#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

const int MAXN = 1e5 + 10;

vector<int> tree[MAXN];

int subtree\_size[MAXN];

// 计算子树大小

void dfs\_subtree(int u, int parent) {

    subtree\_size[u] = 1;

    for (int v : tree[u]) {

        if (v != parent) {  // 避免访问父节点

            dfs\_subtree(v, u);

            subtree\_size[u] += subtree\_size[v];

        }

    }

}

// 找到重心

int find\_centroid(int u, int parent, int total\_size) {

    for (int v : tree[u]) {

        if (v != parent && subtree\_size[v] > total\_size / 2) {

            return find\_centroid(v, u, total\_size);

        }

    }

    return u;

}

// 处理当前子树

void solve(int u) {

    // 在这里处理当前子树的问题

    // 例如，统计路径信息等

}

// 点分治主函数

void centroid\_decomposition(int u, int parent) {

    dfs\_subtree(u, -1);  // 计算子树大小

    int centroid = find\_centroid(u, -1, subtree\_size[u]);  // 找到重心

    // 处理当前重心

    solve(centroid);

    // 递归处理子树

    for (int v : tree[centroid]) {

        if (v != parent) {  // 避免访问父节点

            centroid\_decomposition(v, centroid);

        }

    }

}

int main() {

    int n;

    cin >> n;

    for (int i = 1; i < n; ++i) {

        int u, v;

        cin >> u >> v;

        tree[u].push\_back(v);

        tree[v].push\_back(u);

    }

    centroid\_decomposition(1, -1);  // 从根节点开始点分治，初始 parent 为 -1

    return 0;

}