平衡树之FHQ Treap

前置芝士

BST

平衡树的基本概念

Treap

“Treap=BST+Heap”

它整体是拥有BST的性质的，但每一个节点都有一个附加权值，它的附加权值符合堆的性质。因此树的形态以及平衡与否都是由这个附加权值来决定的。而这个权值是随机选取的，因此这棵树**大概平衡。**

FHQ Treap

普通的Treap很难实现恼人的旋转。FHQ Treap应运而生。

这东西只有两种操作：分离和合并，但是可以实现所有BST、Splay、Treap能搞的东西。

Split

将一颗Treap树拆成两颗。

按权值分配

```
void split(int now,int k,int &x,int &y){
    if(!now) x=y=0;
    else{
        if(val[now]<=k)
            x=now,split(ch[now][1],k,ch[now][1],y);
        else
            y=now,split(ch[now][0],k,x,ch[now][0]);
        update(now);
    }
}
```

对于所有的点，如果它的权值小于k，那么它的所有左子树就放在左边的树里，如何遍历其右儿子；如果大于k,把它的所有右子树分到右边的树里，遍历左儿子。

按前k个分配

```
void split(int now,int k,int &x,int &y){
    if(!now) x=y=0;
    else{
        if(k<=sz[ch[now][0]])
            y=now,split(ch[now][0],k,x,ch[now][0]);
        else
            x=now,split(ch[now][1],k-sz[ch[now][0]]-1,ch[now][1],y);
        update(now);
    }
}
```

merge

把两个Treap合成一个，保证第一个的权值小于第二个。

```
int merge(int x,int y){
    if(!x||!y) return x+y;
    if(pri[x]<pri[y]){
        ch[x][1]=merge(ch[x][1],y);
        update(x);
        return x;
    }else{
        ch[y][0]=merge(x,ch[y][0]);
        update(y);
        return y;
    }
}
```

其他操作

insert：插入元素

插入一个权值为v的点，就把树按v拆分成x和y两部分，把v看成是一棵树，先和x合并，再和y合并就变成了插入之后的树。

```
split(root,v,x,y);
root=merge(merge(x,new_node(v)),y);
```

del：删除元素

删除权值为v的点，把树按v位最大值分成x和z两部分，此时x树中的最大权值为v，然后按v-1将其分成x和y两部分，此时x中最大权值为v-1,权值为y的结点一定为y的根节点，然后我们无视y的根节点，再将其他部分拼回来。

```
split(root,v,x,z);
split(x,v-1,x,y);
y=merge(ch[y][0],ch[y][1]);
root=merge(merge(x,y),z);
```

pre：寻找前驱

找前驱就把root按v-1分成x和y,在x里面找最大的即可。

```
split(root,v-1,x,y);
printf("%d\n",val[kth(x,sz[x])]);
root=merge(x,y);
```

nxt：寻找后继

找后继就把root按v分成x和y，在y里面找最小的。

```
split(root,v,x,y);
printf("%d\n",val[kth(y,1)]);
root=merge(x,y);
```

rank：查询值v的排名

把root按v-1拆分成x,y，v的排名就是x树的sz

```
split(root,v-1,x,y);
printf("%d\n",sz[x]+1);
root=merge(x,y);
```

