// {{{ by HARRYPOTTER0

#include <bits/stdc++.h>

#include <unistd.h>

#define SZ(x) ((int)(x).size())

#define ALL(x) begin(x),end(x)

#define REP(i,n) for ( int i=0; i<int(n); i++ )

#define REP1(i,a,b) for ( int i=(a); i<=int(b); i++ )

#define FOR(it,c) for ( auto it=(c).begin(); it!=(c).end(); it++ )

#define MP make\_pair

#define PB push\_back

using namespace std;

typedef long long LL;

typedef pair<int,int> PII;

typedef vector<int> VI;

#ifdef HARRY

template<typename T>

void \_dump( const char\* s, T&& head ) { cerr<<s<<"="<<head<<endl; }

template<typename T, typename... Args>

void \_dump( const char\* s, T&& head, Args&&... tail ) {

int c=0;

while ( \*s!=',' || c!=0 ) {

if ( \*s=='(' || \*s=='[' || \*s=='{' ) c++;

if ( \*s==')' || \*s==']' || \*s=='}' ) c--;

cerr<<\*s++;

}

cerr<<"="<<head<<", ";

\_dump(s+1,tail...);

}

#define dump(...) do { \

fprintf(stderr, "%s:%d - ", \_\_PRETTY\_FUNCTION\_\_, \_\_LINE\_\_); \

\_dump(#\_\_VA\_ARGS\_\_, \_\_VA\_ARGS\_\_); \

} while (0)

template<typename Iter>

ostream& \_out( ostream &s, Iter b, Iter e ) {

s<<"[";

for ( auto it=b; it!=e; it++ ) s<<(it==b?"":" ")<<\*it;

s<<"]";

return s;

}

template<typename A, typename B>

ostream& operator <<( ostream &s, const pair<A,B> &p ) { return s<<"("<<p.first<<","<<p.second<<")"; }

template<typename T>

ostream& operator <<( ostream &s, const vector<T> &c ) { return \_out(s,ALL(c)); }

template<typename T, size\_t N>

ostream& operator <<( ostream &s, const array<T,N> &c ) { return \_out(s,ALL(c)); }

template<typename T>

ostream& operator <<( ostream &s, const set<T> &c ) { return \_out(s,ALL(c)); }

template<typename A, typename B>

ostream& operator <<( ostream &s, const map<A,B> &c ) { return \_out(s,ALL(c)); }

#else

#define dump(...)

#endif

template<typename T>

void \_R( T &x ) { cin>>x; }

void \_R( int &x ) { scanf("%d",&x); }

void \_R( long long &x ) { scanf("%" PRId64,&x); }

void \_R( double &x ) { scanf("%lf",&x); }

void \_R( char &x ) { scanf(" %c",&x); }

void \_R( char \*x ) { scanf("%s",x); }

void R() {}

template<typename T, typename... U>

void R( T& head, U&... tail ) {

\_R(head);

R(tail...);

}

template<typename T>

void \_W( const T &x ) { cout<<x; }

void \_W( const int &x ) { printf("%d",x); }

template<typename T>

void \_W( const vector<T> &x ) {

for ( auto i=x.cbegin(); i!=x.cend(); i++ ) {

if ( i!=x.cbegin() ) putchar(' ');

\_W(\*i);

}

}

void W() {}

template<typename T, typename... U>

void W( const T& head, const U&... tail ) {

\_W(head);

putchar(sizeof...(tail)?' ':'\n');

W(tail...);

}

#ifdef SHIK

#define FILEIO(...)

#else

#define FILEIO(name) do {\

freopen(name ".in","r",stdin); \

freopen(name ".out","w",stdout); \

} while (0)

#endif

// }}}

int main() {

int n,q;

R(n,q);

vector<LL> a(n);

REP(i,n) R(a[i]);

VI c(n+1);

REP(i,q) {

int l,r;

R(l,r);

c[l-1]++;

c[r]--;

}

REP(i,n) c[i+1]+=c[i];

c.pop\_back();

sort(ALL(a));

sort(ALL(c));

LL ans=0;

REP(i,n) ans+=c[i]\*a[i];

W(ans);

return 0;

}

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

#include <bits/stdc++.h>

#define FOR(i, n) for(lli i = 0; i < (lli)(n); ++i)

#define FORU(i, j, k) for(lli i = (j); i <= (lli)(k); ++i)

#define FORD(i, j, k) for(lli i = (j); i >= (lli)(k); --i)

#define X(A) get<0>(A)

#define Y(A) get<1>(A)

#define Z(A) get<2>(A)

#define W(A) get<3>(A)

#define mp make\_pair

#define mt make\_tuple

#define pb push\_back

//------------------------------------------------------------------------------

using namespace std;

using lli = long long int;

using llu = long long unsigned;

using pii = tuple<lli, lli>;

using ppii = tuple<pii, pii>;

using piii = tuple<lli, lli, lli>;

using vi = vector<lli>;

using vii = vector<pii>;

using viii = vector<piii>;

using vvi = vector<vi>;

using vvii = vector<vii>;

using vviii = vector<viii>;

using pt = complex<lli>;

//------------------------------------------------------------------------------

template<class A, class B>

ostream& operator<<(ostream& s, pair<A, B> const& a){

return s << "(" << X(a) << "," << Y(a) << ")";

}

template<class A, class B>

ostream& operator<<(ostream& s, tuple<A, B> const& a){

return s << "(" << X(a) << "," << Y(a) << ")";

}

template<class A, class B, class C>

ostream& operator<<(ostream& s, tuple<A, B, C> const& a){

return s << "(" << X(a) << "," << Y(a) << "," << Z(a) << ")";

}

template<class T>

ostream& operator<<(ostream& s, vector<T> const& a){

s << '[';

FOR(i, a.size()){

s << a[i];

if(i != (lli)a.size()-1) s << " ";

}

return s << ']';

}

//------------------------------------------------------------------------------

int main(int, char\*\*){

ios::sync\_with\_stdio(false);

lli n, a, b, c; cin >> n >> b >> a >> c;

lli r = 0;

if(b <= a-c){

r = n / b;

}else {

while(n >= a) {

lli delta = (n-c)/(a-c);

r += delta;

n -= delta\*(a-c);

}

r += n / b;

}

cout << r << endl;

return 0;

}

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

#ifndef \_\_clang\_\_

#pragma GCC optimize ("-O3")

