| S.No: 10 | Exp. Name: ***Floyd - Warshall's Algorithm*** | Date:2025-09-24 |
|---|---|---|



ID: 24070521106

## Aim:

Implement the Floyd-Warshall algorithm in C for finding the shortest distances between all pairs of vertices in a weighted directed graph. Prompt the user to input the number of vertices (N) and edges (E), and then accept edge information (source, destination, and weight) to build the adjacency matrix.

## Source Code:

Warshall.c

```
#include <stdio.h>
#define INF 99999
#define MAX_N 20 // Maximum value for N

#include <stdio.h>

#define MAX 100         // Max number of vertices

int main() {
    int n, e;
    int dist[MAX][MAX];

    // Use the INF constant already defined by Codetantra (INF = 99999)

    // Input number of vertices
    printf("Enter the number of vertices : ");
    scanf("%d", &n);

    // Input number of edges
    printf("Enter the number of edges : ");
    scanf("%d", &e);

    // Initialize the distance matrix
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (i == j)
                dist[i][j] = 0;
            else
                dist[i][j] = INF;
        }
    }

    // Read edges
    for (int i = 0; i < e; i++) {
        int u, v, w;
        printf("Enter source : ");
        scanf("%d", &u);
        printf("Enter destination : ");
        scanf("%d", &v);
        printf("Enter weight : ");
        scanf("%d", &w);
        dist[u - 1][v - 1] = w;  // 1-based to 0-based index
    }
```

```
    // 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][k] + dist[k][j] < dist[i][j]) {
                    dist[i][j] = dist[i][k] + dist[k][j];
                }
            }
        }
    }

    // Output the distance matrix
    printf("The following matrix shows the shortest distances between all pairs of th
e vertices.\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (dist[i][j] == INF)
                printf("%5s", "INF");
            else
                printf("%5d", dist[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

## Execution Results - All test cases have succeeded!

| Test Case - 1 |
|---|
| User Output |
| Enter the number of vertices : 4 |
| Enter the number of edges : 5 |
| Enter source : 1 |
| Enter destination : 2 |
| Enter weight : 4 |
| Enter source : 1 |
| Enter destination : 4 |
| Enter weight : 10 |
| Enter source : 1 |
| Enter destination : 3 |
| Enter weight : 6 |
| Enter source : 2 |
| Enter destination : 4 |
| Enter weight : 5 |
| Enter source : 3 |
| Enter destination : 4 |
| Enter weight : 2 |
| The following matrix shows the shortest distances between all pairs of the vertices. |
| 0 4 6 8 |
| INF 0 INF 5 |


ID: 24070521106

```
  INF  INF    0    2
  INF  INF  INF    0
```

| Test Case - 2 |
|---|
| User Output |
| Enter the number of vertices : 5 |
| Enter the number of edges : 6 |
| Enter source : 1 |
| Enter destination : 2 |
| Enter weight : 2 |
| Enter source : 1 |
| Enter destination : 5 |
| Enter weight : 3 |
| Enter source : 2 |
| Enter destination : 4 |
| Enter weight : 4 |
| Enter source : 2 |
| Enter destination : 3 |
| Enter weight : 7 |
| Enter source : 4 |
| Enter destination : 3 |
| Enter weight : 2 |
| Enter source : 5 |
| Enter destination : 4 |
| Enter weight : 1 |
| The following matrix shows the shortest distances between all pairs of the vertices. |
| 0 2 6 4 3 |
| INF 0 6 4 INF |
| INF INF 0 INF INF |
| INF INF 2 0 INF |
| INF INF 3 1 0 |

| Test Case - 3 |
|---|
| User Output |
| Enter the number of vertices : 4 |
| Enter the number of edges : 5 |
| Enter source : 1 |
| Enter destination : 2 |
| Enter weight : 4 |
| Enter source : 3 |
| Enter destination : 2 |
| Enter weight : 5 |
| Enter source : 4 |
| Enter destination : 1 |
| Enter weight : 1 |
| Enter source : 4 |
| Enter destination : 2 |
| Enter weight : 3 |
| Enter source : 4 |
| Enter destination : 3 |
| Enter weight : 8 |

| The following matrix shows the shortest distances between all pairs of the vertices. |
|---|
| 0 4 INF INF |
| INF 0 INF INF |
| INF 5 0 INF |
| 1 3 8 0 |

| Test Case - 4 |
|---|
| User Output |
| Enter the number of vertices : 4 |
| Enter the number of edges : 6 |
| Enter source : 1 |
| Enter destination : 2 |
| Enter weight : 1 |
| Enter source : 1 |
| Enter destination : 4 |
| Enter weight : 3 |
| Enter source : 2 |
| Enter destination : 3 |
| Enter weight : 6 |
| Enter source : 3 |
| Enter destination : 1 |
| Enter weight : -2 |
| Enter source : 4 |
| Enter destination : 2 |
| Enter weight : 5 |
| Enter source : 4 |
| Enter destination : 3 |
| Enter weight : 10 |
| The following matrix shows the shortest distances between all pairs of the vertices. |
| 0 1 7 3 |
| 4 0 6 7 |
| -2 -1 0 1 |
| 8 5 10 0 |


ID: 24070521106