

Codeforces Round #700 (Div. 1)

[比赛链接](#)

A. Searching Local Minimum

题意

给定一个 $1 \sim n$ 的排列，最多允许 100 次询问，每次可以询问指定位置的值。

要求找到一个 i 满足 $a_i < a_{i-1}$ 且 $a_i < a_{i+1}$ 假定 $a_0 = a_{n+1} = \infty$

题解

维护区间 $[l, r]$ 满足 $a_l < a_{l-1}, a_r < a_{r+1}$ 于是当 $l=r$ 时答案位置确定。

接下来二分区间，如果 $a_m < a_{m+1}$ 则将 r 修改为 m 否则将 l 修改为 m 于是可以使用 $\log n$ 次询问得到答案。

ps. 比赛时乱搞了一个单测试点正确率为 95% 的随机算法，我当时脑子指定是有什么问题。

```
int n;
int query(int pos){
    if(pos==n+1)
        return n+1;
    else{
        printf("? %d\n",pos);
        fflush(stdout);
        return read_int();
    }
}
void ans(int pos){
    printf("! %d\n",pos);
    fflush(stdout);
}
int main()
{
    n=read_int();
    int lef=1,rig=n,mid;
    while(lef<rig){
        mid=lef+rig>>1;
        if(query(mid)<query(mid+1))
            rig=mid;
        else
            lef=mid+1;
    }
```

```
}  
ans(lef);  
return 0;  
}
```

C. Continuous City

题意

给定 $[L,R]$ 要求构造一张有向图，图中最多有 32 个点。

使得所有边均从编号小的点指向编号大的点，且从点 1 到图中编号最大的点的所有路径权值互异，且正好构成集合 $[L,R]$

题解

先将 $[L,R]$ 转化为 $[0,R-L]$ 再想一个 $[0,2^k-1]$ 的构造，最后再调整一下就可以得到答案。

```
const int MAXN=33;  
struct Edge{  
    int u,v,w;  
};  
vector<Edge> Edges;  
int v[MAXN];  
int main()  
{  
    int L=read_int(),R=read_int(),pos=31;  
    puts("YES");  
    _rep(i,2,32)Edges.push_back(Edge{1,i,L});  
    for(int i=31;i>1;i--)v[i]=1<<(31-i);  
    R-=L;  
    while(R){  
        int offset=v[pos];  
        _for(i,pos+1,32){  
            if(R>=v[i]){  
                R-=v[i];  
                Edges.push_back(Edge{pos,i,offset-v[i]});  
                offset+=v[i];  
            }  
        }  
        if(R){  
            R--;  
            Edges.push_back(Edge{pos,32,offset});  
        }  
        pos--;  
    }  
}
```

```

printf("%d %d\n", 32, (int)Edges.size());
_for(i, 0, Edges.size())
printf("%d %d %d\n", Edges[i].u, Edges[i].v, Edges[i].w);
return 0;
}

```

D. Odd Mineral Resource

题意

给定一棵点权树 m 次询问。每次询问路径 u_i 到 v_i 上是否有权值出现了奇数次且 $\ln[l_i, r_i]$

题解 1

树上莫队 bitset 维护每个询问对应路径上的所有权值出现次数。

处理询问时，调用 $\text{bitset_Find_next}(l_i - 1)$ 函数找到从 l_i 起第一个非零的位置然后判定是否不超过 r_i

时间复杂度 $O(n\sqrt{m} + \frac{nm}{w})$

ps. 最开始人比较傻用了 $\text{bitset} \gg l_i$ 来处理查询，极限卡常 4991ms/5000ms

```

const int MAXN=3e5+5;
int blk_sz;
struct Edge{
    int to,next;
}edge[MAXN<<1];
int head[MAXN],edge_cnt,dfn1[MAXN],dfn2[MAXN],invn[MAXN<<1],dfs_t;
void AddEdge(int u,int v){
    edge[++edge_cnt]=Edge{v,head[u]};
    head[u]=edge_cnt;
}
void dfs(int u,int fa){
    dfn1[u]=++dfs_t;
    invn[dfs_t]=u;
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==fa)continue;
        dfs(v,u);
    }
    dfn2[u]=++dfs_t;
    invn[dfs_t]=u;
}
namespace LCA{
    int d[MAXN],sz[MAXN],f[MAXN];
}