#endif

#ifdef ONLINE\_JUDGE

#define NDEBUG 1

#endif

#define \_GLIBCXX\_USE\_CXX11\_ABI 0

#include <stdio.h>

#include <bits/stdc++.h>

#define DESTRUCT2(p, a, b) \

auto a = get<0>(p); \

auto b = get<1>(p);

#define DESTRUCT3(p, a, b, c) \

auto a = get<0>(p); \

auto b = get<1>(p); \

auto c = get<2>(p);

#define DESTRUCT4(p, a, b, c, d) \

auto a = get<0>(p); \

auto b = get<1>(p); \

auto c = get<2>(p); \

auto d = get<3>(p);

#define FOR(i, n) for(lli i = 0; i < (lli)(n); ++i)

#define FORU(i, j, k) for(lli i = (j); i <= (lli)(k); ++i)

#define FORD(i, j, k) for(lli i = (j); i >= (lli)(k); --i)

#define SQ(x) ((x)\*(x))

#define all(x) begin(x), end(x)

#define rall(x) rbegin(x), rend(x)

#define mp make\_pair

#define mt make\_tuple

#define pb push\_back

#define eb emplace\_back

using namespace std;

template<typename... As>

struct tpl : public std::tuple<As...> {

using std::tuple<As...>::tuple;

template<typename T = tuple<As...> >

typename tuple\_element<0, T>::type const&

x() const { return get<0>(\*this); }

template<typename T = tuple<As...> >

typename tuple\_element<0, T>::type&

x() { return get<0>(\*this); }

template<typename T = tuple<As...> >

typename tuple\_element<1, T>::type const&

y() const { return get<1>(\*this); }

template<typename T = tuple<As...> >

typename tuple\_element<1, T>::type&

y() { return get<1>(\*this); }

template<typename T = tuple<As...> >

typename tuple\_element<2, T>::type const&

z() const { return get<2>(\*this); }

template<typename T = tuple<As...> >

typename tuple\_element<2, T>::type&

z() { return get<2>(\*this); }

template<typename T = tuple<As...> >

typename tuple\_element<3, T>::type const&

w() const { return get<3>(\*this); }

template<typename T = tuple<As...> >

typename tuple\_element<3, T>::type&

w() { return get<3>(\*this); }

};

using lli = long long int;

using llu = long long unsigned;

using pii = tpl<lli, lli>;

using piii = tpl<lli, lli, lli>;

using piiii = tpl<lli, lli, lli, lli>;

using vi = vector<lli>;

using vii = vector<pii>;

using viii = vector<piii>;

using vvi = vector<vi>;

using vvii = vector<vii>;

using vviii = vector<viii>;

template<class T>

using min\_queue = priority\_queue<T, vector<T>, greater<T> >;

template<class T>

using max\_queue = priority\_queue<T>;

template<size\_t... I>

struct my\_index\_sequence {

using type = my\_index\_sequence;

static constexpr array<size\_t, sizeof...(I)> value = { {I...} };

};

namespace my\_index\_sequence\_detail {

template<typename I, typename J> struct concat;

template<size\_t... I, size\_t... J>

struct concat<my\_index\_sequence<I...>, my\_index\_sequence<J...> > :

my\_index\_sequence<I..., (sizeof...(I)+J)...> { };

template<size\_t N> struct make\_index\_sequence :

concat<typename make\_index\_sequence<N/2>::type, typename make\_index\_sequence<N-N/2>::type>::type { };

template <> struct make\_index\_sequence<0> : my\_index\_sequence<>{};

template <> struct make\_index\_sequence<1> : my\_index\_sequence<0>{};

}

template<class... A>

using my\_index\_sequence\_for = typename my\_index\_sequence\_detail::make\_index\_sequence<sizeof...(A)>::type;

template<class T, size\_t... I>

void print\_tuple(ostream& s, T const& a, my\_index\_sequence<I...>){

using swallow = int[];

(void)swallow{0, (void(s << (I == 0? "" : ", ") << get<I>(a)), 0)...};

}

template<class T>

ostream& print\_collection(ostream& s, T const& a){

s << '[';

for(auto it = begin(a); it != end(a); ++it){

s << \*it;

if(it != prev(end(a))) s << " ";

}

return s << ']';

}

template<class... A>

ostream& operator<<(ostream& s, tpl<A...> const& a){

s << '(';

print\_tuple(s, a, my\_index\_sequence\_for<A...>{});

return s << ')';

}

template<class... A>

ostream& operator<<(ostream& s, tuple<A...> const& a){

s << '(';

print\_tuple(s, a, my\_index\_sequence\_for<A...>{});

return s << ')';

}

template<class A, class B>

ostream& operator<<(ostream& s, pair<A, B> const& a){

return s << "(" << get<0>(a) << ", " << get<1>(a) << ")";

}

template<class T, size\_t I>

ostream& operator<<(ostream& s, array<T, I> const& a) { return print\_collection(s, a); }

template<class T>

ostream& operator<<(ostream& s, vector<T> const& a) { return print\_collection(s, a); }

template<class T, class U>

ostream& operator<<(ostream& s, multimap<T, U> const& a) { return print\_collection(s, a); }

template<class T>

ostream& operator<<(ostream& s, multiset<T> const& a) { return print\_collection(s, a); }

template<class T, class U>

ostream& operator<<(ostream& s, map<T, U> const& a) { return print\_collection(s, a); }

template<class T>

ostream& operator<<(ostream& s, set<T> const& a) { return print\_collection(s, a); }

namespace std {

namespace {

template <class T>

inline void hash\_combine(size\_t& seed, T const& v) {

seed ^= hash<T>()(v) + 0x9e3779b9 + (seed<<6) + (seed>>2);

}

template <class Tuple, size\_t Index = tuple\_size<Tuple>::value - 1>

struct HashValueImpl {

static void apply(size\_t& seed, Tuple const& tuple) {

HashValueImpl<Tuple, Index-1>::apply(seed, tuple);

hash\_combine(seed, get<Index>(tuple));

}

};

template <class Tuple>

struct HashValueImpl<Tuple, 0> {

static void apply(size\_t& seed, Tuple const& tuple) {

hash\_combine(seed, get<0>(tuple));

}

};

}

template <typename ... TT>

struct hash<tuple<TT...>> {

size\_t operator()(tuple<TT...> const& tt) const {

size\_t seed = 0;

HashValueImpl<tuple<TT...> >::apply(seed, tt);

return seed;

}

};

template <typename ... TT>

struct hash<tpl<TT...>> {

size\_t operator()(tpl<TT...> const& tt) const {

size\_t seed = 0;

