**----------------------------------- Tree\_order\_statistics (STL)------------------------------------**

#include <ext/pb\_ds/assoc\_container.hpp>

#include <ext/pb\_ds/trie\_policy.hpp>

using namespace \_\_gnu\_pbds;

typedef tree<

int,

null\_type,

less<int>,

rb\_tree\_tag,

tree\_order\_statistics\_node\_update > ordered\_set;

If we want to get map but not the set, as the second argument type must be used mapped type. Apparently, the tree supports the same operations as the set (at least I haven't any problems with them before), but also there are two new features — it is find\_by\_order() and order\_of\_key(). The first returns an iterator to the k-th largest element (counting from zero), the second — the number of items in a set that are strictly smaller than our item. Example of use:

ordered\_set X;

X.insert(1);

X.insert(2);

X.insert(4);

X.insert(8);

X.insert(16);

cout<<\*X.find\_by\_order(1)<<endl; // 2

cout<<\*X.find\_by\_order(2)<<endl; // 4

cout<<\*X.find\_by\_order(4)<<endl; // 16

cout<<(end(X)==X.find\_by\_order(6))<<endl; // true

cout<<X.order\_of\_key(-5)<<endl; // 0

cout<<X.order\_of\_key(1)<<endl; // 0

cout<<X.order\_of\_key(3)<<endl; // 2

cout<<X.order\_of\_key(4)<<endl; // 2

cout<<X.order\_of\_key(400)<<endl; // 5

**----------------------------------- Tree\_order\_statistics (STL) ------------------------------------**

**1**

**------------------------------------------------ Rabin Miller --------------------------------------------**

ll gcd(ll a, ll b){

return (b == 0 ? a : gcd(b, a % b));

}

ll mul(ull a, ull b, ull mod) {

ll ret = 0;

for (a %= mod, b %= mod; b != 0; b >>= 1, a <<= 1, a = a >= mod ? a - mod : a) {

if (b&1) {

ret += a;

if (ret >= mod) ret -= mod;

}

}

return ret;

}

void exgcd(ll x, ll y, ll &g, ll &a, ll &b) {

if (y == 0)

g = x, a = 1, b = 0;

else

exgcd(y, x%y, g, b, a), b -= (x/y) \* a;

}

ll llgcd(ll x, ll y) {

if (x < 0) x = -x;

if (y < 0) y = -y;

if (!x || !y) return x + y;

ll t;

while (x%y)

t = x, x = y, y = t%y;

return y;

}

ll inverse(ll x, ll p) {

ll g, b, r;

exgcd(x, p, g, r, b);

if (g < 0) r = -r;

return (r%p + p)%p;

}

**2**

ll mpow(ll x, ll y, ll mod) {

ll ret = 1;

while (y) {

if (y&1)

ret = (ret \* x)%mod;

y >>= 1, x = (x \* x)%mod;

}

return ret % mod;

}

ll mpow2(ll x, ll y, ll mod) {

ll ret = 1;

while (y) {

if (y&1)

ret = mul(ret, x, mod);

y >>= 1, x = mul(x, x, mod);

}

return ret % mod;

}

bool isPrime(ll p) { // implements by miller-babin

if (p < 2 || !(p&1)) return 0;

if (p == 2) return 1;

ll q = p-1, a, t;

int k = 0, b = 0;

while (!(q&1)) q >>= 1, k++;

for (int it = 0; it < 2; it++) {

a = rand()%(p-4) + 2;

t = mpow2(a, q, p);

b = (t == 1) || (t == p-1);

for (int i = 1; i < k && !b; i++) {

t = mul(t, t, p);

if (t == p-1)

b = 1;

}

if (b == 0)

return 0;

}

return 1;

}

**------------------------------------------------ Rabin Miller --------------------------------------------**

**3**

**------------------------------------------------ Pollard Rho ---------------------------------------------**

ll pollard\_rho(ll n){

if(n % 2 == 0) return 2;

ll x = rand() % n;

ll c = (rand() % (n - 1)) + 1;

ll d = 1;

ll y = x;

ll i = 1, k = 2;

while(1){

x = (mul(x, x, n) - c + n) % n;

i++;

ll d = gcd(abs(x - y), n);

if(d > 1 && d < n) return d;

if(y == x) return pollard\_rho(n);

if (i == k){

y = x;

k <<= 1;

}

}

return d;

