/\*

\* @Author: Ykui.Chen

\* @Date: 2021-08-18 21:14:39

\* @Last Modified by: Ykui.Chen

\* @Last Modified time: 2021-08-18 21:44:37

\* 主席树（可持久化权值线段树） poj2104

\* 单点修改，区间查询第k大值

\*/

#include <iostream>

#include<stdio.h>

#include<algorithm>

#define ll long long

using namespace std;

const ll maxn = 1e5 + 7;

struct segmentTree {

ll l, r;//左右儿子

ll maxx;

} tree[maxn \* 40];

ll tot = 0;

ll root[maxn];

ll n, m, a[maxn];

ll b[maxn];

ll build(ll l, ll r) {//建立一颗空树

ll p = (++tot); //动态分配结点

if (l == r) {

tree[p].maxx = 0;

return p;

}

ll mid = (l + r) >> 1;

tree[p].l = build(l, mid);

tree[p].r = build(mid + 1, r);

tree[p].maxx = tree[tree[p].l].maxx + tree[tree[p].r].maxx;

return p; //返回当前结点

}

ll insert(ll rt, ll l, ll r, ll x, ll val) {//插入操作，新建一条链

ll p = (++tot); //相当于生成新结点

tree[p] = tree[rt]; //先指向当前根结点

if (l == r) { //到达叶节点更新

tree[p].maxx = val;

return p; //返回

}

//cout<<tree[tree[p].l].maxx<<" "<<tree[tree[p].r].maxx<<endl;

ll mid = (l + r) >> 1;

if (x <= mid) { //向左向右移动

tree[p].l = insert(tree[rt].l, l, mid, x, val);

} else {

tree[p].r = insert(tree[rt].r, mid + 1, r, x, val);

}

tree[p].maxx = tree[tree[p].l].maxx + tree[tree[p].r].maxx; //自底部向上更新权值

return p;

}

ll ask(ll lc,ll rc, ll l, ll r, ll k) {//查询操作

if(l==r)

{

return l;

}

ll mid=(l+r)>>1;

// cout<<tree[tree[rc].l].maxx<<" "<<tree[tree[lc].l].maxx<<endl;

if(tree[tree[rc].l].maxx-tree[tree[lc].l].maxx>=k)

{

return ask(tree[lc].l,tree[rc].l,l,mid,k);

}

else return ask(tree[lc].r,tree[rc].r,mid+1,r,k-(tree[tree[rc].l].maxx-tree[tree[lc].l].maxx));

}

int main() {

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

root[0] = build(1, n);

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

{

scanf("%lld",&a[i]);

b[i-1]=a[i];

}

sort(b,b+n);

ll N=unique(b,b+n)-b;

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

ll x=lower\_bound(b,b+N,a[i])-b;

root[i] = insert(root[i-1], 1, n, x+1, 1); //每插入一次产生一个新根结点

// cout<<tree[root[i]].maxx<<endl;

}

// cout<<tree[tree[14].l].maxx<<" "<<tree[30].maxx<<endl;

while(m--)

{

ll l,r,k;

scanf("%lld%lld%lld",&l,&r,&k);

ll ans=ask(root[l-1],root[r],1,n,k);

printf("%lld\n",b[ans-1]);

}

return 0;

}

/\*

\* @Author: Ykui.Chen

\* @Date: 2021-08-02 21:40:00

\* @Last Modified by: Ykui.Chen

\* @Last Modified time: 2021-08-02 21:58:54

\*

\* ABOUT: KMP

\*/

#include<bits/stdc++.h>

#define ll long long

using namespace std;

const ll maxn=1e5+7;

ll nextval[maxn];

string s,t;

void getnext()

{

ll j=0;

ll k=-1;

nextval[0]=-1;

while(j<t.size())

{

if(k==-1||t[j]==t[k])

{

j++;

k++;

if(t[j]==t[k]) //防止移动后与原先相同

{

nextval[j]=nextval[k];

