Experiment : 2.1

Student Name: SANJIV GUPTA **UID:** 21BCS-*3478*
Branch: CSE **Section/Group:** 21BCS-IOT-602B
Semester: 5th **Date:** 22/09/23
Subject Name: Advanced Programming LAB **Subject Code:** 21CSP-314

AIM:

To implement the concept of Graphs.

OBJECTIVE:

1).Consider an undirected graph where each edge weighs 6 units. Each of the nodes is labeled consecutively from 1 to n.

You will be given a number of queries. For each query, you will be given a list of edges describing an undirected graph. After you create a representation of the graph, you must determine and report the shortest distance to each of the other nodes from a given starting position using the breadth-first search algorithm (BFS). Return an array of distances from the start node in node number order. If a node is unreachable, return for that node.

2.) Markov takes out his Snakes and Ladders game, stares at the board and wonders: "If I can always roll the die to whatever number I want, what would be the least number of rolls to reach the destination?"

***Rules** The game is played with a cubic die of 6 faces numbered 1 to 6.*

1. *Starting from square 1 , land on square 100 with the exact roll of the die. If moving the number rolled would place the player beyond square 100 , no move is made.*
2. *If a player lands at the base of a ladder, the player must climb the ladder. Ladders go up only.*
3. *If a player lands at the mouth of a snake, the player must go down the snake and come out through the tail. Snakes go down only.*

CODE:

Code 1:

```
vector<int> bfs(int n, int m, vector<vector<int>> edges, int s) {
   int dist[n];
   map<int,vector<int>>mp;
   for(auto x:edges)
```

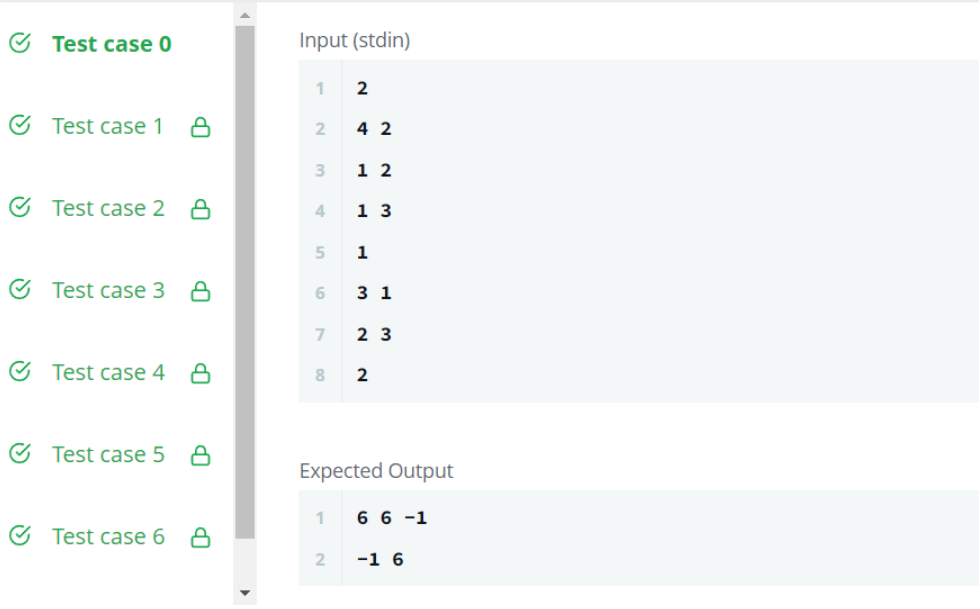
```
    {
        mp[x[0]].push_back(x[1]);
        mp[x[1]].push_back(x[0]);
    }
    for (int i=1;i<=n;i++){
        dist[i] = INT_MAX;
    }
    priority_queue<pair<int,int>,vector<pair<int,int>>,greater<pair<int,int>>>pq;
    pq.push({0,s});
    dist[s] = 0;

    while(!pq.empty())
    {
        int cost = pq.top().first;
        int src = pq.top().second;
        pq.pop();
        for(auto x: mp[src])
        {
            if(dist[x] > cost +6)
            {
                dist[x] = cost+6;
                pq.push({dist[x],x});
            }
        }
    }

    vector<int>ans;
    for(int i=1;i<=n;i++)
    {
        if(i==s)
        {
            continue;
        }
        else if(dist[i] == INT_MAX)
        {
            ans.push_back(-1);
        }
        else {
            ans.push_back(dist[i]);
        }
    }
    return ans;
}
```

Code 2

```
int quickestWayUp(vector<vector<int>> ladders, vector<vector<int>> snakes) {
  map<int,int> ladd, snak;
    for(auto &it: ladders) ladd[it[0]] = it[1];
    for(auto &it: snakes) snak[it[0]] = it[1];
    queue<pair<int,int>> q;
    vector<int> vis(101,0);
    q.push({1,0});
    vis[1] = 1;
    vector<int> dist(101,INT_MAX);

    while(!q.empty()){
        int nd = q.front().first;
        int tym = q.front().second;
        q.pop();

        for(int i=1; i<=6;i++){
            int nxt = nd + i;
            if(ladd[nxt]) nxt = ladd[nxt];
            if(snak[nxt]) nxt = snak[nxt];
            if(vis[nxt]) continue;
            if(nxt == 100) return tym+1;
            if(!vis[nxt]) vis[nxt] = 1;
            q.push({nxt,tym+1});
        }
    }

    return -1;
}
```

OUTPUT:

OUTPUT 1



OUTPUT 2

LEARNING OUTCOMES:

1. *Understood the concept of Graph.*
2. *Understood the concept how to search in graph and perform different operations.*
3. *Learn about algorithm thinking*
4. *Learn about mathematical logic*