Crack – a – Hack

Afraid of the Dark

Course: ALGORITHMIC PROBLEM SOLVING

Course Code: 17ECSE309

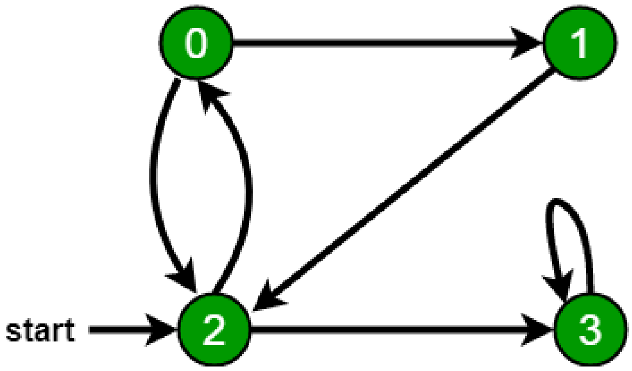
By: Prajakta Desai
01FE15BCS133

Introduction:

Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. One starts at the root (selecting some arbitrary node as the root in the case of a graph) and explores as far as possible along each branch before back-tracking. Time Complexity can be expressed as O(V+E) where V is number of vertices in the graph and E is number of edges in the graph.

Example:

For example, in the following graph, we start traversal from vertex 2. When we come to vertex 0, we look for all adjacent vertices of it. 2 is also an adjacent vertex of 0. If we don't mark visited vertices, then 2 will be processed again and it will become a non-terminating process. A Depth First Traversal of the following graph is 2, 0, 1, 3.



Hackerrank Question Link:

[Women CodeSprint 4 : Afraid of the Dark]()

Problem Description:

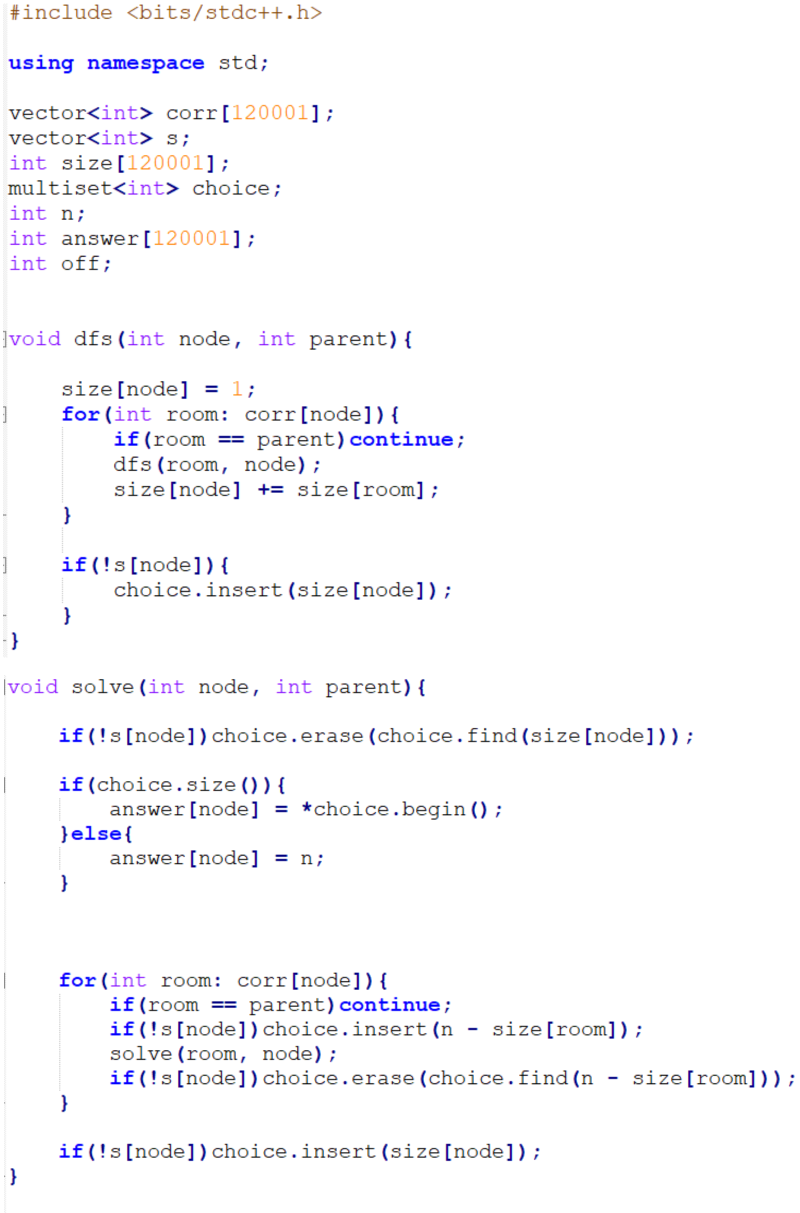
There *n* room labelled from 1 to *n*. There are corridors connecting some pairs of rooms. The corridors and rooms form a tree with nodes representing the rooms and *n-1* edges representing the corridors.

