Name: Ankush Singh
Group: 5C13
Roll No.: 040

Date: 10-10-2023

EXPERIMENT- 5

Aim:

To implement Dijkstra‘s algorithm and analyse its time complexity.

Code:

```
#include <bits/stdc++.h>
using namespace std;

#define INF INT_MAX

class Graph {
public:
   int vertices;
   vector<vector<int>> adjMatrix;

   Graph(int V) {
      vertices = V;
      adjMatrix.resize(V, vector<int>(V, INF));
   }

   void addEdge(int src, int dest, int weight) {
      adjMatrix[src][dest] = weight;
      adjMatrix[dest][src] = weight; // For undirected graph
   }

   int minDistance(vector<int>& dist, vector<bool>& sptSet) {
      int minDist = INF, minIndex;

      for (int v = 0; v < vertices; ++v) {
         if (!sptSet[v] && dist[v] <= minDist) {
            minDist = dist[v];
            minIndex = v;
         }
      }
      return minIndex;
   }

   void dijkstra(int src) {
      vector<int> dist(vertices, INF);
      vector<bool> sptSet(vertices, false);

      dist[src] = 0;

      for (int count = 0; count < vertices - 1; ++count) {
         int u = minDistance(dist, sptSet);
         sptSet[u] = true;
```

```
            for (int v = 0; v < vertices; ++v) {
               if (!sptSet[v] && adjMatrix[u][v] != INF &&
                  dist[u] != INF && dist[u] + adjMatrix[u][v] < dist[v]) {
                  dist[v] = dist[u] + adjMatrix[u][v];
               }
            }
         }
         cout << "Vertex   Distance from Source\n";
         for (int i = 0; i < vertices; ++i) {
            cout << i << "\t\t\t\t" << dist[i] << endl;
         }
      }
};
int main() {
   int vertices = 6;

   Graph graph(vertices);

   graph.addEdge(0, 1, 5);
   graph.addEdge(0, 2, 2);
   graph.addEdge(1, 3, 4);
   graph.addEdge(2, 4, 7);
   graph.addEdge(3, 5, 3);
   graph.addEdge(4, 5, 1);

   int source = 0;
   cout<<"Source: "<<source<<endl;
   clock_t start_time = clock();

   graph.dijkstra(source);

   clock_t end_time = clock();
   double execution_time = static_cast<double>(end_time - start_time)*1000.0 / CLOCKS_PER_SEC;

   cout << "Execution Time: " << execution_time << " milliseconds" << endl;

   return 0;
}
```

Output:

```
Source: 0
Vertex   Distance from Source
0                0
1                5
2                2
3                9
4                9
5                10
Execution Time: 0.046 milliseconds
```