**Bansilal Ramnath Agarwal Charitable Trust’s**

Vishwakarma Institute of Technology, Pune-37

*(Anautonomous Institute of Savitribai Phule Pune University)*

**Department of Computer Engineering**

|  |  |
| --- | --- |
| **Division** | **A** |
| **Batch** | **B2** |
| **GR-no** | **12110353** |
| **Roll no** | **10** |
| **Name** | **Aneesh Hemant Dighe** |

***CODE:***

#include <stdio.h>

#include <limits.h>

#define MAX\_VERTICES 100

struct OSPF {

    int vertices;

    int graph[MAX\_VERTICES][MAX\_VERTICES];

    int parent[MAX\_VERTICES];

};

void init(struct OSPF\* ospf, int vertices) {

    ospf->vertices = vertices;

    for (int i = 0; i < vertices; i++) {

        for (int j = 0; j < vertices; j++) {

            ospf->graph[i][j] = 0;

        }

        ospf->parent[i] = -1;

    }

}

void addEdge(struct OSPF\* ospf, int source, int destination, int weight) {

    ospf->graph[source][destination] = weight;

    ospf->graph[destination][source] = weight;

}

int minDistance(const struct OSPF\* ospf, const int\* dist, const int\* sptSet) {

    int min = INT\_MAX;

    int minIndex = -1;

    for (int v = 0; v < ospf->vertices; v++) {

        if (!sptSet[v] && dist[v] <= min) {

            min = dist[v];

            minIndex = v;

        }

    }

    return minIndex;

}

void printPath(const struct OSPF\* ospf, int destination) {

    if (destination == -1)

        return;

    printPath(ospf, ospf->parent[destination]);

    printf("%d ", destination);

}

void printSolution(const struct OSPF\* ospf, const int\* dist, int source) {

    printf("Vertex\tDistance from Source\tPath\n");

    for (int i = 0; i < ospf->vertices; i++) {

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

        printPath(ospf, i);

        printf("\n");

    }

}

void dijkstra(struct OSPF\* ospf, int source) {

    int dist[MAX\_VERTICES];

    int sptSet[MAX\_VERTICES];

    for (int i = 0; i < ospf->vertices; i++) {

        dist[i] = (i == source) ? 0 : INT\_MAX;

        ospf->parent[i] = -1;

        sptSet[i] = 0;

    }

    for (int count = 0; count < ospf->vertices - 1; count++) {

        int u = minDistance(ospf, dist, sptSet);

        sptSet[u] = 1;

        for (int v = 0; v < ospf->vertices; v++) {

            if (!sptSet[v] && ospf->graph[u][v] != 0 && dist[u] != INT\_MAX

                && dist[u] + ospf->graph[u][v] < dist[v]) {

                dist[v] = dist[u] + ospf->graph[u][v];

                ospf->parent[v] = u;

            }

        }

    }

    printSolution(ospf, dist, source);

}

int getShortestDistance(const struct OSPF\* ospf, int destination) {

    return (ospf->parent[destination] == -1) ? INT\_MAX : ospf->parent[destination];

}

int main() {

    int vertices, edges, sourceVertex, destinationVertex;

    printf("Enter the number of routers: ");

    scanf("%d", &vertices);

    struct OSPF graph;

    init(&graph, vertices);

    printf("Enter the number of edges: ");

    scanf("%d", &edges);

    for (int i = 0; i < edges; i++) {

        int source, destination, weight;

        printf("Enter source, destination, and weight for edge %d: ", i + 1);

        scanf("%d %d %d", &source, &destination, &weight);

        addEdge(&graph, source, destination, weight);

    }

    printf("Enter the source router: ");

    scanf("%d", &sourceVertex);

    printf("Enter the destination router: ");

    scanf("%d", &destinationVertex);

    dijkstra(&graph, sourceVertex);

    int shortestDistance = getShortestDistance(&graph, destinationVertex);

    if (shortestDistance == INT\_MAX) {

        printf("There is no path from source to destination.\n");

    } else {

        printf("Shortest path: ");

        printPath(&graph, destinationVertex);

        printf("\n");

    }

    return 0;

}

***OUTPUT:***

PS D:\5th Sem\CN(Computer Networks)\Assignments> ./Djisktraospf.exe

Enter the number of routers: 6

Enter the number of edges: 5

Enter source, destination, and weight for edge 1: 0 1 4

Enter source, destination, and weight for edge 2: 1 2 5

Enter source, destination, and weight for edge 3: 2 3 4

Enter source, destination, and weight for edge 4: 4 5 6

Enter source, destination, and weight for edge 5: 3 4 3

Enter the source router: 0

Enter the destination router: 4

Vertex Distance from Source Path

0 0 0

1 4 0 1

2 9 0 1 2

3 13 0 1 2 3

4 16 0 1 2 3 4

5 22 0 1 2 3 4 5

Shortest path: 0 1 2 3 4