Team notebook

October 31, 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 DINIC

```
#include <bits/stdc++.h>


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 ll
#define pi pair<int,int>
#define pf pair<float,float>

struct Dinic {   // flow template
using F = ll; // flow type
struct Edge {
        int to;
        F flo, cap;
};
int N;
vector<Edge> eds;
vector<vector<int>> adj;
void init(int _N) {
        N = _N;
        adj.resize(N), cur.resize(N);
}
/// void reset() { trav(e,eds) e.flo =
    0; }
void ae(int u, int v, F cap, F rcap =
    0) {
        assert(min(cap, rcap) >= 0);
        adj[u].pb((eds).size());
        eds.pb({v, 0, cap});
        adj[v].pb(eds.size());
        eds.pb({u, 0, rcap});
}
vector<int> lev;
vector<vector<int>::iterator> cur;
bool bfs(int s, int t) { // level =
    shortest distance from source
        lev = vector<int>(N, -1);
        for(int i=0;i<N;i++) cur[i] =
            begin(adj[i]);
        queue<int> q({s});
        lev[s] = 0;
        while (q.size()) {
                int u = q.front();
                q.pop();
                for (auto e : adj[u]) {
                        const Edge &E =
                            eds[e];
                        int v = E.to;
                        if (lev[v] < 0 &&
                            E.flo <
                            E.cap)
                            q.push(v),
                            lev[v] =
                            lev[u] + 1;
                }
        }
        return lev[t] >= 0;
}
F dfs(int v, int t, F flo) {
        if (v == t) return flo;
        for (; cur[v] != end(adj[v]);
            cur[v]++) {
                Edge &E = eds[*cur[v]];
                if (lev[E.to] != lev[v]
                    + 1 || E.flo ==
                    E.cap) continue;
                F df = dfs(E.to, t,
                    min(flo, E.cap -
                    E.flo));
                if (df) {
                        E.flo += df;
                        eds[*cur[v] ^
                            1].flo -= df;
                        return df;
                } // saturated >=1 one
                    edge
        }
        return 0;
}
F maxFlow(int s, int t) {
        F tot = 0;
        while (bfs(s, t))
                while (F df = dfs(s, t,
                    numeric_limits<F>::max()))
                    tot += df;
        return tot;
}
};

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

    // #ifndef ONLINE_JUDGE
    // freopen("file.txt", "r", stdin);
    // #endif
    int l; int r; int n; cin>>l>>r>>n;
    Dinic d;
    d.init(l+r+2);
    for(int i=0;i<n;i++){
        int a; int b; cin>>a>>b;
        d.ae(a+1,l+b+1,1);
    }
    for(int i=0;i<l;i++){
        d.ae(0,i+1, 1);
    }
    for(int i=0;i<r;i++){
        d.ae(i+1+l,l+r+1, 1);
    }

    cout<< d.maxFlow(0, l+r+1)<< "\n";
    d.bfs(0,l+r+1);

    for(int i=1;i<=l;i++){
        for(int v:d.adj[i]){
            if(d.eds[v].cap==0) continue;
            if(d.eds[v].cap==d.eds[v].flo)
                cout<< i-1 <<" "<<
                d.eds[v].to-l-1<<"\n";
        }
    }
    return 0;


}
```

2.2 DSU$_{size}$

```
#include<bits/stdc++.h>
using namespace std;
int parent[1];//fill
int sz[1]; //fill
void make_set(int v) {
    parent[v] = v;
    sz[v] = 1;
}
int find_set(int v) {
    if (v == parent[v])
        return v;
    return parent[v] = find_set(parent[v]);
}
void union_sets(int a, int b) {
    a = find_set(a);
    b = find_set(b);
    if (a != b) {
        if (sz[a] < sz[b])
            swap(a, b);
        parent[b] = a;
        sz[a] += sz[b];
    }
}
```

