PROGRAM 8

From a given vertex in a weighted connected graph, find shortest paths to other vertices using Dijkstra's algorithm.

ALGORITHM

```
- ALGORITHM:
 Dijkolea (Graph, Lousu):
1. Greate a set minted to store wissted needices Crostalge todales for
 a Goate a distance away 'did 17' and entialize:
     - dict [Rourie] = 0
     - dil [v] = to for all other nectices v fromse
3. Repeat (V-1) dimer:
   a. u - nealen en dell's with the minimum det fut much that
     writed [u] is false
   b. Mark wifed [u] = level
   c. For each neighbour v of u:
       ?. If v & not worked soo there is an edge form u
      to v.
           - tf dist [a] + weight (u,v) < dx[v]:
               -> det[v] = diffu] + weight(u,v)
4. End loop
5. Output the dist [] array (Thortest defance from all source)
```

CODE

```
#include <stdio.h>
#include <limits.h>
#define V 100 // max number of vertices

// Find vertex with minimum distance not yet processed
int minDistance(int dist[], int visited[], int n) {
  int min = INT_MAX, min_index = -1;
  for (int v = 0; v < n; v++) {
    if (!visited[v] && dist[v] <= min) {</pre>
```

```
min = dist[v], min\_index = v;
  return min index;
}
void dijkstra(int graph[V][V], int src, int n) {
  int dist[V];
                // Shortest distances
  int visited[V]; // True if vertex is included in shortest path tree
  // Initialize distances and visited
  for (int i = 0; i < n; i++) {
     dist[i] = INT\_MAX;
     visited[i] = 0;
  }
  dist[src] = 0; // Distance to itself is 0
  // Find shortest path for all vertices
  for (int count = 0; count \leq n - 1; count++) {
     int u = minDistance(dist, visited, n);
     visited[u] = 1;
     // Update distances of adjacent vertices
     for (int v = 0; v < n; v++) {
       if (!visited[v] && graph[u][v] && dist[u] != INT MAX &&
          dist[u] + graph[u][v] < dist[v]) {
          dist[v] = dist[u] + graph[u][v];
```

```
}
  // Print results
  printf("Vertex \t Distance from Source\n");
  for (int i = 0; i < n; i++)
     printf("\%d \t\ \%d\n", i, dist[i]);
}
// Driver code
int main() {
  int n;
  printf("Enter number of vertices: ");
  scanf("%d", &n);
  int graph[V][V];
  printf("Enter the adjacency matrix (use 0 if no edge):\n");
  for (int i = 0; i < n; i++)
     for (int j = 0; j < n; j++)
       scanf("%d", &graph[i][j]);
  int src;
  printf("Enter source vertex: ");
  scanf("%d", &src);
  dijkstra(graph, src, n);
  return 0;
```

OUTPUT

```
Enter number of vertices: 5
Enter the adjacency matrix (use 0 if no edge
   ):
0 10 0 30 100
10 0 50 0 0
0 50 0 20 10
30 0 20 0 60
100 0 10 60 0
Enter source vertex: 0
Vertex
        Distance from Source
         O
1
         10
2
         50
         30
4
         60
```

TRACING

Starting rules from Lower rules 0:

dist[u]= min (dist (u) + wit (u,v), dist [v])

du (un=min (du (un + w(u,v))			
Step	Current nectex	Distances	vended
	0	0 10 INt 30 100	no 3000 6
2	1	0 10 60 30 100	011
3	3	0 10 50 30 90	0,1,3
4	2	0 10 50 30 60	0,1,3,2
5	send of 4th and	0 10 50 30 60	all
			3 - 3 - 3