```

```
int h_son[MAXN], mson[MAXN], p[MAXN];
void dfs_1(int u, int fa, int depth) {
    sz[u] = 1; f[u] = fa; d[u] = depth; mson[u] = 0;
    for (int i = head[u]; i; i = edge[i].next) {
        int v = edge[i].to;
        if (v == fa) continue;
        dfs_1(v, u, depth + 1);
        sz[u] += sz[v];
        if (sz[v] > mson[u])
            h_son[u] = v, mson[u] = sz[v];
    }
}

void dfs_2(int u, int top) {
    p[u] = top;
    if (mson[u]) dfs_2(h_son[u], top);
    for (int i = head[u]; i; i = edge[i].next) {
        int v = edge[i].to;
        if (v == f[u] || v == h_son[u]) continue;
        dfs_2(v, v);
    }
}

void init(int root) { dfs_1(root, 0, 0); dfs_2(root, root); }
int query(int u, int v) {
    while (p[u] != p[v]) {
        if (d[p[u]] < d[p[v]]) swap(u, v);
        u = f[p[u]];
    }
    return d[u] < d[v] ? u : v;
}

};

struct query {
    int l, r, idx, vl, vr;
    bool operator < (const query &b) const {
        if (l / blk_sz != b.l / blk_sz) return l < b.l;
        return ((l / blk_sz) & 1) ? (r < b.r) : (r > b.r);
    }
} q[MAXN];

int a[MAXN], ans[MAXN];
bitset<MAXN> s;

int query(int vl, int vr) {
    int v = s._Find_next(vl - 1);
    if (v > vr) return -1;
    else return v;
}

int main()
{
    int n = read_int(), m = read_int();
    blk_sz = n / sqrt(m) + 1;
```

```

_rep(i,1,n)a[i]=read_int();
_for(i,1,n){
    int u=read_int(),v=read_int();
    AddEdge(u,v);
    AddEdge(v,u);
}
LCA::init(1);
dfs(1,0);
_rep(i,1,m){
    int u=read_int(),v=read_int(),l=read_int(),r=read_int();
    if(dfn1[u]>dfn1[v])swap(u,v);
    if(LCA::query(u,v)==u)
        q[i].l=dfn1[u],q[i].r=dfn1[v];
    else
        q[i].l=dfn2[u],q[i].r=dfn1[v];
    q[i].idx=i,q[i].vl=l,q[i].vr=r;
}
sort(q+1,q+m+1);
int lef=1,rig=0;
_rep(i,1,m){
    while(lef>q[i].l)s.flip(a[invn[--lef]]);
    while(rig<q[i].r)s.flip(a[invn[++rig]]);
    while(lef<q[i].l)s.flip(a[invn[lef++]]);
    while(rig>q[i].r)s.flip(a[invn[rig--]]);
    int u=invn[q[i].l],v=invn[q[i].r],p=LCA::query(u,v);
    if(u==p)
        ans[q[i].idx]=query(q[i].vl,q[i].vr);
    else{
        s.flip(a[p]);
        ans[q[i].idx]=query(q[i].vl,q[i].vr);
        s.flip(a[p]);
    }
}
_rep(i,1,m)
enter(ans[i]);
return 0;
}

```

题解 2

首先给 $1 \sim n$ 每个权值一个 $\text{unsigned long long}$ 范围内的 hash 值。

于是 $v_1, v_2, \dots, v_k$ 中有某个数出现奇数次近似等效为 $\text{hash}(v_1) \oplus \text{hash}(v_2) \oplus \dots \oplus \text{hash}(v_k) \neq 0$

定义 $f(u, l, r)$ 表示从根节点到 u 所有权值在 $[l, r]$ 的权值的 hash 值的异或和。

于是询问等效于查询 $f(u, l, r) \oplus f(v, l, r) \oplus f(\text{lca}(u, v), l, r) \oplus f(\text{p}(\text{lca}(u, v)), l, r)$ 是否等于 0 。

