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 $a_i\lt a_{i-1}$ 且 $a_i\lt a_{i+1}$。假定 $a_0=a_{n+1}=\infty$。

题解

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

接下来二分区间，如果 $a_m\lt a_{m+1}$ 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 $u_i\to v_i$ 上是否有权值出线了奇数次且 $\in [l_i,r_i]$

题解 1

树上莫队 $+\text{bitset}$ 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 $O\left(n\sqrt m+\frac {nm}w\right)$

```
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;
}
```