DIJIKSTRA’S ALGORITHM

AIM:

The code aims to create a graph and to find the shortest path between two vertices using Dijikstra’s Algorithm.

ALGORITHM:

1. Start.
2. The main function calls the create graph function.
3. Input the number of vertices and the adjacency matrix representing the graph.
4. Find the vertex with the minimum distance from the source vertex among the vertices.
5. Then, it implements Dijkstra's algorithm, initializes distance values and sptset for all vertices, and iteratively updates the distance values until all vertices are included in the shortest path tree.
6. Display the Output
7. End.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#define MAX 100
#define INF INT_MAX

int minDistance(int dist[], int sptSet[], int V) {
    int min = INF, min_index;
    for (int v = 0; v < V; v++) {
        if (sptSet[v] == 0 && dist[v] <= min) {
            min = dist[v];
            min_index = v;
        }
    }
    return min_index;
}

void printSolution(int dist[], int V) {
    printf("Vertex \t Distance from Source\n");
    for (int i = 1; i <=V; i++) {
        printf("%d \t\t %d\n", i, dist[i]);
    }
}


void dijkstra(int graph[MAX][MAX], int V, int src) {
    int dist[MAX];
    int sptSet[MAX];
    for (int i = 0; i <=V; i++) {
        dist[i] = INF;
        sptSet[i] = 0;
    }
    dist[src] = 0;
    for (int count = 0; count <V - 1; count++) {
        int u = minDistance(dist, sptSet, V);
        sptSet[u] = 1;
        for (int v = 0; v <=V; v++) {
            if (!sptSet[v] && graph[u][v] && dist[u] != INF && dist[u] + graph[u][v] < dist[v]) {
                dist[v] = dist[u] + graph[u][v];
            }
        }
    }
    printSolution(dist, V);
}

int main() {
    int V, E;
    printf("Enter the number of vertices: ");
    scanf("%d", &V);
    int graph[MAX][MAX] = {0};
    printf("Enter the number of edges: ");
    scanf("%d", &E);
    printf("Enter the edges (source destination weight):\n");
    for (int i = 0; i < E; i++) {
        int src, dest, weight;
        scanf("%d %d %d", &src, &dest, &weight);
        graph[src][dest] = weight;
        graph[dest][src] = weight;
    }
    int srcVertex;
    printf("Enter the source vertex: ");
    scanf("%d", &srcVertex);
    printf("Adjacency Matrix of the graph:\n");
    for (int i = 0; i < V; i++) {
        for (int j = 0; j < V; j++) {
            printf("%d ", graph[i][j]);
        }
        printf("\n");
    }
    printf("\nShortest paths from vertex %d using Dijkstra's algorithm:\n", srcVertex);
    dijkstra(graph, V, srcVertex);
    return 0;
}
```

OUTPUT:

```
Enter the number of vertices: 4
Enter the number of edges: 5
Enter the edges (source destination weight):
1 2 5
1 3 3
2 3 8
2 4 2
3 4 6
Enter the source vertex: 1
Adjacency Matrix of the graph:
0 0 0 0
0 0 5 3
0 5 0 8
0 3 8 0
Shortest paths from vertex 1 using Dijkstra's algorithm:

Vertex          Distance from source
1                0
2                5
3                3
4                7
```

RESULT:

Thus the program has been successfully executed and verified.