}

**------------------------------------------------ Pollard Rho ---------------------------------------------**

**------------------------------------------------ GCD Extended -----------------------------------------**

ll extendedGCD(ll a, ll b, ll& x, ll& y) {

if(!b){

x = 1, y = 0;

return a;

}

ll g = extendedGCD(b, a % b, y, x);

y = (g - (a \* x)) / b;

return g;

}

**------------------------------------------------ GCD Extended -----------------------------------------**

**------------------------------------------------ Mo algorithm ------------------------------------------**

struct query{

int l;

int r;

int id;

}mo[MAXq];

sort(mo, mo + q, [](query& a, query& b){return (a.l / TB != b.l / TB ? a.l < b.l : a.r < b.r);});

**4**

int l = 1;

int r = 0;

for(int i = 0; i < q; i++){

int nl = mo[i].l;

int nr = mo[i].r;

while(r < nr) funt(arr[++r], 1);

while(l < nl) funt(arr[l++], -1);

while(r > nr) funt(arr[r--], -1);

while(l > nl) funt(arr[--l], 1);

sol[mo[i].id] = num;

}

**------------------------------------------------ Mo algorithm ------------------------------------------**

**------------------------------------------------ Teorema de Pick --------------------------------------**

Sea un polígono simple cuyos vértices tienen coordenadas enteras. Si B es el número de puntos enteros en el borde, I el número de puntos enteros en el interior del polígono, entonces el área A del polígono se puede calcular con la fórmula:

A = I + B/2 - 1.

**------------------------------------------------ Teorema de Pick --------------------------------------**

**------------------------------------------------ Números de catalán ---------------------------------**

//Propiedades

The number of correct bracket sequences consisting of n opening and n closing brackets.

The number of binary trees with root n +1 leaves (vertices are numbered).

Number of ways to separate completely the brackets n +1 the multiplier.

The number of triangulations of a convex n +2 -Gon (ie, the number of partitions of a polygon into triangles by nonintersecting diagonals).

Number of ways to connect 2n points on a circle n disjoint chords.

The number of non-isomorphic full binary trees with n internal nodes (ie having at least one son).

The number of monotonic paths from the point of (0,0) the point (N, n) in a square lattice size n \ times n , Not rising above the main diagonal.

The number of permutations of length n You can sort the stack (it can be shown that the permutation is sorted stack if and only if there is no such index i <j <k That a\_k <a\_i <a\_j .)

The number of partitions of a set of continuous n elements (ie, partitions into continuous blocks).

Amount ways to cover the ladder 1 \ ldots n with n rectangles (referring to the figure, consisting of n columns i Th of which has a height i .)

C[n] => FOR(k=0,n-1) C[k] \* C[n-1-k]

C[n] => Comb(2\*n,n) / (n + 1)

C[n] => 2\*(2\*n-3)/n \* C[n-1]

**------------------------------------------------ Números de catalán ---------------------------------**

**5**

**------------------------------------------------ Suffix Array --------------------------------------------**

int n;

int CS[MAXn];

int SA[MAXn];

int LCP[MAXn];

int pot;

bool cmp(int a,int b){

if(CS[a] != CS[b])

return CS[a] < CS[b];

a += pot;

b += pot;

return (mx(a,b) <= n ? CS[a] < CS[b] : a > b);

}

void suffix\_array(string &S){

vector < int > aux(n + 1);

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

SA[i] = i, CS[i] = S[i];

for(int k = 0; (1 << k) <= n; k++){

pot = (1 << k);

stable\_sort(SA + 1, SA + n + 1, cmp);

for(int i = 2; i <= n; i++)

aux[i] = aux[i - 1] + cmp(SA[i - 1],SA[i]);

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

CS[SA[i]] = aux[i];

if(aux[n] == n)

break;

}

}

void lcp(string &S){

vector < int > aux(n + 1);

**6**

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

aux[SA[i]] = i;

for(int i = 1,k = 0; i <= n; i++){

int j = SA[aux[i] - 1];

while(mx(j + k,i + k) <= n && S[i + k] == S[j + k]) k++;

LCP[aux[i]] = k;

if(k) k--;

}

}

**------------------------------------------------ Suffix Array --------------------------------------------**

**------------------------------------------------ Rotar un punto counterclockwise ---------------**

inline double to\_radian(double ang) {

return PI \* ang / 180;

}

inline void rotate(double xx, double yy, double& nx, double& ny, double ang) {

nx = xx \* cos(to\_radian(ang)) - yy \* sin(to\_radian(ang));

ny = xx \* sin(to\_radian(ang)) + yy \* cos(to\_radian(ang));