HashValueImpl<tuple<TT...> >::apply(seed, tt);

return seed;

}

};

}

int read\_positive(){

char c; int x=0;

do { c = getchar(); } while(c<'0' || c>'9');

while(c>='0'&&c<='9') {

x=10\*x+(c-'0');

c = getchar();

}

return x;

}

//------------------------------------------------------------------------------

struct segment\_tree {

struct segment\_node {

lli value=-1; // maximum from point of view of the bottom

bool dirty=false;

lli base;

};

void pushall(lli i, lli a, lli b){

if(i<n){

push(i,a,b);

lli c = (a+b)/2;

pushall(2\*i,a,c);

pushall(2\*i+1,c+1,b);

}

}

void print(){

FOR(i,2\*n) cerr << A[i].value << " "; cerr << endl;

pushall(1,0,n-1);

FOR(i,2\*n) cerr << A[i].value << " "; cerr << endl;

FOR(i,2\*n) cerr << A[i].base << " "; cerr << endl;

FOR(i,2\*n) cerr << A[i].dirty << " "; cerr << endl;

}

lli n;

vector<segment\_node> A;

segment\_tree(){}

segment\_tree(lli n\_) : n((lli)1<<(lli)(log2(n\_)+1)), A(2\*n) {

}

void update(lli i, lli a, lli b) {

if(i < n) {

A[i].value=-1;

lli c = (a+b)/2;

A[i].value = max(A[2\*i].value + (b-c), A[2\*i+1].value);

}

}

void push(lli i, lli a, lli b){

if(i < n){

if(A[i].dirty){

lli c = (a+b)/2;

set\_\_(2\*i, a, c, A[i].base);

set\_\_(2\*i+1, c+1, b, A[i].base + (c-a+1));

A[i].dirty=0;

}

// add\_\_(2\*i , a, c, A[i].delta);

// add\_\_(2\*i+1, c+1, b, A[i].delta);

}

}

void set\_\_(lli i, lli a, lli b, lli v){

A[i].value=v+(b-a);

A[i].base=v;

A[i].dirty=1;

}

void set\_(lli i, lli a, lli b, lli l, lli r, lli v) {

if(l > b) return;

if(r < a) return;

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

set\_\_(i, a, b, v);

return;

}

push(i,a,b);

lli c = (a+b)/2;

set\_(2\*i , a , c, l, r, v);

set\_(2\*i+1, c+1, b, l, r, v + (c+1-a));

update(i,a,b);

}

void set(lli l, lli r, lli v){

set\_(1, 0, n-1, l, r, v - l);

}

#define C\_YELLOW "\033[33m"

#define C\_BYELLOW "\033[33;1m"

#define C\_RESET "\033[0m" // Reset the terminal state

// largest larger/or equal to v

pii query\_(lli i, lli a, lli b, lli l, lli r, lli v) {

//cerr << C\_YELLOW << i << " " << a << " " << b<< " " << l << " " << r << " " << v << " " << A[i].value << C\_RESET << endl;

if(l > b) return mt(-1,-1);

if(r < a) return mt(-1,-1);

push(i,a,b);

lli c = (a+b)/2;

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

if(A[i].value <= v) return mt(-1,-1);

if(i>=n) return mt(a,A[i].value);

pii r0 = query\_(2\*i+1, c+1, b, l, r, v);

if(r0==mt(-1,-1)) return query\_(2\*i, a, c, l, r, v+(b-c));

else return r0;

}else{

pii ans=max(query\_(2\*i+1, c+1, b, l, r, v),

query\_(2\*i, a, c, l, r, v+(b-c)));

return ans;

}

}

pii query(lli l, lli r, lli v){

return query\_(1, 0, n-1, l, r, v+r-(n-1));

}

};

// --- END SNIPPET SET SEGMENT TREE ---

// --- BEGIN SNIPPET TREE ---

#define WITH\_HLD 1

struct treechain {

lli root = -1;

lli size = 0;

vi nodes;

treechain() { }

#ifdef WITH\_HLD

using segment = segment\_tree; // segment tree / bit / other

segment A;

void build(){ A = segment(size); /\* make segment tree \*/ }

#else

void build(){ }

#endif

};

struct tree {

lli n;

vvi G;

vi parent;

vi depth;

vi size;

vi order, rorder;

vii eorder;

vector<treechain> chains;

vector<lli> chain;

vector<lli> chainIx;

tree(lli n\_) : n(n\_), G(n) { }

void read\_parent() {

FOR(i, n-1) {

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

G[1+i].pb(u); G[u].pb(1+i);

}

}

void calcparent() {

if(parent.empty()) {

parent.assign(n, -1);

order.clear();

vi E(n, 0);

vector<lli> Q; Q.pb(0);

while(!Q.empty()) {

lli i = Q.back(); Q.pop\_back();

order.pb(i);

E[i] = 1;

for(auto j : G[i]) if(!E[j]) {

parent[j] = i;

Q.pb(j);

}

}

rorder = order;

reverse(all(rorder));

}

}

template<typename T>

inline vector<vector<T> > makeEVec(T const& def = T()) {

vector<vector<T> > v(n); FOR(i, n) v[i].assign(G[i].size(), def);

return v;

}

void calceorder() {

if(eorder.empty()) {

vvi E = makeEVec(0ll);

FOR(i, n) FOR(j\_, G[i].size()) if(!E[i][j\_]) {

vii Q; Q.pb(mt(i,j\_));

vii tmp;

while(!Q.empty()) {

DESTRUCT2(Q.back(), i, j\_); Q.pop\_back();

tmp.pb(mt(i, j\_)); E[i][j\_] = 1;

lli j = G[i][j\_];

FOR(k\_, G[j].size()) if(G[j][k\_] != i && !E[j][k\_]) Q.pb(mt(j, k\_));

}

reverse(all(tmp));

eorder.insert(end(eorder), all(tmp));

}

}

}

void calcsize() {

calcparent();

if(size.empty()) {

size.assign(n, 1);

for(auto i : rorder) if(parent[i]!=-1) size[parent[i]] += size[i];

}

}

void calcdepth() {

calcparent();

if(depth.empty()) {

depth.assign(n, 0);

for(auto i : order) if(parent[i] != -1) depth[i] = depth[parent[i]] + 1;