主席树维护根节点到每个节点的权值区间异或和，查询时先找到权值区间内不为 0 的范围，然后再 $\log n$ 找到答案即可。

时间复杂度 $O(n \log n)$

```
const int MAXN=3e5+5;
mt19937_64 mt(time(NULL));
struct Node{
    int ch[2];
    unsigned long long val;
}node[MAXN*40];
#define lch(k) node[node[k].ch[0]]
#define rch(k) node[node[k].ch[1]]
#define f(k1,k2,k3,k4)
(node[k1].val^node[k2].val^node[k3].val^node[k4].val)
int root[MAXN],node_cnt;
int nodecopy(int k){
    node[++node_cnt]=node[k];
    return node_cnt;
}
void update(int &k,int p,int lef,int rig,int pos,unsigned long long v){
    k=nodecopy(p);
    node[k].val^=v;
    if(lef==rig)return;
    int mid=lef+rig>>1;
    if(mid>=pos)
        update(node[k].ch[0],node[p].ch[0],lef,mid,pos,v);
    else
        update(node[k].ch[1],node[p].ch[1],mid+1,rig,pos,v);
}
int query_ans(int k1,int k2,int k3,int k4,int lef,int rig){
    int mid=lef+rig>>1;
    if(lef==rig)return mid;
    if(f(node[k1].ch[0],node[k2].ch[0],node[k3].ch[0],node[k4].ch[0])!=0)
        return
        query_ans(node[k1].ch[0],node[k2].ch[0],node[k3].ch[0],node[k4].ch[0],lef,m
        id);
    else
        return
        query_ans(node[k1].ch[1],node[k2].ch[1],node[k3].ch[1],node[k4].ch[1],mid+1
        ,rig);
}
int query(int k1,int k2,int k3,int k4,int lef,int rig,int ql,int qr){
    if(ql<=lef&&rig<=qr){
        if(f(k1,k2,k3,k4)==0)
            return -1;
        else
            return query_ans(k1,k2,k3,k4,lef,rig);
    }
}
```

```

    int mid=lef+rig>>1;
    if(mid>=qr)
        return
    query(node[k1].ch[0],node[k2].ch[0],node[k3].ch[0],node[k4].ch[0],lef,mid,q
    l,qr);
    else if(mid<ql)
        return
    query(node[k1].ch[1],node[k2].ch[1],node[k3].ch[1],node[k4].ch[1],mid+1,rig
    ,ql,qr);
    int
    temp=query(node[k1].ch[0],node[k2].ch[0],node[k3].ch[0],node[k4].ch[0],lef,
    mid,ql,qr);
    if(temp!=-1)return temp;
    return
    query(node[k1].ch[1],node[k2].ch[1],node[k3].ch[1],node[k4].ch[1],mid+1,rig
    ,ql,qr);
}
struct Edge{
    int to,next;
}edge[MAXN<<1];
int head[MAXN],edge_cnt;
void Insert(int u,int v){
    edge[++edge_cnt]=Edge{v,head[u]};
    head[u]=edge_cnt;
}
namespace LCA{
    int d[MAXN],sz[MAXN],f[MAXN];
    int h_son[MAXN],mson[MAXN],p[MAXN];
    void dfs_1(int u,int fa,int depth){
        sz[u]=1;f[u]=fa;d[u]=depth;mson[u]=0;
        for(int i=head[u];i;i=edge[i].next){
            int v=edge[i].to;
            if(v==fa)
                continue;
            dfs_1(v,u,depth+1);
            sz[u]+=sz[v];
            if(sz[v]>mson[u])
                h_son[u]=v,mson[u]=sz[v];
        }
    }
    void dfs_2(int u,int top){
        p[u]=top;
        if(mson[u])dfs_2(h_son[u],top);
        for(int i=head[u];i;i=edge[i].next){
            int v=edge[i].to;
            if(v==f[u]||v==h_son[u])
                continue;
            dfs_2(v,v);
        }
    }
    void init(int root){dfs_1(root,0,0);dfs_2(root,root);}
}

```

```
int query(int u,int v){
    while(p[u]!=p[v]){
        if(d[p[u]]<d[p[v]]) swap(u,v);
        u=f[p[u]];
    }
    return d[u]<d[v]?u:v;
}

};
int a[MAXN],n;
unsigned long long h[MAXN];
void dfs(int u,int fa){
    update(root[u],root[fa],1,n,a[u],h[a[u]]);
    for(int i=head[u];i;i=edge[i].next){
        int v=edge[i].to;
        if(v==fa) continue;
        dfs(v,u);
    }
}

int main()
{
    n=read_int();
    int q=read_int();
    _rep(i,1,n) a[i]=read_int();
    _rep(i,1,n) h[i]=mt();
    _for(i,1,n){
        int u=read_int(),v=read_int();
        Insert(u,v);
        Insert(v,u);
    }
    LCA::init(1);
    dfs(1,0);
    while(q--){
        int
u=read_int(),v=read_int(),ql=read_int(),qr=read_int(),p=LCA::query(u,v);
        enter(query(root[u],root[v],root[p],root[LCA::f[p]],1,n,ql,qr));
    }
    return 0;
}
```

From:
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:contest:cf_700_div_1

Last update: 2021/02/10 11:29