}

else nextval[j]=k;

}

else k=nextval[k];

}

}

bool KMP()

{

ll i=0;

ll j=0;

getnext();

while(i<s.size()&&j<t.size())

{

if(j==-1||s[i]==t[j])

{

i++;

j++;

}

else j=nextval[j];

}

if(j>=t.size())

{

return true;

}

else return false;

}

int main()

{

return 0;

}

#include <bits/stdc++.h>

#define ll long long

using namespace std;

const ll maxn = 1000005;

typedef struct node {

ll val;

node \*l, \*r;

node(ll x)

{

val=x;

l=r=nullptr;

}

} node;

ll pre[maxn];

ll in[maxn];

ll n, m;

ll l[maxn];

ll r[maxn];

ll s[maxn];

ll vis[maxn];

ll z[maxn];

ll mp[maxn];

ll find(ll x) {

if (x != s[x]) {

return s[x] = find(s[x]);

} else

return s[x];

}

void init()

{

for(ll i=0;i<=1000005;i++){

vis[i]=0;

s[i]=i;

z[i]=-1;

}

}

void lca(node\* p,node\* pre) {

if (p->l) { //dfs

lca(p->l,p);

ll x=find(p->val);

ll y=find(p->l->val);

if(x!=y)

{

s[y]=x;

}

}

if (p->r) {

lca(p->r,p);

ll x=find(p->val);

ll y=find(p->r->val);

if(x!=y)

{

s[y]=x;

}

}

vis[p->val]=1; //标记该点以及该点的子树已经被遍历完成

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

{

if(vis[l[i]]&&vis[r[i]]) //如果两个点都被标记完成

{

if(l[i]==p->val) //且其中有一个点是当前点

{

z[i]=find(r[i]);

}

else if(r[i]==(p->val))

{

z[i]=find(l[i]);

}

else continue;

}

}

}

node\* insert(node\* p, ll x) {

if (p == nullptr) {

return new node(x);

}

if (x < p->val) {

p->l = insert(p->l, x);

} else

p->r = insert(p->r, x);

return p;

}

int main() {

cin >> m >> n;

node\* root = nullptr;

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

cin >> in[i];

root=insert(root,in[i]);

mp[in[i]]=1;

}

ll s = 1;

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

cin >> l[i] >> r[i];

}

node \*pre=new node(1000005);

init();

// pre->val=1000005;

lca(root,pre);

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

{

if(z[i]==-1)

{

if(!mp[l[i]]&&!mp[r[i]])

{

cout<<"ERROR: "<<l[i]<<" and "<<r[i]<<" are not found."<<endl;

}

else if(!mp[l[i]])

{

cout<<"ERROR: "<<l[i]<<" is not found."<<endl;

}

else

{

cout<<"ERROR: "<<r[i]<<" is not found."<<endl;

}

}

else

{

if(z[i]==l[i])

{

cout<<l[i]<<" is an ancestor of "<<r[i]<<"."<<endl;

}

else if(z[i]==r[i])

{

cout<<r[i]<<" is an ancestor of "<<l[i]<<"."<<endl;

}

else

{

cout<<"LCA of "<<l[i]<<" and "<<r[i]<<" is "<<z[i]<<"."<<endl;

}

}

}

return 0;

}

#include<bits/stdc++.h>

#define ll long long

using namespace std;

const ll maxn=1e6+7;

vector<ll>e[maxn];

ll in[maxn];//时间戳——进

ll out[maxn];//时间戳——出

ll num[maxn];

ll fa[maxn][30];//代表第i个结点的2^(j-1)个父亲

ll t[maxn]; //温度

ll dep[maxn];//树的深度

ll lg[maxn];

ll tim=0;

ll n;

typedef struct node//保存询问结点

{

ll i,l,x;

bool operator < (const node & y)const

{

return l<y.l;

}

}node;

struct tree //线段树

{

ll l,r,sum;

}tree[maxn\*4];

void build(ll rt,ll l,ll r)

