Description | Hints | Submissions | Discussions | Notes

Centroid

Difficulty	Time Limit	Memory	Score
[illegible]	1 sec	256000KB	100

80/80 XP | 30/30

Description

Given a tree of n nodes, your task is to find a *centroid*, i.e., a node such that when it is appointed the root of the tree, each subtree has at most $\lfloor n/2 \rfloor$ nodes.

Input Format

The first input line contains an integer n: the number of nodes. The nodes are numbered 1,2,...,n.

Then there are $n-1$ lines describing the edges. Each line contains two integers a and b: there is an edge between nodes a and b.

Output Format

Print one integer: a centroid node. If there are several possibilities, print the smallest one.

Constraints

$1 \le n \le 2 \cdot 10^5$

$1 \le a, b \le n$

Sample Input 1 Copy

```
5
1 2
2 3
3 4
3 5
```

C++14 | 00:00:00 | 12 px

```
int n;
vector<int>parent;
vector<int>sub_size;
vector<vector<int>>tree;
vector<int>centroids;
void bfs(int node,int par){
    parent[node]=par;
    sub_size[node]=1;
    bool valid=true;
    for(auto v:tree[node]){
        if(v!=par){
            bfs(v,node);
            sub_size[node]+=sub_size[v];
            if(sub_size[v]>n/2)valid=false;
        }
    }
    if(n-sub_size[node]>n/2)valid=false;
    if(valid)centroids.emplace_back(node);
}
int main(){
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    cin>>n;
    tree.resize(n+1);
    sub_size.resize(n+1);
```

Sample Tests | Manual Tests

Test Case 1

ACCEPTED

Console | Run on Sample