}

**------------------------------------------------ Rotar un punto counterclockwise ---------------**

**------------------------------------------------ SCC\_Tarjan ---------------------------------------------**

const int MAXn = 100;

int n,m;

bool mark[MAXn];

vector < int > adya[MAXn+1];

stack < int > pila;

int vdfs[MAXn+1];

int vlowlink[MAXn+1];

int cont;

void dfs(int v){

pila.push(v);

vdfs[v] = vlowlink[v] = v;

for(auto to: adya[v]){

if(!vdfs[to]){

**7**

dfs(to);

vlowlink[v] = mn(vlowlink[v],vlowlink[to]);

}

else if(!mark[to])

vlowlink[v] = mn(vlowlink[v],vdfs[to]);

}

if (vdfs[v] == vlowlink[v]){

cont++;

cout<<"componente numero "<<cont<<"\n";

while(!mark[v]){

cout<<pila.top()<<" ";

mark[pila.top()] = true;

pila.pop();

}

cout<<"\n";

}

}

**------------------------------------------------ SCC\_Tarjan ---------------------------------------------**

**------------------------------------------------ KMP ------------------------------------------------------**

const int SIZE = 1e5 + 1;

int TF[SIZE];

int TS = S.size(),TP = P.size();

S = " " + S + " ";

P = " " + P + " ";

TF[1] = 0;

for(int i = 2;i <= TP;i++){

int aux = TF[i - 1];

while(aux > 0 && P[aux + 1] != P[i]) aux = TF[aux];

if(P[aux + 1] == P[i]) aux++;

TF[i] = aux;

}

int sol = 0,aux = 0;

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

while(aux > 0 && P[aux + 1] != S[i]) aux = TF[aux];

**8**

if(P[aux + 1] == S[i]) aux++;

if(aux == TP) sol++;

}

cout << sol << '\n';

**------------------------------------------------ KMP ------------------------------------------------------**

**------------------------------------------------ Articulation Points -----------------------------------**

int sol = 0;

void DFS(int v, int p){

vdfs[v] = vlow[v] = ++cont;

int cant = 0;

bool yes = false;

for(auto to: adya[v]){

if(to == p) continue;

if(!vdfs[to]){

DFS(to, v);

cant++;

vlow[v] = mn(vlow[v], vlow[to]);

if(v != 1 && vlow[to] >= vdfs[v]) yes = true;

}

else

vlow[v] = mn(vlow[v], vdfs[to]);

}

if(v == 1){

if(cant > 1) sol++;

return;

}

if(yes) sol++;

}

**------------------------------------------------ Articulation Points -----------------------------------**

**------------------------------------------------ Heavy Light Decomposition -----------------------**

#include <bits/stdc++.h>

#define mn(a, b) (a < b ? a : b)

#define mx(a, b) (a > b ? a : b)

#define f first

#define s second

#define all(v) (v).begin(), (v).end()

#define base 331

using namespace std;

**9**

typedef long long ll;

typedef unsigned long long ull;

typedef pair < int , int > par;

int const MAXn = 1e5 + 2, loga = log2(MAXn) + 2;

int CI[MAXn];

int CH[MAXn];

int subtree[MAXn];

vector < int > adya[MAXn];

int INV[MAXn];

int tree[MAXn \* 6];

int cn;

int pa[MAXn][loga];

int lvl[MAXn];

int sz;

void update(int nod, int l, int r, int pos, int val){

if(l > r || l > pos || r < pos)

return;

if(l == r){

tree[nod] += val;

return;

}

int mit = (l + r) >> 1;

update(nod << 1, l, mit, pos, val);

update(nod << 1 | 1, mit + 1, r, pos, val);

tree[nod] = mx(tree[nod << 1], tree[nod << 1 | 1]);

}

int query(int nod, int l, int r, int a, int b){

if(l > r || l > b || r < a)

return 0;

if(l >= a && r <= b)

return tree[nod];

int mit = (l + r) >> 1;

return max(query(nod << 1, l, mit, a, b), query(nod << 1 | 1, mit + 1, r, a, b));

}

void HLD(){

stack < par > pila;

**10**

pila.push({1, ++cn});

while(!pila.empty()){

par v = pila.top();

pila.pop();

if(CH[v.s] == -1) CH[v.s] = v.f;

CI[v.f] = v.s;

