Experiment: 5

Write a program to implement Dijkstra's algorithm for a graph.

Student Name: Nitish Rai

UID: 23MCA20326

Branch: Computers Application

Section/Group: 4(A)

Semester: 2nd

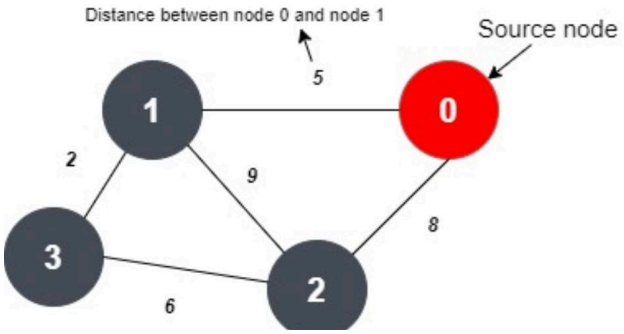
Date of Performance: 28/02/2024

Subject Name: Design and Analysis of Algorithms

Subject Code: 23CAH-511

A. The task is to:

Implement Dijkstra's algorithm to calculate the single-source shortest paths from a source vertex to all other vertices in the graph.



B. Steps of Experiment:

- Understand and divide the problem into parts.
- Install JDK: If Java isn't installed on your system, download and install it.
- Open IDE(Integrated development environment) like VS code.
- Start creating a new Java file with a .java extension (e.g., program.java).
- Write your code in a structural manner taking care of indentation to maintain readability.
- Execute your code

C. Algorithm:

1. Mark the source node with a current distance of 0 and the rest with infinity.

2. Set the non-visited node with the smallest current distance as the current node, let's say C.

3. For each neighbour N of the current node C: add the current distance of C with the weight of the edge connecting C-N. If it is smaller than the current distance of N, set it as the new current distance of N.

4. Mark the current node C as visited.

5. Go to step 2 if there are any nodes are unvisited.

D. Pseudocode:

```
function Dijkstra(Graph, source):
  for each vertex v in Graph:
    distance[v] = infinity

 distance[source] = 0
 G = the set of all nodes of the Graph

 while G is non-empty:
     Q = node in G with the least dist[ ]
     mark Q visited
     for each neighbour N of Q:
         alt_dist = distance[Q] + dist_between(Q, N)
         if alt-dist < distance[N]
             distance[N]:= alt_dist

 return distance[ ]
```

E. Code:

```
  // Applying Dijkstra's Algorithm

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

vector<int> shortestPath(vector< vector<int> > &edges, int n, int m, int src) {
    // Creating adj list
    unordered_map< int, list<pair<int,int>> > adjList;
    for(int i=0; i<m; i++) {
        int u = edges[i][0];
        int v = edges[i][1];
        int w = edges[i][2];

        adjList[u].push_back({v,w});
        adjList[v].push_back({u,w});
    }

    vector<int> distance(n,INT_MAX);
    set<pair<int,int>> st;

    distance[src] = 0;
    st.insert({0,src});

    while(!st.empty()) {
        // Fetching top pair
        pair<int,int> curr = *(st.begin());

        int nodeDistance = curr.first;
        int topNode = curr.second;

        // Removing top pair
        st.erase(st.begin());

        for(auto neigh : adjList[topNode]) {
            if(nodeDistance + neigh.second < distance[neigh.first]) {
                auto record = st.find({distance[neigh.first], neigh.first});

                // Erasing old record
                if(record != st.end()) {
                    st.erase(record);
                }

                // Upadting distance
                distance[neigh.first] = nodeDistance + neigh.second;

                // Inserting new pair in set
                st.insert({distance[neigh.first], neigh.first});
            }
        }
    }

    return distance;
}
```

```
int main() {
    vector<vector<int>> edges;
    int n, m, src;

    cout << "Enter the number of nodes : ";
    cin >> n;

    cout << "Enter the number of edges : ";
    cin >> m;

    cout << "Enter the edges : " << endl;
    for(int i=0; i<m; i++) {
        int u, v, w;
        cin >> u >> v >> w;
        edges.push_back({u,v,w});
    }

    cout << "Enter the source : ";
    cin >> src;

    vector<int> answer = shortestPath(edges, n, m, src);
    cout << "The shortest path : ";
    for(int x : answer) {
        cout << x << " ";
    }

    return 0;
```

F. Output:

```
PS C:\Users\Nitish\Desktop\JAVA\javaComeBack> cd "c:\Users\Nitish\Desktop\JAVA\javaComeBack\" ; if ($?) { javac
DijkstrasAlgo.java } ; if ($?) { java DijkstrasAlgo }
Enter the number of nodes : 4
Enter the number of edges : 5
Enter the edges :
0 1 5
0 2 8
1 2 9
1 3 2
2 3 6
Enter the source : 0
The shortest path : 0 5 8 7
PS C:\Users\Nitish\Desktop\JAVA\javaComeBack>
```

G. Time Complexity:

O(E log V)

Learning outcomes:

- Understand how to use sets.
- Understand the concept of the creation of a graph.
- Understanding how to deal with errors..
- Understand what Dijkstra's algorithm is and how to implement it.
- Understand the adjacency matrix.