#include <cctype>

#include <chrono>

#include <cstdio>

#include <random>

#include <set>

#include <vector>

typedef unsigned long long ull;

const ull mask = std::chrono::steady\_clock::now().time\_since\_epoch().count();

ull shift(ull x) {

x ^= mask;

x ^= x << 13;

x ^= x >> 7;

x ^= x << 17;

x ^= mask;

return x;

}

const int N = 1e6 + 10;

int n;

ull hash[N];

std::vector<int> edge[N];

std::set<ull> trees;

void getHash(int x, int p) {

hash[x] = 1;

for (int i : edge[x]) {

if (i == p) {

continue;

}

getHash(i, x);

hash[x] += shift(hash[i]);

}

trees.insert(hash[x]);

}

int main() {

scanf("%d", &n);

for (int i = 1; i < n; i++) {

int u, v;

scanf("%d%d", &u, &v);

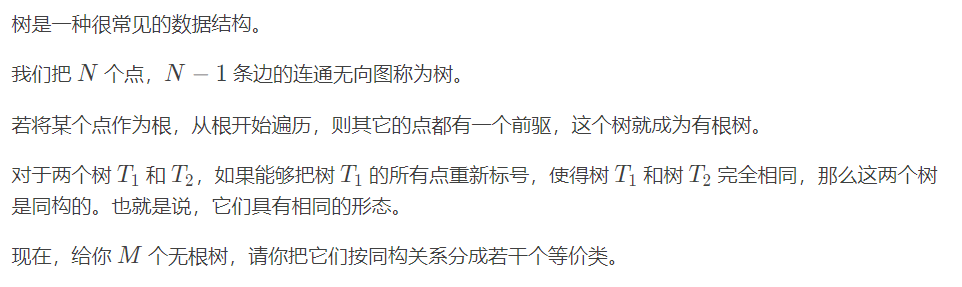
edge[u].push\_back(v);

edge[v].push\_back(u);

}

getHash(1, 0);

printf("%lu", trees.size());

}#include <chrono>

#include <cstdio>

#include <map>

#include <random>

#include <set>

#include <vector>

typedef unsigned long long ull;

const int N = 60, M = 998244353;

const ull mask = std::chrono::steady\_clock::now().time\_since\_epoch().count();

ull shift(ull x) {

x ^= mask;

x ^= x << 13;

x ^= x >> 7;

x ^= x << 17;

x ^= mask;

return x;

}

std::vector<int> edge[N];

ull sub[N], root[N];

std::map<ull, int> trees;

void getSub(int x) {

sub[x] = 1;

for (int i : edge[x]) {

getSub(i);

sub[x] += shift(sub[i]);

}

}

void getRoot(int x) {

for (int i : edge[x]) {

root[i] = sub[i] + shift(root[x] - shift(sub[i]));

getRoot(i);

}

}

int main() {

int m;

scanf("%d", &m);

for (int t = 1; t <= m; t++) {

int n, rt = 0;

scanf("%d", &n);

for (int i = 1; i <= n; i++) {

int fa;

scanf("%d", &fa);

if (fa) {

edge[fa].push\_back(i);

} else {

rt = i;

}

}

getSub(rt);

root[rt] = sub[rt];

getRoot(rt);

ull hash = 1;

for (int i = 1; i <= n; i++) {

hash += shift(root[i]);

}

if (!trees.count(hash)) {

trees[hash] = t;

}

printf("%d\n", trees[hash]);

for (int i = 1; i <= n; i++) {

edge[i].clear();

}

}

}