INV[v.f] = ++sz;

int ind = -1, may = -1;

for(auto to: adya[v.f]){

if(to == pa[v.f][0]) continue;

if(subtree[to] > may){

may = subtree[to];

ind = to;

}

}

for(auto to: adya[v.f]){

if(to == pa[v.f][0] || to == ind) continue;

pila.push({to, ++cn});

}

if(ind != -1) pila.push({ind, v.s});

}

}

void BFS(){

queue < int > cola;

cola.push(1);

stack < int > pila;

pila.push(1);

while(!cola.empty()){

int v = cola.front();

cola.pop();

for(auto to: adya[v]){

if(lvl[to]) continue;

lvl[to] = lvl[v] + 1;

pa[to][0] = v;

cola.push(to);

pila.push(to);

}

}

while(!pila.empty()){

int v = pila.top();

**11**

pila.pop();

subtree[v]++;

subtree[pa[v][0]] += subtree[v];

}

}

void build\_lca(int n){

for(int e = 1; e < loga; e++){

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

pa[i][e] = pa[pa[i][e - 1]][e - 1];

}

}

int LCA(int a, int b){

if(lvl[a] < lvl[b])

swap(a, b);

for(int i = loga - 1; i >= 0; i--){

if(lvl[pa[a][i]] >= lvl[b])

a = pa[a][i];

}

if(a == b) return a;

for(int i = loga - 1; i >= 0; i--){

if(pa[a][i] != pa[b][i]){

a = pa[a][i];

b = pa[b][i];

}

}

a = pa[a][0];

return a;

}

int query\_up(int u, int v){

int cv = CI[v];

int cu;

int sol = 0;

while(1){

cu = CI[u];

if(cu == cv){

sol = max(sol, query(1, 1, sz, INV[v], INV[u]));

break;

}

**12**

int head = CH[cu];

sol = max(sol, query(1, 1, sz, INV[head], INV[u]));

u = pa[head][0];

}

return sol;

}

int main(){

ios\_base::sync\_with\_stdio(0);

cin.tie(0);

cout.tie(0);

int n, q;

cin>>n;

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

int n1, n2;

cin>>n1>>n2;

adya[n1].push\_back(n2);

adya[n2].push\_back(n1);

}

lvl[1] = 1;

BFS();

build\_lca(n);

cn = 1;

fill(CH + 1, CH + n + 1, -1);

HLD();

cin>>q;

while(q--){

char x2;

int a1, b1;

cin>>x2>>a1>>b1;

if(x2 == 'I')

update(1, 1, sz, INV[a1], b1);

else{

int lca = LCA(a1, b1);

cout<<max(query\_up(a1, lca), query\_up(b1, lca))<<"\n";

}

}

return 0;

}

**------------------------------------------------ Heavy Light Decomposition -----------------------**

**13**

**------------------------------------------------ Centroid Decomposition ---------------------------**

#include <bits/stdc++.h>

#define mn(a, b) (a < b ? a : b)

#define mx(a, b) (a > b ? a : b)

#define f first

#define s second

#define all(v) (v).begin(), (v).end()

#define base 331

using namespace std;

typedef long long ll;

typedef unsigned long long ull;

int const MAXn = 1e5 + 2;

vector < int > adya[MAXn];

int subtr[MAXn];

int lvl[MAXn];

int padre[MAXn];

void DFS(int v, int p){

subtr[v] = 1;

for(auto to: adya[v]){

if(p == to || lvl[to]) continue;

DFS(to, v);

subtr[v] += subtr[to];

}

}

int find\_root(int v, int p, int w){

for(auto to: adya[v]){

if(to == p || lvl[to]) continue;

if(subtr[to] > w / 2)

return find\_root(to, v, w);

}

return v;

}

void find\_centroid\_decomposition(int v, int level, int p){

DFS(v, -1);

**14**

int root = find\_root(v, -1, subtr[v]);

lvl[root] = level;

padre[root] = p;

for(auto to: adya[root]){

if(!lvl[to])

find\_centroid\_decomposition(to, level + 1, root);

}

}

int main(){

//freopen(".in","r",stdin);

ios\_base::sync\_with\_stdio(0);

cin.tie(0);

cout.tie(0);

int n;

cin>>n;

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

int n1, n2;

cin>>n1>>n2;

adya[n1].push\_back(n2);

adya[n2].push\_back(n1);

}

find\_centroid\_decomposition(1, 1, -1);

return 0;

}

**------------------------------------------------ Centroid Decomposition ---------------------------**