{

tree[rt].l=l;

tree[rt].r=r;

tree[rt].sum=r-l+1;

if(l!=r)

{

ll mid=(l+r)>>1;

build(rt<<1,l,mid);

build(rt<<1|1,mid+1,r);

}

}

void change(ll rt,ll x)

{

tree[rt].sum--;

if(tree[rt].l!=tree[rt].r)

{

ll mid=(tree[rt].l+tree[rt].r)>>1;

if(x<=mid)

{

change(rt<<1,x);

}

else change(rt<<1|1,x);

}

}

ll ask(ll rt,ll l,ll r)

{

if(tree[rt].l>=l&&tree[rt].r<=r)

{

return tree[rt].sum;

}

ll mid=(tree[rt].l+tree[rt].r)>>1;

ll ans=0;

if(l<=mid)

{

ans+=ask(rt<<1,l,r);

}

if(r>mid) ans+=ask(rt<<1|1,l,r);

return ans;

}

//------------------end

vector<node>ak;

ll ans[maxn];

void dfs1(ll x,ll f)//求dfs序列，将树节点转换为线性区间

{

in[x]=(++tim); //进

num[tim]=x;

for(ll i=0;i<e[x].size();i++)

{

ll v=e[x][i];

if(v!=f)

{

dfs1(v,x);

}

}

out[x]=tim; // 出

}

void dfs2(ll x)//书上倍增求父亲

{

for(ll i=1;i<=lg[dep[x]];i++)

{

if(fa[x][i-1])

{

fa[x][i]=fa[fa[x][i-1]][i-1];

}

else break;

}

for(ll i=0;i<e[x].size();i++)

{

ll v=e[x][i];

if(v!=fa[x][0])

{

fa[v][0]=x;

dep[v]=dep[x]+1;

dfs2(v);

}

}

}

ll lca(ll x,ll r)// 最近公共祖先求最高的小于r的结点

{

ll y=x;

for(ll i=lg[dep[x]];i>=0;i--)

{

if(t[fa[y][i]]<=r)

{

y=fa[y][i];

}

}

return y;

}

bool cmp(ll a,ll b)

{

return t[a]<t[b];

}

int main()

{

deque<ll>Q;

scanf("%lld",&n);

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

lg[i] = lg[i - 1] + (1<<lg[i - 1] == i);

t[0]=0x3f3f3f3f;

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

{

ll u,v;

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

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

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

}

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

{

scanf("%lld",&t[i]);

Q.push\_back(i);

}

sort(Q.begin(),Q.end(),cmp);

dfs1(1,0);

dfs2(1);

ll q;

scanf("%lld",&q);

for(ll i=1;i<=q;i++){

node tmp;

tmp.i=i;

ll x,l,r;

scanf("%lld%lld%lld",&x,&l,&r);

if(t[x]<l||t[x]>r)

{

tmp.l=-1;

}

else

{

ll y=lca(x,r);//求出最高的结点

tmp.x=y;

tmp.l=l;

//利用树状数组维护dfs序

}

ak.push\_back(tmp);

}

build(1,1,n);

sort(ak.begin(),ak.end());

for(ll i=0;i<ak.size();i++)

{

node tmp=ak[i];

if(tmp.l==-1)

{

ans[ak[i].i]=0;

}

else

{

while(!Q.empty()&&t[Q[0]]<ak[i].l)

{

change(1,in[Q[0]]);

Q.pop\_front();

}

// cout<<in[ak[i].x]<<" "<<out[ak[i].x]<<endl;

ans[ak[i].i]=ask(1,in[ak[i].x],out[ak[i].x]);

}

}

for(ll i=1;i<=q;i++)

{

cout<<ans[i]<<endl;

}

return 0;

}

/\*

11

1 9

1 10

1 5

1 4

9 2

9 7

9 6

4 8

3 7

7 11

399 289 285 390 396 287 288 286 398 397 280

\*/