}

}

void calchld() {

if(chains.empty()){

calcsize();

calcdepth();

chain.assign(n, -1);

for(auto i : rorder) {

lli best = -1;

for(auto j : G[i]) if(j != parent[i]) {

if(best == -1 || size[j] > size[best]) best = j;

}

if(best == -1) {

chain[i] = chains.size();

chains.eb();

}else{

chain[i] = chain[best];

}

chains[chain[i]].size += 1;

}

chainIx.assign(n, -1);

for(auto i : order) {

if(parent[i]==-1 || chain[i] != chain[parent[i]]) chains[chain[i]].root = i;

chainIx[i] = chains[chain[i]].nodes.size();

chains[chain[i]].nodes.pb(i);

};

for(auto& c : chains) c.build();

}

}

// O(log n) lca

lli lca(lli a, lli b) {

calchld();

while(chain[a] != chain[b]) {

if(depth[chains[chain[a]].root]>depth[chains[chain[b]].root]) {

a = parent[chains[chain[a]].root];

}else{

b = parent[chains[chain[b]].root];

}

}

if(depth[a]<depth[b]) return a; else return b;

}

lli dist(lli a, lli b) {

calchld();

return depth[a]+depth[b]-2\*depth[lca(a, b)];

}

#ifdef WITH\_HLD

// arguments to f :

// segment, segment range, range in path from a to b (a = 0, b = dist(a, b))

// the segment range is increasing

// the range in path can be decreasing

// O (log² n)

void hld\_query(lli a, lli b, function<void(treechain::segment&, lli, lli, lli, lli)> f) {

calchld();

lli from = 0, to = dist(a, b);

while(chain[a] != chain[b]) {

if(depth[chains[chain[a]].root]>depth[chains[chain[b]].root]) {

lli a\_ = parent[chains[chain[a]].root];

lli d = depth[a]-depth[a\_];

f(chains[chain[a]].A, 0, chainIx[a], from+d, from);

from += d;

a = a\_;

}else{

lli b\_ = parent[chains[chain[b]].root];

lli d = depth[b]-depth[b\_];

f(chains[chain[b]].A, 0, chainIx[b], to-d, to);

to -= d;

b = b\_;

}

}

if(depth[a]<depth[b]) {

f(chains[chain[a]].A, chainIx[a]+1, chainIx[b], from, to);

}else{

f(chains[chain[a]].A, chainIx[b]+1, chainIx[a], from, to);

}

}

#endif

};

// --- END SNIPPET TREE ---

#ifdef LOCAL

int main(int,char\*\*){

int n,m; scanf("%d%d", &n, &m);

tree tr(n+1); tr.read\_parent();

tr.calcparent();

tr.calcdepth();

tr.calchld();

vii M; M.reserve(m);

FOR(i, m) {

lli x, t; cin >> x >> t; M.pb(mt(x,t));

}

min\_queue<tpl<int, int, int, int, int, int> > Q;

FOR(i, m) {

auto p = M[i];

Q.push(mt(p.y(), 1, p.x(), p.x(), -1, i));

}

vi ANS(m);

vi W(1+n, -1);

vi C(1+n, -1);

while(!Q.empty()) {

auto p=Q.top(); Q.pop();

int t=get<0>(p);

int init=get<2>(p);

int cur=get<3>(p);

int ch=get<4>(p);

int ix=get<5>(p);

if(ix == W[cur]) {

if(cur==init) {

ANS[ix]=t;

W[cur]=-1; C[cur]=-1;

}else{

W[cur]=-1;

Q.push(mt(t+1,0,init,C[cur],cur,ix));

C[cur]=-1;

}

}else if(cur==0||W[cur]!=-1) {

if(ch!=-1)

Q.push(mt(t+1,0,init,ch,cur,ix));

else {

ANS[ix]=t;

}

}else{

W[cur]=ix; C[cur]=ch; Q.push(mt(t+1,1,init,tr.parent[cur],cur,ix));

}

}

FOR(i, m){

printf("%lld ", ANS[i]);

}

printf("\n");

return 0;

}

#else

int main(int, char\*\*){

int n,m; scanf("%d%d", &n, &m);

tree tr(n+1); tr.read\_parent();

tr.calcparent();

tr.calcdepth();

tr.calchld();

vii M; M.reserve(m);

FOR(i, m) {

lli x, t; cin >> x >> t; M.pb(mt(x,t));

}

auto sorter = [&](pii const& a){

return mt(a.y()+tr.depth[a.x()], a.x());

};

vi I(m); iota(all(I), 0);

vi ANS(m);

sort(all(I), [&](lli i, lli j){return sorter(M[i])<sorter(M[j]); });

//

for(lli i : I) {

lli x = M[i].x(), t = M[i].y();

// cerr << i << " " << M[i] << endl;

// find blocking !

lli r0=-1; lli t1;

{ lli x0=x;

lli t0=t;

while(x0!=-1){

pii r = tr.chains[tr.chain[x0]].A.query(0, tr.chainIx[x0], t0);

if(r.x()!=-1){

//cerr << r << endl;

r0=tr.chains[tr.chain[x0]].nodes[r.x()];

break;

}

lli nx0 = tr.parent[tr.chains[tr.chain[x0]].root];

t0 += tr.depth[x0] - tr.depth[nx0];

x0=nx0;

}

}

if(r0==-1) {

r0=0;

}

t1=t+tr.depth[x]-tr.depth[r0];

//cerr << i << " " << r0 << " " << t << " " << t1 << endl;

ANS[i] = t1+(t1-t);

lli aD = tr.depth[r0];

lli tgtC = tr.chain[r0];

while(1) {

auto& ch = tr.chains[tr.chain[x]];

lli ix0=0; if(tr.chain[x] == tgtC) { ix0 = tr.chainIx[r0]+1; }

if(ix0<=tr.chainIx[x]) {

ch.A.set(ix0, tr.chainIx[x], t1+tr.depth[ch.nodes[ix0]]-aD); // TODO

//cerr << "upd " << ix0 << " " << tr.chainIx[x] << " " << t1+tr.depth[ch.nodes[ix0]]-aD << endl;

}

// ch.A.print();

if(tr.chain[x] == tgtC) break;

x = tr.parent[tr.chains[tr.chain[x]].root];

}

}

//

FOR(i, m){

printf("%lld ", ANS[i]);

}

printf("\n");

return 0;

}

#endif

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

// eddy1021

#include <bits/stdc++.h>

using namespace std;

typedef double D;

typedef long double LD;

typedef long long ll;

typedef long long LL;

typedef pair<int,int> PII;

typedef pair<LL,LL> PLL;

typedef pair<LD,LD> Pt;

typedef tuple<int,int,int> tiii;

typedef tuple<LL,LL,LL> tlll;

#define mod9 1000000009ll

#define mod7 1000000007ll

#define INF 1023456789ll

#define INF16 10000000000000000ll

#define FI first

#define SE second

#define X FI

#define Y SE

#define PB push\_back

#define MP make\_pair

#define MT make\_tuple

#define eps 1e-9

#define SZ(x) (int)(x).size()

#define ALL(x) (x).begin(), (x).end()

#ifndef ONLINE\_JUDGE

#define debug(...) printf(\_\_VA\_ARGS\_\_)

#else

#define debug(...)

