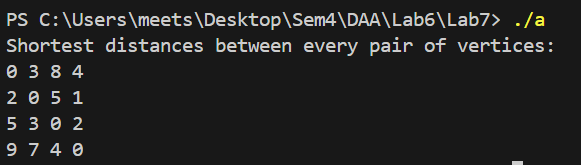
    };

    floydWarshall(graph, V);

    return 0;

}

**Output:**

****

**Time Complexity:**

* **Time Complexity**: O(V3)
  + Three nested loops each iterate over all vertices (V), making the overall complexity cubic.

**Space Complexity:**

* **Space Complexity**: O(V2)
  + The algorithm uses a 2D distance matrix to store the shortest paths between all pairs of vertices.