2.3 HUNGRY

```
/*
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 pi pair<int,int>
#define f first
#define s second
#define int int64_t


int ckmin(int &a, int b) { return a > b ? ((a
    = b), true) : false; }

/**
 * @return the jobs of each worker in the
     optimal assignment,
 * or -1 if the worker is not assigned
 */
template <class T> vector<int>
    hungarian(const vector<vector<T>> &C) {
       int J = C.size();
       int W = C[0].size();
       assert(J <= W);

       // job[w] = job assigned to w-th
           worker, or -1 if no job assigned
       // note: a W-th worker was added for
           convenience
       vector<int> job(W + 1, -1);
       vector<T> h(W); // Johnson potentials

       const T inf = numeric_limits<T>::max();
       // assign j_cur-th job using Dijkstra
           with potentials
       for (int j_cur = 0; j_cur < J;
           j_cur++) {
              int w_cur = W; // unvisited
                  worker with minimum distance
              job[w_cur] = j_cur;

              vector<T> dist(W + 1, inf); //
                  Johnson-reduced distances
              dist[W] = 0;
              vector<bool> vis(W + 1);  //
                  whether visited yet
              vector<int> prv(W + 1, -1); //
                  previous worker on shortest
                  path
              while (job[w_cur] != -1) { //
                  Dijkstra step: pop min
                  worker from heap
                     T min_dist = inf;
                     vis[w_cur] = true;
                     int w_next = -1; // next
                         unvisited worker
                         with minimum distance

                     // consider extending
                         shortest path by
                         w_cur -> job[w_cur]
                         -> w
                     for (int w = 0; w < W;
                         w++) {
                            if (!vis[w]) {
                                   // sum of
                                       reduced
                                       edge
                                       weights
                                       w_cur
                                       ->
                                       job[w_cur]
                                       -> w
                                   T edge =
                                       C[job[w_cur]]
                                       -
                                       h[w];
                                   if (w_cur
                                       != W)
                                       {
                                          edge
                                              -=
                                              C[job[
                                              -
                                              h[w_cu
                                          assert(edg
                                              >=
                                              0);
                                   }
                                   if
                                       (ckmin(dist[w
                                       dist[w_cur]
                                       +
                                       edge))
                                       {
                                       prv[w]
                                       =
                                       w_cur;
                                       }
                                   if
                                       (ckmin(min_di
                                       dist[w]))
                                       {
                                       w_next
                                       = w; }
                            }
                     }
```

```
                w_cur = w_next;
            }

            for (int w = 0; w < W; w++) {
                // update potentials
                ckmin(dist[w],
                    dist[w_cur]);
                h[w] += dist[w];
            }

            while (w_cur != W) { // update
                job assignment
                job[w_cur] =
                    job[prv[w_cur]];
                w_cur = prv[w_cur];
            }
        }

        return job;
}

signed main(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
       #ifndef ONLINE_JUDGE
    freopen("file.txt", "r", stdin);
    #endif
    int n;
    cin>> n;
    vector<vector<int>> table(n,
         vector<int>(n));
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            cin>>table[j][i];
        }
    }
    vector<int> sol = hungarian(table);
    int cost=0;
    for(int i=0;i<n;i++)
         cost+=table[sol[i]][i];
    cout<< cost<<"\n";
    for(int i=0;i<n;i++){
        cout << sol[i]+1<<" "<< i+1;
        cout<< "\n";
    }
}
```

2.4 KOSARAJU

```
#include <bits/stdc++.h>


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 ll
#define pi pair<int,int>
#define pf pair<float,float>

vector<int> adj[500005];
vector<int> adjr[500005];
int visited[500005]={0};
vector<int> order;
vector<int> scc[500005];
int k = 0;

void dfs(int x){
    visited[x] = 1;
    for(auto nex : adj[x]){
        if(!visited[nex])dfs(nex);
    }
    order.push_back(x);
}

void dfsr(int x){
    visited[x] = k;
    scc[k].pb(x);
    for(auto nex : adjr[x]){
        if(!visited[nex]) dfsr(nex);
    }
}

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

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

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

    k=0;

    for(int i=0;i<n;i++){
        if(!visited[i]) dfs(i);
    }

    reverse(order.begin(),order.end());
    for(int i=0;i<500003;i++) visited[i]=0;

    for(int x : order){
        if(!visited[x]){
            k++;
            dfsr(x);
        }
    }

    cout << k << "\n";
    for(int i=1;i<=k;i++){
        cout << scc[i].size();
        for(auto x : scc[i]){
            cout << " "<<x;
        }
        if(i!=k) cout << "\n";
    }
}
```

