Experiment 23: Dijkstra's Algorithm

Aim:
To write a C program to find the shortest path using Dijkstra's algorithm.

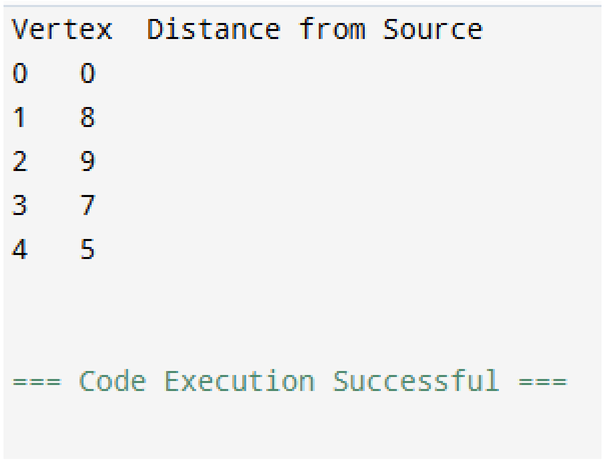
Algorithm:

1. Start the program.
2. Initialize distance array with infinity and visited array as false.
3. Set distance of source to 0.
4. Repeat for all vertices:
    - Pick unvisited vertex with minimum distance.
    - Mark it visited.
    - Update distances of its adjacent vertices.
5. Stop when all vertices are visited.
6. Print distances.

Code:

```
#include <stdio.h>
#define INF 9999
#define V 5

int minDistance(int dist[], int visited[]) {
    int min = INF, min_index = -1;
    for (int v = 0; v < V; v++) {
        if (!visited[v] && dist[v] <= min) {
            min = dist[v]; min_index = v;
        }
    }
    return min_index;
}

void dijkstra(int graph[V][V], int src) {
    int dist[V], visited[V] = {0};
    for (int i = 0; i < V; i++) dist[i] = INF;
    dist[src] = 0;

    for (int count = 0; count < V-1; count++) {
        int u = minDistance(dist, visited);
        visited[u] = 1;
        for (int v = 0; v < V; v++) {
            if (!visited[v] && graph[u][v] && dist[u] + graph[u][v] < dist[v])
                dist[v] = dist[u] + graph[u][v];
        }
    }

    printf("Vertex\tDistance from Source\n");
    for (int i = 0; i < V; i++)
        printf("%d\t%d\n", i, dist[i]);
}

int main() {
    int graph[V][V] = {
        {0,10,0,0,5},
        {0,0,1,0,2},
        {0,0,0,4,0},
        {7,0,6,0,0},
        {0,3,9,2,0}
    };
    dijkstra(graph, 0);
    return 0;
}
```

Sample Output:



```
Vertex   Distance from Source
0    0
1    8
2    9
3    7
4    5


=== Code Execution Successful ===
```

Result:
The program successfully finds the shortest paths using Dijkstra’s algorithm.