#endif

inline LL getint(){

LL \_x=0,\_tmp=1; char \_tc=getchar();

while( (\_tc<'0'||\_tc>'9')&&\_tc!='-' ) \_tc=getchar();

if( \_tc == '-' ) \_tc=getchar() , \_tmp = -1;

while(\_tc>='0'&&\_tc<='9') \_x\*=10,\_x+=(\_tc-'0'),\_tc=getchar();

return \_x\*\_tmp;

}

inline LL add( LL \_x , LL \_y , LL \_mod = mod7 ){

\_x += \_y;

return \_x >= \_mod ? \_x - \_mod : \_x;

}

inline LL sub( LL \_x , LL \_y , LL \_mod = mod7 ){

\_x -= \_y;

return \_x < 0 ? \_x + \_mod : \_x;

}

inline LL mul( LL \_x , LL \_y , LL \_mod = mod7 ){

\_x \*= \_y;

return \_x >= \_mod ? \_x % \_mod : \_x;

}

LL mypow( LL \_a , LL \_x , LL \_mod ){

if( \_x == 0 ) return 1LL;

LL \_ret = mypow( mul( \_a , \_a , \_mod ) , \_x >> 1 , \_mod );

if( \_x & 1 ) \_ret = mul( \_ret , \_a , \_mod );

return \_ret;

}

LL mymul( LL \_a , LL \_x , LL \_mod ){

if( \_x == 0 ) return 0LL;

LL \_ret = mymul( add( \_a , \_a , \_mod ) , \_x >> 1 , \_mod );

if( \_x & 1 ) \_ret = add( \_ret , \_a , \_mod );

return \_ret;

}

inline bool equal( D \_x , D \_y ){

return \_x > \_y - eps && \_x < \_y + eps;

}

int \_\_ = 1 , \_cs;

/\*\*\*\*\*\*\*\*\*default\*\*\*\*\*\*\*\*\*/

#define N 202020

void build(){

}

LL c , n , a[ N ] , b[ N ] , s[ N ];

void init(){

c = getint();

n = getint();

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

a[ i ] = getint();

sort( a + 1 , a + n + 1 );

}

inline bool okay( LL x ){

LL bl = 1 , br = n , ba = 0;

while( bl <= br ){

LL mid = ( bl + br ) >> 1;

if( a[ mid ] < x )

ba = mid, bl = mid + 1;

else

br = mid - 1;

}

if( ba == 0 ) return false;

LL res = b[ ba ] , bb;

if( res >= x ) res -= x;

while( ba > 0 && res > 0 ){

bl = 1; br = ba; ba = 0;

while( bl <= br ){

LL mid = ( bl + br ) >> 1;

if( a[ mid ] <= res )

ba = mid, bl = mid + 1;

else

br = mid - 1;

}

if( ba == 0 ) break;

bl = 1; br = ba; bb = ba;

while( bl <= br ){

LL mid = ( bl + br ) >> 1;

if( s[ ba ] - s[ mid - 1 ] <= res )

bb = mid , br = mid - 1;

else

bl = mid + 1;

}

res -= s[ ba ] - s[ bb - 1 ];

ba = bb - 1;

}

return res > 0;

}

void solve(){

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

s[ i ] = s[ i - 1 ] + a[ i ];

LL ret = c;

for( int i = n ; i >= 1 ; i -- ){

b[ i ] = ret;

if( ret >= a[ i ] )

ret -= a[ i ];

}

for( int i = 1 ; i <= 200000 ; i ++ )

if( okay( i ) ){

printf( "%d\n" , i );

exit( 0 );

}

puts( "Greed is good" );

}

int main(){

build();

//\_\_ = getint();

while( \_\_ -- ){

init();

solve();

}

}

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

#include <bits/stdc++.h>

using namespace std;

#define PB push\_back

#define MP make\_pair

#define LL long long

#define int LL

#define FOR(i,a,b) for(int i = (a); i <= (b); i++)

#define RE(i,n) FOR(i,1,n)

#define REP(i,n) FOR(i,0,(int)(n)-1)

#define R(i,n) REP(i,n)

#define VI vector<int>

#define PII pair<int,int>

#define LD long double

#define FI first

#define SE second

#define st FI

#define nd SE

#define ALL(x) (x).begin(), (x).end()

#define SZ(x) ((int)(x).size())

template<class C> void mini(C&a4, C b4) { a4 = min(a4, b4); }

template<class C> void maxi(C&a4, C b4) { a4 = max(a4, b4); }

template<class TH> void \_dbg(const char \*sdbg, TH h){cerr<<sdbg<<"="<<h<<endl;}

template<class TH, class...TA> void \_dbg(const char \*sdbg, TH h, TA... a) {

while (\*sdbg!=',')cerr<<\*sdbg++;cerr<<'='<<h<<",";\_dbg(sdbg+1, a...);

}

template<class T> ostream& operator<<(ostream &os, vector<T> V) {

os<<"[";for(auto vv:V)os<<vv<<","; os<<"]";

return os;

}

template<class L, class R> ostream& operator<<(ostream &os, pair<L, R> P) {

os<<"("<<P.first<<","<<P.second<<")";

return os;

}

#ifdef LOCAL

#define debug(...) \_dbg(#\_\_VA\_ARGS\_\_, \_\_VA\_ARGS\_\_)

#else

#define debug(...) (\_\_VA\_ARGS\_\_)

