**Name**: Ravi Jamanbhai Makwana

**Roll No**: 21BCP418

Div-6, G-12

**Assignment-7**

**Aim:**

Write a program to perform Floyd Warshall using Dynamic Programming.

**Theory:**

• The Floyd Warshall Algorithm is for solving all pairs of shortest-path problems. The problem is to find the shortest distances between every pair of vertices in a given edge-weighted directed Graph.

• It is an algorithm for finding the shortest path between all the pairs of vertices in a weighted graph. This algorithm follows the dynamic programming approach to find the shortest path.

• A C-function for a N x N graph is given below. The function stores the all-pair shortest path in the matrix cost [N][N]. The cost matrix of the given graph is available in cost Mat [N][N].

**Program:**

#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

#define INF INT\_MAX

#define V 4 // number of vertices

void printSolution(int dist[][V]) {

    printf("Shortest distances between all pairs of vertices:\n");

    for (int i = 0; i < V; i++) {

        for (int j = 0; j < V; j++) {

            if (dist[i][j] == INF) {

                printf("INF ");

            }

            else{

                printf("%d   ", dist[i][j]);

            }

        }

        printf("\n");

    }

}

void floydWarshall(int graph[][V]) {

    int dist[V][V], i, j, k;

    for (i = 0; i < V; i++) {

        for (j = 0; j < V; j++) {

            dist[i][j] = graph[i][j];

        }

    }

    for (k = 0; k < V; k++) {

        for (i = 0; i < V; i++) {

            for (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];

                }

            }

        }

    }

    printSolution(dist);

}

int main() {

    int graph[V][V], i, j;

    printf("Enter the graph in the form of an adjacency matrix of 4\*4:\n");

    for (i = 0; i < V; i++) {

        for (j = 0; j < V; j++)  {

            scanf("%d", &graph[i][j]);

            if (graph[i][j] == -1) { // -1 indicates absence of edge between vertices i and j

                graph[i][j] = INF;

            }

        }

        printf("\n");

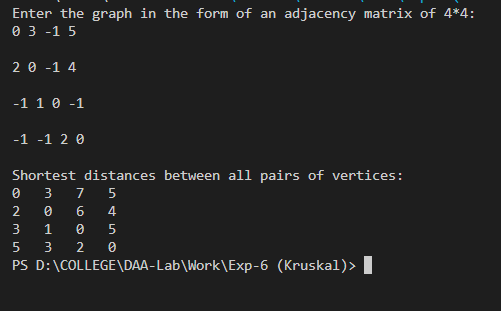
    }

    floydWarshall(graph);

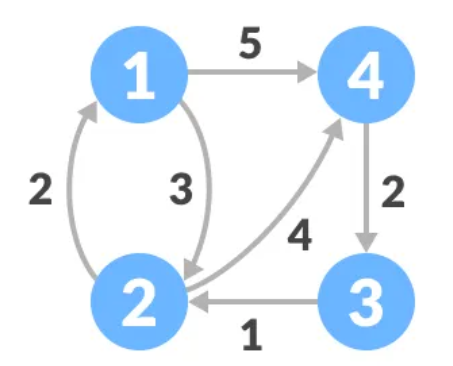
    return 0;

}

**Output:**



The above output is for the following graph:



**Analysis:**

**Time Complexity Analysis:**

• The time complexity of Floyd Warshall algorithm is O(V3), where V is the number of vertices in the graph.

• This is because the algorithm considers all pairs of vertices (i, j) and all possible intermediate vertices k. For each pair (i, j) and intermediate vertex k, it checks if the distance between i and k plus the distance between k and j is less than the current distance between i and j. If it is, the distance between i and j is updated. This process is repeated for all pairs (i, j) and all intermediate vertices k.

• Since there are V2 pairs of vertices and V possible intermediate vertices, the total number of operations performed by the algorithm is V3. Therefore, the time complexity of Floyd Warshall algorithm is O(V3).

• Note that in practice, the algorithm is not very efficient for very large graphs due to its high time complexity. However, it is a useful algorithm for finding the shortest path between all pairs of vertices in a dense graph, where other algorithms like Dijkstra's algorithm may be inefficient.