Team notebook

October 25, 2024

Contents

1 Geometry

1.1 icpc

```
/*
ID: sahajrastogi
LANG: C++11
*/

#include <iostream>
#include <bits/stdc++.h>
#include <unordered_set>
// #include <ext/pb_ds/assoc_container.hpp>
// #include <ext/pb_ds/tree_policy.hpp>

typedef long long ll;

using namespace std;
//using namespace __gnu_pbds;
#define ordered_set tree<int, null_type,less<int>,
    rb_tree_tag,tree_order_statistics_node_update>
#define pb push_back
#define f first
#define s second
#define int int64_t
#define pi pair<int,int>
#define pf pair<float,float>


signed main(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    #ifndef ONLINE_JUDGE
    freopen("file.txt", "r", stdin);
    #endif



}
```

2 Graphs

2.1 icpc

```
/*
ID: sahajrastogi
LANG: C++11
*/

#include <iostream>
#include <bits/stdc++.h>
#include <unordered_set>
// #include <ext/pb_ds/assoc_container.hpp>
// #include <ext/pb_ds/tree_policy.hpp>

typedef long long ll;

using namespace std;
//using namespace __gnu_pbds;
#define ordered_set tree<int, null_type,less<int>,
    rb_tree_tag,tree_order_statistics_node_update>
#define pb push_back
#define f first
#define s second
#define int int64_t
#define pi pair<int,int>
#define pf pair<float,float>


signed main(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    #ifndef ONLINE_JUDGE
    freopen("file.txt", "r", stdin);
    #endif



}
```

3 Math

3.1 icpc

```
/*
ID: sahajrastogi
LANG: C++11
*/

#include <iostream>
#include <bits/stdc++.h>
#include <unordered_set>
// #include <ext/pb_ds/assoc_container.hpp>
// #include <ext/pb_ds/tree_policy.hpp>

typedef long long ll;

using namespace std;
//using namespace __gnu_pbds;
#define ordered_set tree<int, null_type,less<int>,
    rb_tree_tag,tree_order_statistics_node_update>
#define pb push_back
#define f first
#define s second
#define int int64_t
#define pi pair<int,int>
#define pf pair<float,float>


signed main(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    #ifndef ONLINE_JUDGE
    freopen("file.txt", "r", stdin);
    #endif



}
```

4 Range Query

4.1 BIT

```cpp
#include <bits/stdc++.h>
using namespace std;
int sum(int i, vector<int> &bit){
    int res = 0; while(i>=0) res+=bit[i]; i=((i+1)&i)-1; return res;
}
void upd(int i, int wt, vector<int> &bit){
    while(i<bit.size()) bit[i]+=wt; i=(i+1)|i;
}
int range(int a, int b,vector<int>&bit){
    if(a == 0) return sum(b,bit); // care for indexing
    return sum(b,bit) - sum(a-1,bit);
}
```

4.2 SEGTREEBigStepper

```cpp
#include <bits/stdc++.h>
using namespace std;
template <class T> struct SegTree { // cmb(ID,b) = b
        const T ID{0};
        T cmb(T a, T b) { }
        int n; vector<T> seg;
        void init(int _n) { // upd, query also work if n = _n
                for (n = 1; n < _n; ) n *= 2;
                seg.assign(2*n,ID);
    }
        void pull(int p) {
        seg[p] = cmb(seg[2*p],seg[2*p+1]);
    }
        void upd(int p, T val) { // set val at position p
                seg[p += n] += val;
        for (p /= 2; p; p /= 2) pull(p);
        }
        T query(int l, int r) { // zero-indexed, inclusive
                T ra = ID, rb = ID;
                for (l += n, r += n+1; l < r; l /= 2, r /= 2) {
                        if (l&1) ra = cmb(ra,seg[l++]);
                        if (r&1) rb = cmb(seg[--r],rb);
                }
                return cmb(ra,rb);
        }

        int bSearch(int target){
                int p = 1;
                if(seg[p] < target) return 0;
                while(p < n){
                        if(seg[2*p] < target){
                                p = 2*p+1;
                        } else {
                                p = 2*p;
                        }
                }
                return p-n+1;
        }
        // int first_at_least(int lo, int val, int ind, int l, int r) { //
            if seg stores max across range
        //      if (r < lo || val > seg[ind]) return -1;
        //      if (l == r) return l;
        //      int m = (l+r)/2;
        //      int res = first_at_least(lo,val,2*ind,l,m); if (res != -1)
            return res;
        //      return first_at_least(lo,val,2*ind+1,m+1,r);
        // }
};
```