#define cerr if(0)cout

#endif

const int MaxPrice = 2e5 + 5,

MaxN = 2e5 + 5;

int price, N;

int cntCoins[MaxN];

int whoNow[MaxPrice];

int prevNow[MaxPrice];

bool isCorrect[MaxPrice];

vector<pair<int, int>> coinsElems;

bool simulate(int coin) {

int toGet = price;

int ptrCoin = 0;

debug(coin);

bool used = false;

int who = price + 1;

while (toGet > 0) {

debug(prevNow[who], whoNow[toGet]);

who = min(prevNow[who], whoNow[toGet]);

if (who == -1) { return true; }

debug(toGet, who);

if (who <= coin && !used) {

debug("!");

if (toGet < coin) { return false; }

toGet -= coin;

used = true;

}

int numFit = min(cntCoins[who], toGet / who);

toGet -= who \* numFit;

debug(numFit, toGet);

}

return toGet > 0;

}

int32\_t main() {

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

cin.tie(0);

cout << fixed << setprecision(12);

cerr << fixed << setprecision(6);

cin >> price >> N;

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

int v;

cin >> v;

cntCoins[v]++;

}

whoNow[0] = prevNow[0] = -1;

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

if (cntCoins[i]) {

whoNow[i] = i;

prevNow[i] = whoNow[i - 1];

} else {

whoNow[i] = whoNow[i - 1];

prevNow[i] = prevNow[i - 1];

}

}

prevNow[price + 1] = whoNow[price];

for (int add = 1; add <= price; add++) {

if (simulate(add)) {

cout << add << "\n";

return 0;

}

}

cout << "Greed is good\n";

}

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

/\*\*

\* code generated by JHelper

\* More info: https://github.com/AlexeyDmitriev/JHelper

\* @author RiaD

\*/

#include <iostream>

#include <fstream>

#include <iostream>

#include <iterator>

#include <string>

#include <stdexcept>

#ifndef SPCPPL\_ASSERT

#ifdef SPCPPL\_DEBUG

#define SPCPPL\_ASSERT(condition) \

if(!(condition)) { \

throw std::runtime\_error(std::string() + #condition + " in line " + std::to\_string(\_\_LINE\_\_) + " in " + \_\_PRETTY\_FUNCTION\_\_); \

}

#else

#define SPCPPL\_ASSERT(condition)

#endif

#endif

/\*\*

\* Support decrementing and multi-passing, but not declared bidirectional(or even forward) because

\* it's reference type is not a reference.

\*

\* It doesn't return reference because

\* 1. Anyway it'll not satisfy requirement [forward.iterators]/6

\* If a and b are both dereferenceable, then a == b if and only if \*a and

\* b are bound to the same object.

\* 2. It'll not work with reverse\_iterator that returns operator \* of temporary which is temporary for this iterator

\*

\* Note, reverse\_iterator is not guaranteed to work now too since it works only with bidirectional iterators,

\* but it's seems to work at least on my implementation.

\*

\* It's not really useful anywhere except iterating anyway.

\*/

template <typename T>

class IntegerIterator: public std::iterator<std::input\_iterator\_tag, T, std::ptrdiff\_t, T\*, T> {

public:

explicit IntegerIterator(T value): value(value) {

}

IntegerIterator& operator++() {

++value;

return \*this;

}

IntegerIterator operator++(int) {

IntegerIterator copy = \*this;

++value;

return copy;

}

IntegerIterator& operator--() {

--value;

return \*this;

}

IntegerIterator operator--(int) {

IntegerIterator copy = \*this;

--value;

return copy;

}

T operator\*() const {

return value;

}

bool operator==(IntegerIterator rhs) const {

return value == rhs.value;

}

bool operator!=(IntegerIterator rhs) const {

return !(\*this == rhs);

}

private:

T value;

};

template <typename T>

class IntegerRange {

public:

IntegerRange(T begin, T end): begin\_(begin), end\_(end) {

SPCPPL\_ASSERT(begin <= end);

}

IntegerIterator<T> begin() const {

return IntegerIterator<T>(begin\_);

}

IntegerIterator<T> end() const {

return IntegerIterator<T>(end\_);

}

private:

T begin\_;

T end\_;

};

template <typename T>

class ReversedIntegerRange {

typedef std::reverse\_iterator<IntegerIterator<T>> IteratorType;

public:

ReversedIntegerRange(T begin, T end): begin\_(begin), end\_(end) {

SPCPPL\_ASSERT(begin >= end);

}

IteratorType begin() const {

return IteratorType(IntegerIterator<T>(begin\_));

}

IteratorType end() const {

return IteratorType(IntegerIterator<T>(end\_));

}

private:

T begin\_;

T end\_;

};

template <typename T>

IntegerRange<T> range(T to) {

return IntegerRange<T>(0, to);

}

template <typename T>

IntegerRange<T> range(T from, T to) {

return IntegerRange<T>(from, to);

}

template <typename T>

IntegerRange<T> inclusiveRange(T to) {

return IntegerRange<T>(0, to + 1);

}

template <typename T>

IntegerRange<T> inclusiveRange(T from, T to) {

return IntegerRange<T>(from, to + 1);

}

template <typename T>

ReversedIntegerRange<T> downrange(T from) {

return ReversedIntegerRange<T>(from, 0);

}

template <typename T>

ReversedIntegerRange<T> downrange(T from, T to) {

return ReversedIntegerRange<T>(from, to);

}

template <typename T>

ReversedIntegerRange<T> inclusiveDownrange(T from) {

return ReversedIntegerRange<T>(from + 1, 0);

}

template <typename T>

ReversedIntegerRange<T> inclusiveDownrange(T from, T to) {

return ReversedIntegerRange<T>(from + 1, to);

}

#include <algorithm>

#include <vector>

template <typename R>

void sort(R& range) {

std::sort(range.begin(), range.end());

}

template <typename R, typename Comp>

void sort(R& range, Comp comp) {

std::sort(range.begin(), range.end(), comp);

}

template <typename R>

void reverse(R& range) {

std::reverse(range.begin(), range.end());

}

template <typename R, typename T>

auto lower\_bound(const R& range, const T& value) -> decltype(range.begin()) {

return std::lower\_bound(range.begin(), range.end(), value);

}

template <typename R, typename T, typename Comp>

auto lower\_bound(const R& range, const T& value, Comp comp) -> decltype(range.begin()) {

return std::lower\_bound(range.begin(), range.end(), value, comp);

}

template <typename R, typename T>

auto upper\_bound(const R& range, const T& value) -> decltype(range.begin()) {

return std::upper\_bound(range.begin(), range.end(), value);

}

template <typename R, typename T, typename Comp>

auto upper\_bound(const R& range, const T& value, Comp comp) -> decltype(range.begin()) {

return std::upper\_bound(range.begin(), range.end(), value, comp);