3 Math

3.1 BigInt

```
#include <bits/stdc++.h>
using namespace std;
const int BASE_LENGTH = 1;
const int BASE = (int) pow(10, BASE_LENGTH);
const int MAX_LENGTH = 100005;

string int_to_string(int i, int width, bool
     zero) {
    string res = "";
    while (width--) {
        if (!zero && i == 0) return res;
        res = (char)(i%10 + '0') + res;
        i /= 10;
    }
    return res;
}

struct bigint {
    int len;
    vector<int> s;

    bigint() {
        s.assign(0,MAX_LENGTH);
        len = 1;
    }

    bigint(unsigned long long num) {
        len = 0;
        while (num >= BASE) {
            s[len] = num % BASE;
            num /= BASE;
            len ++;
        }
        s[len++] = num;
    }

    bigint(const char* num) {
        int l = strlen(num);
        len = l/BASE_LENGTH;
        if (l % BASE_LENGTH) len++;
        int index = 0;
        for (int i = l - 1; i >= 0; i -=
             BASE_LENGTH) {
            int tmp = 0;
            int k = i - BASE_LENGTH + 1;
            if (k < 0) k = 0;
            for (int j = k; j <= i; j++) {
                tmp = tmp*10 + num[j] - '0';
            }
            s[index++] = tmp;
        }
    }

    void clean() {
        while(len > 1 && !s[len-1]) len--;
    }

    string str() const {
        string ret = "";
        if (len == 1 && !s[0]) return "0";
        for(int i = 0; i < len; i++) {
            if (i == 0) {
                ret += int_to_string(s[len - i
                     - 1], BASE_LENGTH, false);
            } else {
                ret += int_to_string(s[len - i
                     - 1], BASE_LENGTH, true);
            }
        }
        return ret;
    }

    unsigned long long ll() const {
        unsigned long long ret = 0;
        for(int i = len-1; i >= 0; i--) {
            ret *= BASE;
            ret += s[i];
        }
        return ret;
    }

    bigint operator + (const bigint& b) const {
        bigint c = b;
        while (c.len < len) c.s[c.len++] = 0;
        c.s[c.len++] = 0;
        bool r = 0;
        for (int i = 0; i < len || r; i++) {
            c.s[i] += (i<len)*s[i] + r;
            r = c.s[i] >= BASE;
            if (r) c.s[i] -= BASE;
        }
        c.clean();
        return c;
    }

    bigint operator - (const bigint& b) const {
        if (operator < (b)) throw "cannot do
             subtract";
        bigint c = *this;
        bool r = 0;
        for (int i = 0; i < b.len || r; i++) {
            c.s[i] -= b.s[i];
            r = c.s[i] < 0;
            if (r) c.s[i] += BASE;
        }
        c.clean();
        return c;
    }

    bigint operator * (const bigint& b) const {
        bigint c;
        c.len = len + b.len;
        for(int i = 0; i < len; i++)
            for(int j = 0; j < b.len; j++)
                c.s[i+j] += s[i] * b.s[j];
        for(int i = 0; i < c.len-1; i++){
            c.s[i+1] += c.s[i] / BASE;
            c.s[i] %= BASE;
        }
        c.clean();
        return c;
    }
    bigint operator / (const int b) const {
        bigint ret;
        int down = 0;
        for (int i = len - 1; i >= 0; i--) {
            ret.s[i] = (s[i] + down * BASE) / b;
            down = s[i] + down * BASE -
                 ret.s[i] * b;
        }
        ret.len = len;
        ret.clean();
        return ret;
    }

    bool operator < (const bigint& b) const {
        if (len < b.len) return true;
        else if (len > b.len) return false;
        for (int i = 0; i < len; i++)
            if (s[i] < b.s[i]) return true;
            else if (s[i] > b.s[i]) return
                 false;
        return false;
    }

    bool operator == (const bigint& b) const {
        return !(*this<b) && !(b<(*this));
    }

    bool operator > (const bigint& b) const {
        return b < *this;
    }
};


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

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

    int t; cin >> t;
    bigint a; bigint b;
    for(int i=0;i<t;i++){
        string s1; string s2;cin >> s1 >> s2;
```

```
    }
    
    
    
    
    
}
```