4.3 SEGTREELazy

```cpp
#include <bits/stdc++.h>
struct Node{
        bool isID = false;
        int sum =0;
        Node(bool x, int s) : isID(x), sum(s){}

};

struct lNode{
        bool isID = false;
        int m=1;
        int c=0;
        lNode(bool x) : isID(x){}
```

```
};

Node idnode(true,0);
lNode lazynode(true);
template <class T, class Q> struct SegTree { // cmb(ID,b) = b
	const T ID{idnode}; const Q IDQ{lazynode};
	T cmb(T a, T b) {
	// if(a.isID) return b;
		// if(b.isID) return a;
		Node res(false,0);
		res.sum = (a.sum+b.sum)%mod;
		return res;
    }

	Q lazycmb(Q a, Q b){
		if(a.isID) return b;
		if(b.isID) return a;
		lNode res(false);
		res.m=(a.m*b.m)%mod;
		res.c=(a.m*b.c + a.c)%mod;
		return res;
	}

	// void cmbTQ(T a, Q b){
	//	if(b.isID) return;
	//	if(a.isID) {

	//	}
	// }
	int n; vector<T> seg; vector<Q> lazy;
	void init(int _n) { // upd, query also work if n = _n
		for (n = 1; n < _n; ) n *= 2;
		seg.assign(2*n,ID);
		lazy.assign(2*n,IDQ);
    }


	void printTree(){
		for(int i=1;i<2*n;i++){
			cout << seg[i].sum << " ";
		}
		cout << "\n";
	}
	void push(int node, int l, int r){
		seg[node].sum = ((seg[node].sum*lazy[node].m)%mod +
		    (lazy[node].c*(r-l+1))%mod)%mod; // operation dependent
		if(l != r){
			lazy[2*node] = lazycmb(lazy[node],lazy[2*node]);
			lazy[2*node+1] = lazycmb(lazy[node],lazy[2*node+1]);
		}
		lazy[node] = IDQ;
    }
    void pull(int p) {
    seg[p] = cmb(seg[2*p],seg[2*p+1]);
}

    void upd(int l, int r, Q val){
		upd(l,r,val,0,n-1,1);
    }

    void upd(int l, int r, Q val, int start, int end, int node) {
		push(node,start,end);
		if(r < start || l > end) return; // maybe not needed

		if(l <= start && end <= r){
			lazy[node] = val;
			push(node,start,end);
			return;
		}
		int mid = (start + end)/2;
		//if(start <=l && r <= mid){
			upd(l,r,val,start,mid,2*node);
		//} else {
			upd(l,r,val,mid+1,end,2*node+1);
		//}
		pull(node);
}

    T query(int l, int r){
		return query(l,r,0,n-1,1);
    }
    T query(int l, int r, int start, int end, int node) { //
	    zero-indexed, inclusive
		push(node,start,end);
		if(r < start || l > end){
			return ID;
		}
		if(l <= start && end <= r){
			return seg[node];
```

```
		} else {
			int mid = (start + end)/2;
			T x = query(l,r, start, mid,2*node);
			T y = query(l,r, mid+1, end,2*node+1);
			return cmb(x,y);
		}
	}
};
```

4.4 SEGTREERecursive

```
#include <bits/stdc++.h>
template <class T> struct SegTree { // cmb(ID,b) = b
	const T ID{0}; T cmb(T a, T b) {
	if(a == ID){
			return b;
		}
		if(b == ID){
			return a;
		}
		return min(a,b);
    }

	int n; vector<T> seg;
	void init(int _n) { // upd, query also work if n = _n
		for (n = 1; n < _n; ) n *= 2;
		seg.assign(2*n,ID);
    }
	void pull(int p) {
	seg[p] = cmb(seg[2*p],seg[2*p+1]);
    }
	void upd(int p,T val) upd(p, val,0,n-1,1);
	void upd(int p, T val, int start, int end, int node) { // set val
	    at position p

		if(p < start || p > end) return; // maybe not needed

		if(start == end){
			seg[node] = val;
			return;
		}

		int mid = (start + end)/2;
		if(start <=p && p <= mid){
			upd(p,val,start,mid,2*node);
		} else {
			upd(p,val,mid+1,end,2*node+1);
		}
		pull(node);
    }

	T query(int l, int r) query(l,r,1,0,n-1)
	T query(int l, int r, int node, int start, int end) { //
	    zero-indexed, inclusive

		if(r < start || l > end){
			return ID;
		}
		if(l <= start && end <= r){
			return seg[node];
		} else {
			int mid = (start + end)/2;
			T x = query(l,r,2*node, start, mid);
			T y = query(l,r,2*node+1, mid+1, end);
			return cmb(x,y);
		}
	}
};
```

5 Trees

5.1 LCA

```
#include <bits/stdc++.h>
#define pb push_back
using namespace std;

int n; int q;
int par[200005][21];
int depth[200005];
vector<int> adj[200005];

void buildArr(int node, int p){
    par[node][0] = p;
    for(int i=1;i<20;i++){
```

```
        if(par[node][i-1] != -1){
            par[node][i] = par[par[node][i-1]][i-1];
        }
    }
    if(p == -1) depth[node] = 0;
    else depth[node] = depth[p] + 1;

    for(auto x : adj[node]){
        if(x == p) continue;
        buildArr(x,node);
    }
}


int bigStepper(int node, int k){
    int x = 0;
    for(int i=0;i<20;i++){
        if(k%2==1) node = par[node][i];
        k /= 2;
    }
    return node;
}


int lca(int a, int b){
    if (depth[a] > depth[b]) swap(a,b);

    b = bigStepper(b,depth[b] - depth[a]);
    //cout << b;
    if(a == b) return a;
    for(int i=19;i>=0;i--){
        if(par[a][i] != par[b][i]){
            a = par[a][i];
            b = par[b][i];
        }
    }
    return par[a][0];
}
signed main(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    #ifndef ONLINE_JUDGE
    freopen("file.txt", "r", stdin);
    #endif
    cin >> n >> q;

    for(int i=0;i<=n;i++){
        for(int j=0;j<20;j++){
            par[i][j] = -1;
        }
    }

    for(int i=0;i<n-1;i++){
        int a; int b; cin >> a >> b;
        adj[a].pb(b);
        adj[b].pb(a);
    }

    buildArr(1,-1);
    for(int i =0;i<q;i++){
        int a; int b; cin >> a >>b;
        cout << depth[a] + depth[b] - 2*depth[lca(a,b)] << "\n";
    }
}
```