

这是一个复杂版的文艺splay，因为里面包含了很多操作，比如维护最大子段和，序列求和，序列修改为某值，在序列中某位置加入一段序列（直接建出这一段序列的平衡树再插入），删除某一段序列（平衡树get到删除的序列之后直接删除），同时还有将删除的节点回收利用等操作

#include<bits/stdc++.h>

#define ll long long

#define inf 1e18

#define mn 1000005

using namespace std;

ll rt=0,ans=0,tot=0,n,m,a[mn],

c[mn][2],v[mn],fa[mn],mx[mn],s[mn],lx[mn],rx[mn],id[mn],

sum[mn],pos[mn],tag[mn],rev[mn];

queue<ll>q;

//mx[]表示的是子树里面最大子段和，lx[]是子树里面靠在左侧的最大子段和，rx[]同理

//sum[]表示的是子树的权值和，s[]表示树的节点总数，

//v[]表示节点的权值，tag[]是懒惰标记，表示这个节点下的节点是否全部被改成了某个数，

//rev[]是懒惰标记，表示的是这个节点下是否需要旋转

inline void in(ll &x)

{

ll s=0,w=1;

char ch=getchar();

while(ch<'0'||ch>'9'){if(ch=='-')w=-1;ch=getchar();}

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

x=s\*w;

}

void up(ll x)//上传标记

{ll l=c[x][0],r=c[x][1];

sum[x]=sum[l]+sum[r]+v[x];

s[x]=s[l]+s[r]+1;

mx[x]=max(mx[l],max(mx[r],rx[l]+v[x]+lx[r]));

lx[x]=max(lx[l],sum[l]+v[x]+lx[r]);

rx[x]=max(rx[r],sum[r]+v[x]+rx[l]);

}

void down(ll x)//下传标记

{ll l=c[x][0],r=c[x][1];

if(tag[x])

{

if(l){tag[l]=1;v[l]=v[x];sum[l]=v[x]\*s[l];lx[l]=rx[l]=max(sum[l],0ll);mx[l]=max(sum[l],v[l]);}

if(r){tag[r]=1;v[r]=v[x];sum[r]=v[x]\*s[r];lx[r]=rx[r]=max(sum[r],0ll);mx[r]=max(sum[r],v[r]);}

tag[x]=0;

}

if(rev[x])

{

rev[x]=0;rev[l]^=1;rev[r]^=1;

swap(lx[l],rx[l]);swap(lx[r],rx[r]);

swap(c[l][0],c[l][1]);swap(c[r][0],c[r][1]);

}

}

void rot(ll x,ll &f)//旋转操作

{ll y=fa[x],z=fa[y],l,r;

l=(c[y][0]!=x);r=l^1;

if(y==f)f=x;

else {if(c[z][0]==y)c[z][0]=x;else c[z][1]=x;}

fa[x]=z;fa[y]=x;fa[c[x][r]]=y;

c[y][l]=c[x][r];c[x][r]=y;

up(y);up(x);

}

void spl(ll x,ll &f)//把x节点旋转到f这个位置

{

while(x!=f)

{

ll y=fa[x],z=fa[y];

if(y!=f){if(c[y][0]==x^c[z][0]==y)rot(x,f);else rot(y,f);}

rot(x,f);

}

}

ll find(ll x,ll k)//在x为根的子树中找序列中的第k个是哪个节点

{

down(x);

ll l=c[x][0],r=c[x][1];

if(s[l]+1==k)return x;

if(s[l]>=k)return find(l,k);

else return find(r,k-s[l]-1);

}

ll get(ll l,ll r)//将区间(l,r)旋转到根的右儿子的左儿子上，返回根的右儿子

{ll x=find(rt,l),y=find(rt,r+2);

spl(x,rt);

spl(y,c[rt][1]);

return y;

}

void change(ll l,ll r,ll val)//将l到r间的序列改成val

{ll y=get(l,r),x=c[y][0];

v[x]=val;tag[x]=1;sum[x]=s[x]\*val;

if(val>=0)lx[x]=rx[x]=mx[x]=sum[x];

else {lx[x]=rx[x]=0;mx[x]=val;}

up(y);up(fa[y]);

}

void ask(ll k)//询问序列中的第k个的权值是多少

{ll x=find(rt,k+1);

spl(x,rt);

printf("%lld\n",v[rt]);

}

void Sum(ll l,ll r)//求l到r间的权值和

{ll y=get(l,r);

printf("%lld\n",sum[c[y][0]]);

}

void Rev(ll l,ll r)//翻转l到r的这个区间

{ll y=get(l,r),x=c[y][0];

if(tag[x]==0)

{

rev[x]^=1;

swap(c[x][0],c[x][1]);

swap(lx[x],rx[x]);

up(y);up(fa[y]);

}

}

void recycle(ll x)//回收以x为根节点的平衡树

{ll &l=c[x][0],&r=c[x][1];

if(l)recycle(l);

if(r)recycle(r);

q.push(x);

fa[x]=l=r=tag[x]=rev[x]=0;

}

void del(ll l,ll r)//将l到r间的序列直接删除

{ll y=get(l,r);

recycle(c[y][0]);c[y][0]=0;

up(y);up(fa[y]);

}

void bui(ll l,ll r,ll f)//直接建立出l到r之间的平衡树

{ll mid=(l+r)/2,x=id[mid],prt=id[f];

if(l==r)

{

mx[x]=sum[x]=a[l];

tag[x]=rev[x]=0;

lx[x]=rx[x]=max(a[l],0ll);

s[x]=1;

}

if(l<mid)bui(l,mid-1,mid);

if(mid<r)bui(mid+1,r,mid);

v[x]=a[mid];fa[x]=prt;

up(x);c[prt][mid>=f]=x;

}

void ins(ll k,ll all)//在序列的第k项后面插入all个数

{ll x,y,z,i;

for(i=1;i<=all;i++)in(a[i]);

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

if(!q.empty()){id[i]=q.front();q.pop();}

else id[i]=++tot;

bui(1,all,0);

z=id[(1+all)>>1];

x=find(rt,k+1);y=find(rt,k+2);

spl(x,rt);spl(y,c[x][1]);

fa[z]=y;c[y][0]=z;

up(y);up(x);

}

void dfs(ll x)//遍历平衡树输出序列,调试用

{

down(x);

if(c[x][0])dfs(c[x][0]);

if(abs(v[x])!=inf)printf("%lld ",v[x]);

if(c[x][1])dfs(c[x][1]);

}

int main()

{

ll x,y,i,j,k,z;

char ch[10];

cin>>n>>m;

mx[0]=a[1]=a[n+2]=-inf;//因为up的时候l或r可能等于0

for(i=1;i<=n;i++)in(a[i+1]);

for(i=1;i<=n+2;i++)id[i]=i;

bui(1,n+2,0);

rt=(n+3)/2;tot=n+2;

while(m--)

{

scanf("%s",ch);

if(ch[0]!='M'||ch[2]!='X'){in(k);in(x);}

if(ch[0]=='I')ins(k,x);

if(ch[0]=='D')del(k,k+x-1);

if(ch[0]=='M')

{

if(ch[2]=='X')printf("%lld\n",mx[rt]);

else {in(z);change(k,k+x-1,z);}

}

if(ch[0]=='R')Rev(k,k+x-1);

if(ch[0]=='G')Sum(k,k+x-1);

}

return 0;

}