CHITTAGONG UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Department of Computer Science & Engineering

Course No: CSE- 244

Course Title:
Algorithm Design & Analysis (Sessional)

Experiment No: 07

Name Of the Experiment: Shortest Path, MST.

Identity Details

Name of the Student: Md. Safayat Bin Nasir

Student Id Number: 2004099

Level: 2 Term: 2 Section: B

Group: B1

Date of performance: 12-10-2023

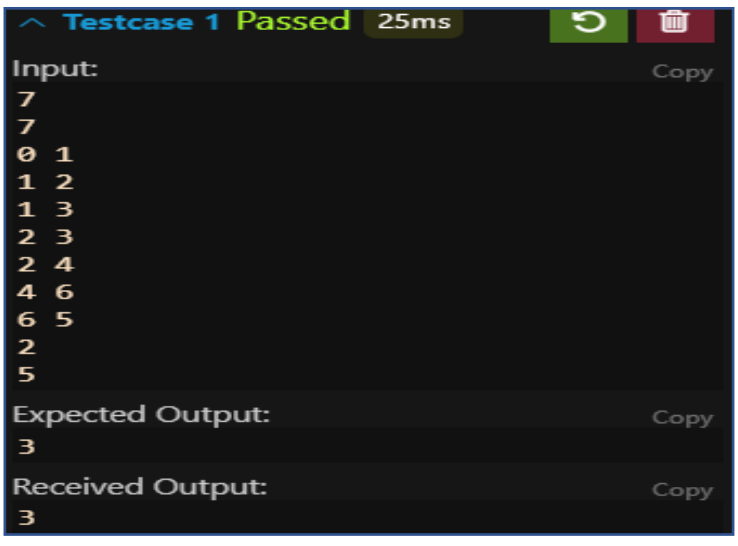
Date of Submission: 19-11-2023

Remarks

Program 1: Given an undirected unweighted graph. Find the minimum distance between two nodes using BFS algorithm.

```cpp
#include <bits/stdc++.h>
using namespace std;
bool visited[10000000];
long long dist[10000000];
vector<long long> graph[10000];
void bfs(long long source)
{
    queue<long long> q;
    visited[source] = 1;
    dist[source] = 0;
    q.push(source);
    while (!q.empty())
    {
        int node = q.front();
        q.pop();
        for (long long i = 0; i < graph[node].size(); i++)
        {
            long long next = graph[node][i];
            if (visited[next] == 0)
            {
                visited[next] = 1;
                dist[next] = dist[node] + 1;
                q.push(next);
            }
        }
    }
}
int main()
{
    long long node, edge;
    cin >> node >> edge;
    for (long long i = 1; i <= edge; i++)
    {
        long long u, v;
        cin >> u >> v;
        graph[u].push_back(v) , graph[v].push_back(u);
    }
    int source , dest;
    cin >> source >> dest;
    bfs(source);
    cout << dist[dest] << endl;
}
```

Output:



Program 2: Given an undirected weighted graph. Find the minimum distance between two nodes using Dijkstra algorithm.

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5+10;
const int INF = 1e9+10;
vector<pair<int , int>> g[N];
vector<int>dist(N, INF);
void dijkstra(int source , int n){
    vector<int>vis(N,0);
    set<pair<int,int>>s;
    s.insert({0 , source});
    dist[source] = 0;
    while(s.size() > 0){
        auto node = *s.begin();
        int v = node.second;
        s.erase(s.begin());
        if(vis[v] ) continue;
        vis[v] = 1;
        int dis = node.first;
        for(auto child:g[v]){
            int childv = child.first;
            int wt = child.second;
            if(dist[v] + wt < dist[childv]){
                dist[childv] = dist[v] + wt;
                s.insert({dist[childv], childv});
            }
        }
    }
}
int main()
{
    int n   , m;
    cin >> n >> m;
    for(int i = 0 ; i < m; i++){
        int u , v , w;
        cin >> u >> v >> w;
        g[u].push_back({v, w});
        g[v].push_back({u, w});
    }
    int source , dest;
    cin >> source >> dest;
    dijkstra(source , dest);
    cout << dist[dest] << endl;
}
```

Output:

Testcase 1 Passed 26ms

Input: Copy

```
5
7

0 1 3
0 3 7
0 4 8
1 2 1
1 3 4
2 3 2
3 4 3
2
0
```

Expected Output: Copy

```
4
```

Received Output: Copy

```
4
```