}

template <typename R>

auto min\_element(const R& range) -> decltype(range.begin()) {

return std::min\_element(range.begin(), range.end());

}

template <typename R>

auto max\_element(const R& range) -> decltype(range.begin()) {

return std::max\_element(range.begin(), range.end());

}

template <typename R>

bool next\_permutation(R& range) {

return std::next\_permutation(range.begin(), range.end());

}

template <typename T>

void unique(std::vector<T>& range) {

range.erase(std::unique(range.begin(), range.end()), range.end());

}

template <typename R>

R sorted(R range) {

sort(range);

return range;

}

template <typename R, typename Comp>

R sorted(R range, Comp comp) {

sort(range, comp);

return range;

}

template <typename R>

R reversed(R range) {

reverse(range);

return range;

}

using namespace std;

class TaskF {

public:

void solve(std::istream& in, std::ostream& out) {

int n;

in >> n;

vector<pair<int, int>> p;

p.reserve(2 \* n);

int64\_t ans = 0;

for (int i: range(n)) {

int a, b, c, d;

in >> a >> b >> c >> d;

if (a <= d && b <= c) {

continue;

}

if (a + b > c + d) {

p.emplace\_back(a, b);

p.emplace\_back(c, d);

} else {

if (a > d) {

ans += a - d;

} else {

ans += c - b;

}

}

}

sort(p, [](auto a, auto b) {

return a.first + a.second > b.first + b.second;

});

for (int i: range(p.size())) {

//cerr << p[i].first << ' ' << p[i].second << endl;

if (i % 2 == 0) {

//cerr << "+" << p[i].first << endl;

ans += p[i].first;

} else {

//cerr << "-" << p[i].second << endl;

ans -= p[i].second;

}

}

out << ans << "\n";

}

};

int main() {

std::ios\_base::sync\_with\_stdio(false);

TaskF solver;

std::istream& in(std::cin);

std::ostream& out(std::cout);

solver.solve(in, out);

in.tie(0);

out << std::fixed;

out.precision(20);

return 0;

}

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

#include <bits/stdc++.h>

using namespace std;

typedef signed long long ll;

#undef \_P

#define \_P(...) (void)printf(\_\_VA\_ARGS\_\_)

#define FOR(x,to) for(x=0;x<(to);x++)

#define FORR(x,arr) for(auto& x:arr)

#define ITR(x,c) for(\_\_typeof(c.begin()) x=c.begin();x!=c.end();x++)

#define ALL(a) (a.begin()),(a.end())

#define ZERO(a) memset(a,0,sizeof(a))

#define MINUS(a) memset(a,0xff,sizeof(a))

//-------------------------------------------------------

int N;

int A1,B1,A2,B2;

ll ret;

void solve() {

int i,j,k,l,r,x,y; string s;

cin>>N;

priority\_queue<int> Q;

FOR(i,N) {

cin>>A1>>B1>>A2>>B2;

if(A1<=B2 && B1<=A2) continue;

if(A1+B1<=A2+B2) {

if(A1-B2>=0) ret +=A1-B2;

else ret +=A2-B1;

}

else {

ret += A1+A2;

Q.push(A1+B1);

Q.push(A2+B2);

}

}

for(i=0;Q.size();i^=1,Q.pop()) ret -= i\*Q.top();

cout<<ret<<endl;

}

int main(int argc,char\*\* argv){

string s;int i;

if(argc==1) ios::sync\_with\_stdio(false), cin.tie(0);

FOR(i,argc-1) s+=argv[i+1],s+='\n';

FOR(i,s.size()) ungetc(s[s.size()-1-i],stdin);

solve(); return 0;

}

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

#include <bits/stdc++.h>

#define SZ(X) ((int)(X).size())

#define ALL(X) (X).begin(), (X).end()

#define REP(I, N) for (int I = 0; I < (N); ++I)

#define REPP(I, A, B) for (int I = (A); I < (B); ++I)

#define RI(X) scanf("%d", &(X))

#define RII(X, Y) scanf("%d%d", &(X), &(Y))

#define RIII(X, Y, Z) scanf("%d%d%d", &(X), &(Y), &(Z))

#define DRI(X) int (X); scanf("%d", &X)

#define DRII(X, Y) int X, Y; scanf("%d%d", &X, &Y)

#define DRIII(X, Y, Z) int X, Y, Z; scanf("%d%d%d", &X, &Y, &Z)

#define RS(X) scanf("%s", (X))

#define CASET int \_\_\_T, case\_n = 1; scanf("%d ", &\_\_\_T); while (\_\_\_T-- > 0)

#define MP make\_pair

#define PB push\_back

#define MS0(X) memset((X), 0, sizeof((X)))

#define MS1(X) memset((X), -1, sizeof((X)))

#define LEN(X) strlen(X)

#define PII pair<int,int>

#define VI vector<int>

#define VPII vector<pair<int,int> >

#define PLL pair<long long,long long>

#define VPLL vector<pair<long long,long long> >

#define F first

#define S second

typedef long long LL;

using namespace std;

const int MOD = 1e9+7;

const int SIZE = 262144;

int c,a[SIZE];

LL BIT[262144];

void ins(int x,int v){

for(;x<SIZE;x+=x&-x)BIT[x]+=v;

}

int qq(LL low\_bd){

LL me=0;

int now=0;

for(int it=17;it>=0;it--){

if(me+BIT[now|(1<<it)]<low\_bd){

now|=1<<it;

me+=BIT[now];

}

}

return now+1;

}

LL qq2(int x){

LL res=0;

for(;x;x-=x&-x)res+=BIT[x];

return res;

}

bool solve(int n){

int r=c;

while(r){

int it=upper\_bound(a+1,a+1+n,r)-1-a;

LL all=qq2(it);

if(!it||all<r)return 0;

if(all==r||a[it]==r)return 1;

int ll=qq(all-r);

r-=all-qq2(ll);

n=ll;

}

return 1;

}

int main(){

RI(c);

DRI(n);

REPP(i,1,n+1)RI(a[i]);

sort(a+1,a+1+n);

REPP(i,1,n+1)ins(i,a[i]);

if(!solve(n))return 0\*puts("Greed is good");

n++;

for(int i=n;i>1;i--)a[i]=a[i-1];

a[1]=1;

MS0(BIT);

REPP(i,1,n+1)ins(i,a[i]);

int it=1;

