Name of Experiment :

Introduction to Dijkstra Algorithm

Objective:

1. To understand the working principle of Dijkstra Algorithm
2. To implement Dijkstra algorithm using adjacency list
3. To analyze the time complexity and space complexity of Dijkstra Algorithm

Theory:

Dijkstra algorithm is one of the most famous algorithm to find shortest path from a a source to all other vertex for weighted undirected graph . It maintains a set of vertex whose distance from the source is already known .

Algorithm:

Step 01: Initialize distance to all vertex as infinity and distance from source to source is 0
Step 02: Insert the source node into a priority queue.
Step 03: While the queue is not empty
    Extract the vertex u with the smallest distance
    For each neighbor v of u , if the path through u gives smaller distance to v , update it.
    Continue till all vertex are processed
Step 04: output the shortest distance

Code :

```
#include <bits/stdc++.h>
using namespace std ;
#define ll long long
const ll INF = 1e18;

int main(){
        ll n, m;
        cin>>n>>m;
        vector<pair<ll,ll>> g[n+1];

  for(ll i=0; i<m; i++)
  {
```

```
    ll x, y, z;
    cin>>x>>y>>z;
    g[x].push_back({y, z});
  }

  vector<ll> d(n+1, INF);
  priority_queue<pair<ll,ll> , vector<pair<ll,ll>> , greater<pair <ll,ll>>> pq;

  d[1]=0;
  pq.push(make_pair (0LL,1LL));

  while(!pq.empty()){
    pair<ll ,ll> top= pq.top();
    pq.pop();
    ll curr_node= top.second;
    ll curr_distance=top.first;
    if (curr_distance > d[curr_node]) continue;

    for ( size_t i=0; i< g[curr_node].size();i++){
        ll child_node = g[curr_node][i].first;
        ll child_distance = g[curr_node][i].second;
        if(curr_distance + child_distance < d[child_node]){
                        d[child_node]=curr_distance + child_distance;
                        pq.push(make_pair (d[child_node], child_node));
        }
    }
  }
  for (ll i=1; i<=n;i++){
       cout<<d[i]<<" ";
  }
  return 0;
}
```

Time Complexity:

O((V+E) log V) using a priority queue (binary heap)

Space Complexity:

O(V+E)

Conclusion:

In the experiment , we have implemented the Dijkstra algorithm using c++ . it works efficiently when we need to find the single source shortest path for an unweighted undirected graph with non- negative edge weights.
Understanding Dijkstra algorithm helps in building strong foundation in all other graph algorithms like Bellman- Ford Algorithm and Floyd Warshel Algorithm .

Reference:

1. Cormen, Leiserson, Rivest, and Stein, *Introduction to Algorithms*, MIT Press.
2. GeeksforGeeks: https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm/
3. Programiz: https://www.programiz.com/dsa/dijkstra-algorithm
4. Class Lecture