3.2 icpc

```
#include <bits/stdc++.h>
using namespace std;
const int BASE_LENGTH = 2;
const int BASE = (int) pow(10, BASE_LENGTH);
const int MAX_LENGTH = 100005;

string int_to_string(int i, int width, bool
     zero) {
    string res = "";
    while (width--) {
        if (!zero && i == 0) return res;
        res = (char)(i%10 + '0') + res;
        i /= 10;
    }
    return res;
}

struct bigint {
    int len, s[MAX_LENGTH];

    bigint() {
        memset(s, 0, sizeof(s));
        len = 1;
    }

    bigint(unsigned long long num) {
        len = 0;
        while (num >= BASE) {
            s[len] = num % BASE;
            num /= BASE;
            len ++;
        }
        s[len++] = num;
    }

    bigint(const char* num) {
        int l = strlen(num);
        len = l/BASE_LENGTH;
        if (l % BASE_LENGTH) len++;
        int index = 0;
        for (int i = l - 1; i >= 0; i -=
             BASE_LENGTH) {
            int tmp = 0;
            int k = i - BASE_LENGTH + 1;
            if (k < 0) k = 0;
            for (int j = k; j <= i; j++) {
                tmp = tmp*10 + num[j] - '0';
            }
            s[index++] = tmp;
        }
    }

    void clean() {
        while(len > 1 && !s[len-1]) len--;
    }

    string str() const {
        string ret = "";
        if (len == 1 && !s[0]) return "0";
        for(int i = 0; i < len; i++) {
            if (i == 0) {
                ret += int_to_string(s[len - i
                     - 1], BASE_LENGTH, false);
            } else {
                ret += int_to_string(s[len - i
                     - 1], BASE_LENGTH, true);
            }
        }
        return ret;
    }

    unsigned long long ll() const {
        unsigned long long ret = 0;
        for(int i = len-1; i >= 0; i--) {
            ret *= BASE;
            ret += s[i];
```

```cpp
        }
        return ret;
    }

    bigint operator + (const bigint& b) const {
        bigint c = b;
        while (c.len < len) c.s[c.len++] = 0;
        c.s[c.len++] = 0;
        bool r = 0;
        for (int i = 0; i < len || r; i++) {
            c.s[i] += (i<len)*s[i] + r;
            r = c.s[i] >= BASE;
            if (r) c.s[i] -= BASE;
        }
        c.clean();
        return c;
    }

    bigint operator - (const bigint& b) const {
        if (operator < (b)) throw "cannot do
            subtract";
        bigint c = *this;
        bool r = 0;
        for (int i = 0; i < b.len || r; i++) {
            c.s[i] -= b.s[i];
            r = c.s[i] < 0;
            if (r) c.s[i] += BASE;
        }
        c.clean();
        return c;
    }

    bigint operator * (const bigint& b) const {
        bigint c;
        c.len = len + b.len;
        for(int i = 0; i < len; i++)
            for(int j = 0; j < b.len; j++)
                c.s[i+j] += s[i] * b.s[j];
        for(int i = 0; i < c.len-1; i++){
            c.s[i+1] += c.s[i] / BASE;
            c.s[i] %= BASE;
        }
        c.clean();
        return c;
    }

    bigint operator / (const int b) const {
        bigint ret;
        int down = 0;
        for (int i = len - 1; i >= 0; i--) {
            ret.s[i] = (s[i] + down * BASE) / b;
            down = s[i] + down * BASE -
                ret.s[i] * b;
        }
        ret.len = len;
        ret.clean();
        return ret;
    }

    bool operator < (const bigint& b) const {
        if (len < b.len) return true;
        else if (len > b.len) return false;
        for (int i = 0; i < len; i++)
            if (s[i] < b.s[i]) return true;
            else if (s[i] > b.s[i]) return
                false;
        return false;
    }

    bool operator == (const bigint& b) const {
        return !(*this<b) && !(b<(*this));
    }

    bool operator > (const bigint& b) const {
        return b < *this;
    }
};


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

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



}
```

4 Matrix

4.1 matrix

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int MN = 205;
const int mod = 998244353;

struct matrix {
    int r, c;
    int m[MN][MN];
    matrix (int _r, int _c) : r (_r), c (_c) {
        memset(m, 0, sizeof m);
    }
    void print() {
        for (int i = 0; i < r; ++i) {
            for (int j = 0; j < c; ++j)
                cout << m[i][j] << " ";
            cout << endl;
        }
    }

    matrix operator *(const matrix &b){
        matrix res(r, b.c);
        if(c!=b.r) cout<< "bad matrix
            multiplication";
        for(int i=0;i<r;i++){
            for(int j=0;j<b.c;j++){
                for(int k=0;k<c;k++){
                    res.m[i][j]+=m[i][k]*b.m[k][j];
                    res.m[i][j]%=mod;
                }
            }
        }
        return res;
    }

    void operator *=(const matrix &b){
        *this = *this * b;
        //return *this;
    }
    matrix operator ^(int e){
        matrix res(r,r);
        //matrix id(r,r);
        matrix b = *this;
        for (int i = 0; i < r; ++i)
            res.m[i][i] = 1;
        if (e == 0) return res;
        while (true) {
            if (e & 1) res *= b;
            if ((e >>= 1) == 0) break;
            b *= b;
        }
        return res;
    }

    void operator ^=(int e){
        *this = *this ^ e;
        //return *this;
    }
};
```

5 Range Query

5.1 BIT

```
#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);
}
```

5.2 SEGTREEBigStepper

```
#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);
        // }
};
```

5.3 SEGTREELazy

```
#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);
            }
        }
};
```

5.4 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);
        }
    }
};
```

6 Syntax and Headers

6.1 CustomComparator

```
#include <bits/stdc++.h>
using namespace std;
struct cc{
    bool operator()(const int &a, const int
        &b) const{return b<a;}
};
set<int,cc> S;
```

7 Trees

7.1 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";
    }
}
```