REPP(i,1,c){

if(!solve(n))return 0\*printf("%d\n",i);

ins(it,1);

a[it]++;

while(it<n&&a[it]>a[it+1]){

ins(it,-1);

ins(it+1,1);

swap(a[it],a[it+1]);

it++;

}

}

return 0\*puts("Greed is good");

}

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

#include<bits/stdc++.h>

#define HEAP priority\_queue

#define rep(i,n) for(int i=0;i<(n);i++)

#define per(i,n) for(int i=(n)-1;i>=0;i--)

#define forn(i,l,r) for(int i=(l);i<=(r);i++)

#define nrof(i,r,l) for(int i=(r);i>=(l);i--)

#define mp make\_pair

#define mt make\_tuple

#define pb push\_back

#define X first

#define Y second

#define eps 1e-20

#define pi 3.1415926535897932384626433832795

#define orz int

#define yjz main

#define fizzydavid return

#define ak 0

#define la ;

#define debug puts("OK");

#define SZ(x) (int)x.size()

#define ALL(x) x.begin(),x.end()

using namespace std;

typedef long long LL;

typedef double flt;

typedef vector<int> vi;

typedef pair<int,int> pii;

const int iinf=1e9+7;

const LL linf=1ll<<60;

const flt dinf=1e10;

inline LL read()

{

bool f=0; LL x=0; char c=getchar();

while((c<'0' || c>'9') && c!='-') c=getchar();

if(c=='-') { f=1; c=getchar(); }

while(c>='0' && c<='9') { x=x\*10+c-'0'; c=getchar(); }

if(f) x=-x; return x;

}

inline void scf(int &x)

{

bool f=0; x=0; char c=getchar();

while((c<'0' || c>'9') && c!='-') c=getchar();

if(c=='-') { f=1; c=getchar(); }

while(c>='0' && c<='9') { x=x\*10+c-'0'; c=getchar(); }

if(f) x=-x; return;

}

void scf(int &x,int &y) { scf(x); return scf(y); }

void scf(int &x,int &y,int &z) { scf(x); scf(y); return scf(z); }

//---------------------------head----------------------------

const int N=4e5+100;

int n,m,top,clk;

vector<pii> g[N];

pii E[N];

int dfn[N],low[N];

int stk[N];

bool vis[N];

int mx,rt;

inline void dfs(int u=1,int par=0)

{

vis[u]=1;

stk[top]=u; top++;

clk++; dfn[u]=low[u]=clk;

for(auto i:g[u])

{

int v=i.X,id=i.Y;

if(v==par) continue;

if(!vis[v])

{

dfs(v,u);

E[id]=mp(v,u);

}

else E[id]=mp(u,v);

low[u]=min(low[u],low[v]);

}

if(low[u]==dfn[u])

{

int cnt=0;

while(top-1>=0 && stk[top-1]!=u) cnt++, top--;

cnt++; top--;

if(cnt>mx) mx=cnt, rt=u;

}

}

orz yjz()

{

scf(n,m);

rep(i,m)

{

int u,v;

scf(u,v);

E[i]=mp(u,v);

g[u].pb(mp(v,i));

g[v].pb(mp(u,i));

}

dfs();

printf("%d\n",mx);

memset(vis,0,sizeof(vis));

dfs(rt);

rep(i,m) printf("%d %d\n",E[i].X,E[i].Y);

fizzydavid ak la

}

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

#include<cstdio>

#include<cstdlib>

#include<algorithm>

#include<cstring>

#include<vector>

#include<queue>

#define mp make\_pair

#define pb push\_back

using namespace std;

const int maxn= 1e6;

typedef pair<int,int> PA;

struct edge{

int v,id;

};

vector<edge> g[maxn];

queue<int> Q;

PA ans[maxn];

PA da[maxn];

vector<int> bcc[maxn];

int h[maxn],nex[maxn],des[maxn],id[maxn];

int pre[maxn],low[maxn],cut[maxn];

int sta[maxn],bnum[maxn],num[maxn],vis[maxn];

int n,m,i,j,x,y,tot,tim,la,cnt,r;

void addedge(int x,int y,int i){

nex[++tot]= h[x];

h[x]= tot;

des[tot]= y;

id[tot]= i;

}

void dfs(int o,int fa){

pre[o]= low[o]= ++tim;

int p= h[o];

while (p){

int v= des[p];

if (!pre[v]){

dfs(v,p|1);

low[o]= min(low[o],low[v]);

if (low[v]>pre[o]){

cut[p]= cut[p^1]= 1;

sta[++la]= id[p];

}

}else if (pre[v]<pre[o] && (p|1)!=fa)

low[o]= min(low[o],pre[v]);

p= nex[p];

}

}

void dfs1(int o){

bnum[o]= cnt;

num[cnt]++;

int i;

for (i= h[o];i;i= nex[i]){

if (vis[id[i]]) continue;

int v= des[i];

vis[id[i]]= 1;

if (cut[i]) continue;

ans[id[i]]= mp(o,v);

if (!bnum[v])

dfs1(v);

}

}

void bfs(int s){

memset(vis,0,sizeof(vis));

Q.push(s);

vis[s]= 1;

while (!Q.empty()){

int u= Q.front(),len= g[u].size(),i;

int x,y;

Q.pop();

for (i= 0;i<len;i++){

int v= g[u][i].v,j= g[u][i].id;

x= da[j].first;

y= da[j].second;

if (bnum[x]==u)

ans[j]= mp(x,y);

else ans[j]= mp(y,x);

if (!vis[v]){

vis[v]= 1;

Q.push(v);

}

}

}

}

int main()

{

//freopen("1.in","r",stdin);

//freopen("1.out","w",stdout);

scanf("%d%d",&n, &m);

tot= 1;

for (i= 1;i<=m;i++){

scanf("%d%d",&x, &y);

addedge(x,y,i);

addedge(y,x,i);

da[i]= mp(x,y);

}

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

if (!pre[i])

dfs(i,-1);

r= 1;

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

if (!bnum[i]){

cnt++;

dfs1(i);

if (num[cnt]>num[r])

r= cnt;

}

for (i= 1;i<=la;i++){

x= da[sta[i]].first;

y= da[sta[i]].second;

g[bnum[x]].pb((edge){bnum[y],sta[i]});

g[bnum[y]].pb((edge){bnum[x],sta[i]});

}

bfs(r);

printf("%d\n",num[r]);

for (i= 1;i<=m;i++)

printf("%d %d\n",ans[i].first,ans[i].second);

return 0;

}

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx