[Splay] BZOJ 3729 Gty的游戏

〃

题解：

博弈论+Splay维护dfs序

想会做这道题首先要知道两个Nim游戏的经典变形。

第一个是加入一次只能选m个的限制。

第二个是“阶梯博弈”(POJ 1704)，就是有一个楼梯，每次可以把一阶的任意个棋子移到下面一个台阶，不能移动（0号台阶不能向下移动）的玩家输。

第一个问题的解决方法是把所有的数 mod (m+1)，因为显然加入这个限制之后每个子游戏的sg函数值变成了sg(n) = n%(m+1)。

第二个问题可以转换成Nim游戏，方法是如果对方移动了偶数层的棋子，那么你下一步可以把他刚移动的棋子再向下移动。这样偶数层上的棋子就可以视为不存在了，如果把一个奇数层的棋子移动到下一层，那么我们把它看成消失了，这样就变成了Nim游戏，也就是说只用考虑奇数层的棋子sg函数的异或值就行了。

在树上也同理，对于任意一棵子树，如果把根深度定义为0，那么也只要考虑深度为奇数的异或和。

题目就变成了支持动态修改，加点，维护子树信息，随便用个什么数据结构维护一下dfs序就行了。

〃

传送门：http://timeplayer.blog.163.com/blog/static/203718254201410206373276 3/

Orzzz

```cpp
#include<cstdio>
#include<cstdlib>
#include<map>
#include<algorithm>
#define V G[p].v
using namespace std;

inline char nc()
{
```

```
    static char buf[100000],*p1=buf,*p2=buf;
    if (p1==p2) { p2=(p1=buf)+fread(buf,1,100000,stdin); if (p1==p2) return EOF; }
    return *p1++;
}

inline void read(int &x)
{
    char c=nc(),b=1;
    for (;!(c>='0' && c<='9');c=nc()) if (c=='-') b=-1;
    for (x=0;c>='0' && c<='9';x=x*10+c-'0',c=nc()); x*=b;
}

struct Splay{
    #define oo 1<<30
    #define ND_MAX 500005
    struct node{
        int a[2],d,val;
        int size;
        node *p,*ch[2];
        bool dir() { return p->ch[1]==this; }
        void setc(node *x,int d) { ch[d]=x; x->p=this; }
        void update(){
            size=ch[0]->size+ch[1]->size+1;
            a[0]=(ch[0]->a[0])^(ch[1]->a[0]);
            a[1]=(ch[0]->a[1])^(ch[1]->a[1]);
            a[d]^=val;
        }
    }*root,*null;
    node Mem[ND_MAX],*Stack[ND_MAX];
    int top;
    inline void init_Memory(){
        for (int i=0;i<ND_MAX;i++) Stack[i]=Mem+i;
        top=ND_MAX-1;
    }
    inline node* New_Node(){
        node *p=Stack[top--];
        p->p=p->ch[0]=p->ch[1]=null;
        p->size=1;
        return p;
    }
    inline void Del_Node(node *p){
        Stack[++top]=p;
    }
    Splay() { init_Memory(); root=null=New_Node(); null->p=null->ch[1]=null->ch[0]=null; null->size=0; }
    inline void rot(node *x){
        if (x==null) return;
        if (x->p==root) root=x;
        bool d=x->dir(); node *p=x->p;
        if (p->p!=null) p->p->setc(x,p->dir()); else x->p=null;
        p->setc(x->ch[d^1],d); x->setc(p,d^1); x->update(); p->update();
    }
    inline void splay(node *&rt,node *x){
        if (x==null) return;
        while (x!=rt)
            if (x->p==rt)
                rot(x);
```

```
66.             else
67.                 x->dir()==x->p->dir()?(rot(x->p),rot(x)):(rot(x),rot(x));
68.         rt=x; x->update();
69.     }
70.     inline void insert(node *z){
71.         node *x=root,*y=null;
72.         if (root==null) { root=z; return; }
73.         while (x!=null)
74.             y=x,x=x->ch[1];
75.         y->setc(z,1);
76.         splay(root,z);
77.     }
78.     inline node* findkth(node *&rt,int k){
79.         if (k>rt->size) return null;
80.         node *x=rt;
81.         while (k){
82.             if (k==x->ch[0]->size+1) break;
83.             k>x->ch[0]->size+1?k-=x->ch[0]->size+1,x=x->ch[1]:x=x->ch[0];
84.         }
85.         splay(root,x); return x;
86.     }
87.     inline node *nxt(node *x){
88.         node *p=x->ch[1];
89.         while (p->ch[0]!=null) p=p->ch[0];
90.         return p;
91.     }
92.     inline void print(node *x){
93.         if (x==null) return;
94.         printf("%d",x->val);
95.         putchar('('); print(x->ch[0]); putchar(')');
96.         putchar('('); print(x->ch[1]); putchar(')');
97.     }
98. }splay;
99.
100. struct edge{
101.     int u,v,next;
102. };
103.
104. int head[200005],inum;
105. edge G[400005];
106.
107. inline void add(int u,int v,int p)
108. {
109.     G[p].u=u; G[p].v=v; G[p].next=head[u]; head[u]=p;
110. }
111.
112. map<int,int>id;
113. int n,L,ynum;
114. int w[200005],d[200005];
115. Splay::node *pos[400005],*last[400005];
116.
117. inline void dfs(int u,int fa)
118. {
119.     d[u]=d[fa]^1;
120.     Splay::node *p=splay.New_Node();
121.     p->d=d[u]; p->val=w[u];
122.     splay.insert(p);
```

```
123.     pos[u]=p;
124.     for (int p=head[u];p;p=G[p].next)
125.         if (V!=fa)
126.             dfs(V,u);
127.     p=splay.New_Node();
128.     splay.insert(p);
129.     last[u]=p;
130. }
131. 
132. int main()
133. {
134.     int _u,_v,_x,Q,order,Xor;
135.     freopen("t.in","r",stdin);
136.     freopen("t.out","w",stdout);
137.     read(n); read(L);
138.     for (int i=1;i<=n;i++)
139.         read(w[i]),w[i]%=(L+1),id[i]=i;
140.     for (int i=1;i<n;i++)
141.         read(_u),read(_v),add(_u,_v,++inum),add(_v,_u,++inum);
142.     dfs(1,0);
143.     for (int i=1;i<=10;i++)
144.         splay.splay(splay.root,pos[rand()%n+1]);
145. //  splay.print(splay.root); printf("\n");
146.     read(Q);
147.     while (Q--)
148.     {
149.         read(order);
150.         if (order==1){
151.             read(_u); _u^=ynum; _u=id[_u];
152.             splay.splay(splay.root,pos[_u]);
153. //          splay.print(splay.root);printf("\n");
154.             splay.splay(splay.root->ch[1],last[_u]);
155. //          splay.print(splay.root);printf("\n");
156.             Xor=splay.root->ch[1]->ch[0]->a[d[_u]^1];
157.             if (Xor==0)
158.                 printf("GTY\n");
159.             else
160.                 ynum++,
161.                 printf("MeiZ\n");
162.         }
163.         else if (order==2){
164.             read(_u); read(_v); _u^=ynum; (_v^=ynum)%=(L+1); _u=id[_u];
165.             splay.splay(splay.root,pos[_u]);
166. //          splay.print(splay.root);printf("\n");
167.             splay.root->val=_v;
168.             splay.root->update();
169.         }
170.         else if (order==3){
171.             read(_u); read(_v); read(_x); _u^=ynum; _v^=ynum; _x^=ynum; _x%=(L+1);
172.             id[_v]=++n; _u=id[_u];
173.             d[n]=d[_u]^1;
174.             pos[n]=splay.New_Node();
175.             pos[n]->d=d[_u]^1; pos[n]->val=_x;
176.             last[n]=splay.New_Node();
177.             splay.splay(splay.root,pos[_u]);
178. //          splay.print(splay.root);printf("\n");
179.             Splay::node *t=splay.nxt(pos[_u]);
```

```
180.                splay.splay(splay.root->ch[1],t);
181.  //            splay.print(splay.root);printf("\n");
182.                splay.root->ch[1]->setc(pos[n],0);
183.                splay.root->ch[1]->ch[0]->setc(last[n],1);
184.                splay.root->ch[1]->ch[0]->update();
185.                splay.root->ch[1]->update();
186.                splay.root->update();
187.  //            splay.print(splay.root);printf("\n");
188.            }
189.        }
190.        return 0;
191.  }
```