**Experiment 5:**

**Implement Dijkstra's shortest path algorithm through a graph.**

#include <stdio.h>

#define INFINITY 9999

#define MAX 20

// Function to find the minimum of two values

int minimum(int a, int b) {

return (a <= b) ? a : b;

}

int main() {

int i, j, k, n, start, end, adj[MAX][MAX], path[MAX][MAX];

// Input number of vertices

printf("Enter number of vertices: ");

scanf("%d", &n);

// Input adjacency matrix

printf("Enter weighted matrix (0 for no path):\n");

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

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

// Initialize path matrix

path[i][j] = (i != j && adj[i][j] == 0) ? INFINITY : adj[i][j];

}

}

// Floyd-Warshall algorithm

for (k = 0; k < n; k++) {

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

if (path[i][k] != INFINITY && path[k][j] != INFINITY) {

path[i][j] = minimum(path[i][j], path[i][k] + path[k][j]);

}

}

}

}

// Print shortest path matrix

printf("\nShortest path matrix:\n");

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

if (path[i][j] == INFINITY)

printf("%6s", "INF");

else

printf("%6d", path[i][j]);

}

printf("\n");

}

// Query shortest path between two vertices

printf("\nEnter start vertex: ");

scanf("%d", &start);

printf("Enter end vertex: ");

scanf("%d", &end);

if (path[start][end] == INFINITY)

printf("No path exists between %d and %d\n", start, end);

else

printf("The minimum cost between %d and %d is %d\n", start, end, path[start][end]);

return 0;

}

**Output:**

Enter number of vertices: 4

Enter weighted matrix (0 for no path):

0 3 0 7

3 0 1 5

0 1 0 2

7 5 2 0

Shortest path matrix:

0 3 4 6

3 0 1 3

4 1 0 2

6 3 2 0

Enter start vertex: 0

Enter end vertex: 3

The minimum cost between 0 and 3 is 6