**Assignment 5**

## Name : Shrinivas Hatyalikar

## Div: TY-B (B2)

## Roll No: 26

Q) Write a program to find the shortest path using Dijkstra Equation for Link State Routing Protocol which is used by Open Shortest Path First Protocol (OSPF) in the Internet for the network flow provided by instructor.

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <stdbool.h>

#include <limits.h>

#define MAX\_NODES 100

// Structure to represent a node in the network graph

struct Node {

    int id;

    int cost[MAX\_NODES];  // Array to store link costs to other nodes

};

// Function to find the node with the minimum distance

int minDistance(int dist[], bool sptSet[], int numNodes) {

    int min = INT\_MAX;

    int minIndex;

    for (int v = 0; v < numNodes; v++) {

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

            min = dist[v];

            minIndex = v;

        }

    }

    return minIndex;

}

// Function to perform Dijkstra's algorithm

void dijkstra(struct Node graph[MAX\_NODES], int numNodes, int src) {

    int dist[MAX\_NODES];

    bool sptSet[MAX\_NODES];

    // Initialize distance and sptSet

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

        dist[i] = INT\_MAX;

        sptSet[i] = false;

    }

    // Distance of source node from itself is always 0

    dist[src] = 0;

    // Find shortest path for all nodes

    for (int count = 0; count < numNodes - 1; count++) {

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

        sptSet[u] = true;

        // Update dist[] value of adjacent nodes

        for (int v = 0; v < numNodes; v++) {

            if (!sptSet[v] && graph[u].cost[v] && dist[u] != INT\_MAX

                && dist[u] + graph[u].cost[v] < dist[v]) {

                dist[v] = dist[u] + graph[u].cost[v];

            }

        }

    }

    // Print the shortest path distances

    printf("Shortest Path Distances from Node %d:\n", src);

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

        printf("Node %d: %d\n", i, dist[i]);

    }

}

int main() {

    int numNodes;

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

    scanf("%d", &numNodes);

    struct Node graph[MAX\_NODES];

    // Input link costs between nodes

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

        graph[i].id = i;

        printf("Enter link costs for Node %d to other nodes (0 for no connection):\n", i);

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

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

        }

    }

    int sourceNode;

    printf("Enter the source node: ");

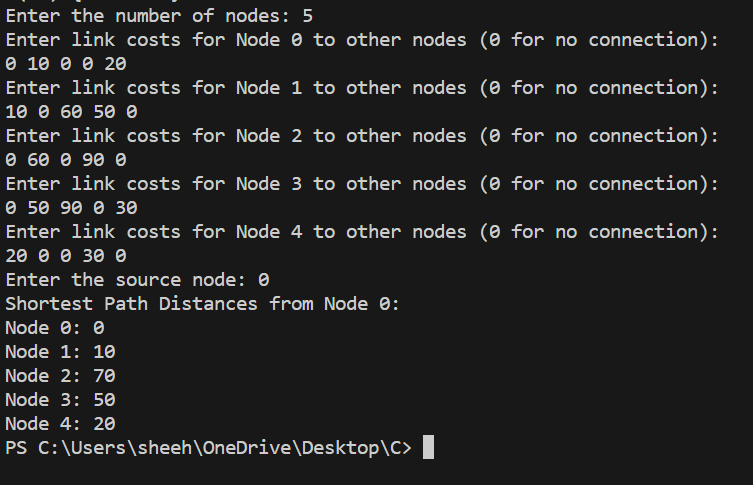
    scanf("%d", &sourceNode);

    dijkstra(graph, numNodes, sourceNode);

    return 0;

}

**Output:**

****