Each room contains a light bulb which is either *on* or *off*. Starting at a room, visit as many possible rooms as possible but to visit a room and pass through its incident corridors, the light bulb in that room should be on.

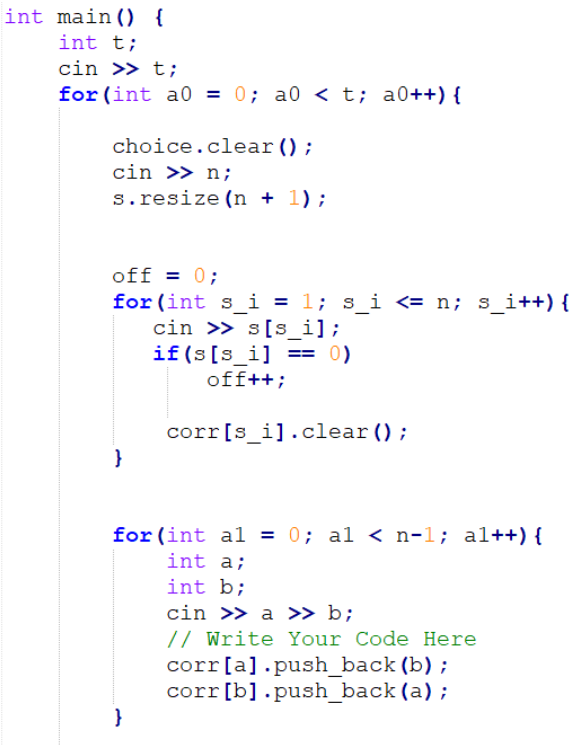
Constraints: When a room with off light bulb is visited, the bulb can be turned on. Only even number of times a bulb can be turned on.

Solution:

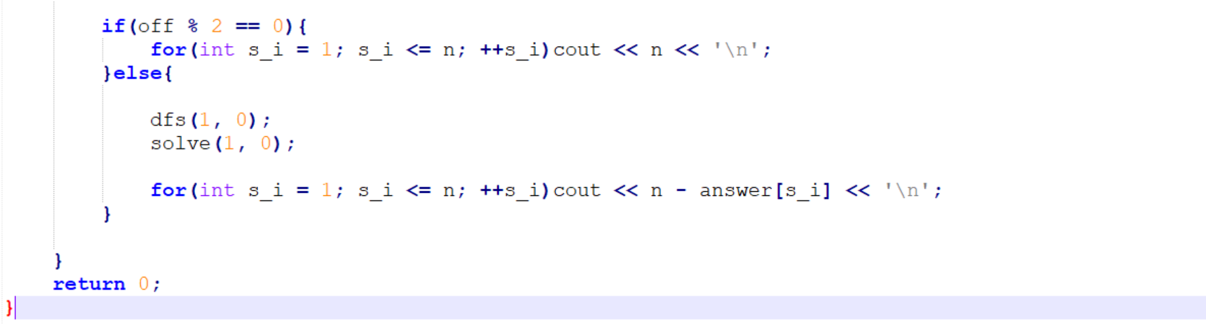
The solution is written in c++:



```
#include <bits/stdc++.h>

using namespace std;

vector<int> corr[120001];
vector<int> s;
int size[120001];
multiset<int> choice;
int n;
int answer[120001];
int off;


void dfs(int node, int parent){

    size[node] = 1;
    for(int room: corr[node]){
        if(room == parent)continue;
        dfs(room, node);
        size[node] += size[room];
    }

    if(!s[node]){
        choice.insert(size[node]);
    }
}

void solve(int node, int parent){

    if(!s[node])choice.erase(choice.find(size[node]));

    if(choice.size()){
        answer[node] = *choice.begin();
    }else{
        answer[node] = n;
    }



    for(int room: corr[node]){
        if(room == parent)continue;
        if(!s[node])choice.insert(n - size[room]);
        solve(room, node);
        if(!s[node])choice.erase(choice.find(n - size[room]));
    }

    if(!s[node])choice.insert(size[node]);
}
```



```
int main() {
    int t;
    cin >> t;
    for(int a0 = 0; a0 < t; a0++){

        choice.clear();
        cin >> n;
        s.resize(n + 1);


        off = 0;
        for(int s_i = 1; s_i <= n; s_i++){
           cin >> s[s_i];
           if(s[s_i] == 0)
               off++;

            corr[s_i].clear();
        }


        for(int a1 = 0; a1 < n-1; a1++){
            int a;
            int b;
            cin >> a >> b;
            // Write Your Code Here
            corr[a].push_back(b);
            corr[b].push_back(a);
        }
```



```
        if(off % 2 == 0){
            for(int s_i = 1; s_i <= n; ++s_i)cout << n << '\n';
        }else{

            dfs(1, 0);
            solve(1, 0);

            for(int s_i = 1; s_i <= n; ++s_i)cout << n - answer[s_i] << '\n';
        }

    }
    return 0;
}
```

➢ Time Complexity:

It takes a time complexity of O(n).

➢ References:

1. https://en.wikipedia.org/wiki/Depth-first_search
2. https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/