kth：查询第k小的数

因为是BST，所以和其他平衡树的做法是一样的，k小于左子树大小则迭代，否则减去。

```
printf("%d\n",val[kth(rt,k)]);
```

区间操作

先把root的前r个split出来，再把前面的子树中split前l-1个，如何我们就得到了操作区间。详见例题。

例题

luoguP3369 【模板】普通平衡树

包含插入、删除、求第k大值，求值的排名，求前驱后继的经典模板。

```
#include<bits/stdc++.h>
//#define int long long
using namespace std;
const int N=1e5+7;

int read(){ int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}

int ch[N][3],val[N],pri[N],siz[N];//左右儿子,数值,随机权值,节点数量
int n,sz;//总结点的数量

void update(int x){
    siz[x]=1+siz[ch[x][0]]+siz[ch[x][1]];
}

int new_node(int v){
    siz[++sz]=1;// 新开辟一个节点
    val[sz]=v;
    pri[sz]=rand();
    return sz;
}

int merge(int x,int y){ // 合并操作
    if(!x||!y) return x+y; //x和y中必定有一个是0
    if(pri[x]<pri[y]){ //比较两棵树根的随机权值
        ch[x][1]=merge(ch[x][1],y);  //把x加到左边的树上
        update(x); //更新大小
        return x;
    }else{
        ch[y][0]=merge(x,ch[y][0]);
        update(y);
        return y;
    }
}
void split(int now,int k,int &x,int &y) {//分离操作
    if(!now) x=y=0;// 到达叶子节点
    else{
        if(val[now]<=k) x=now,split(ch[now][1],k,ch[now][1],y); // 分离右子树
        else y=now,split(ch[now][0],k,x,ch[now][0]);
        update(now);
    }
}
int kth(int now,int k){// 查询排名
    while(1){
```

```
        if(k<=siz[ch[now][0]]) now=ch[now][0];// 在左子树中，且数量小于左子树的大小，迭代寻找
        else if(k==siz[ch[now][0]]+1) return now;// 找到了
        else k-=siz[ch[now][0]]+1,now=ch[now][1];// 去右子树找
    }
}

signed main(){
    srand(time(0));
    n=read();
    int root=0,x,y,z,op,a;
    for(int i=1;i<=n;i++){
        op=read();a=read();
        if(op==1){// 插入
            split(root,a,x,y);
            root=merge(merge(x,new_node(a)),y);
        }else if(op==2){//删除x
            split(root,a,x,z);
            split(x,a-1,x,y);
            y=merge(ch[y][0],ch[y][1]);
            root=merge(merge(x,y),z);
        }else if(op==3){ //查询x的排名
            split(root,a-1,x,y);
            printf("%d\n",siz[x]+1);
            root=merge(x,y);
        }else if(op==4){  // 查询排名为a的数
            printf("%d\n",val[kth(root,a)]);
        }else if(op==5){   // 求a的前驱
            split(root,a-1,x,y);
            printf("%d\n",val[kth(x,siz[x])]);
            root=merge(x,y);
        }else if(op==6){ // 求a的后继
            split(root,a,x,y);
            printf("%d\n",val[kth(y,1)]);
            root=merge(x,y);
        }
    }
    return 0;
}
```

luoguP3391 【模板】文艺平衡树

这是一道平衡树的区间操作的模板题。

```
#include<bits/stdc++.h>
//#define int long long
using namespace std;
const int N=2e5+7;

int read(){ int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}

int ch[N][2],val[N],pri[N],sz[N],mrk[N],cnt,rt,n,m;
```

```
void update(int x){
    sz[x]=1+sz[ch[x][0]]+sz[ch[x][1]];
}

void pushdown(int x){
    if(x&&mrk[x]){
        mrk[x]=0;
        swap(ch[x][0],ch[x][1]);
        if(ch[x][0]) mrk[ch[x][0]]^=1;
        if(ch[x][1]) mrk[ch[x][1]]^=1;
    }
}

int new_node(int v){
    sz[++cnt]=1;
    val[cnt]=v;
    pri[cnt]=rand();
    return cnt;
}

int merge(int x,int y){
    if(!x||!y) return x+y;
    pushdown(x);pushdown(y);
    if(pri[x]<pri[y]){
        ch[x][1]=merge(ch[x][1],y);
        update(x);
        return x;
    }else{
        ch[y][0]=merge(x,ch[y][0]);
        update(y);
        return y;
    }
}

void split(int now,int k,int &x,int &y){
    if(!now) x=y=0;
    else{
        pushdown(now);
        if(k<=sz[ch[now][0]])
            y=now,split(ch[now][0],k,x,ch[now][0]);
        else
            x=now,split(ch[now][1],k-sz[ch[now][0]]-1,ch[now][1],y);
        update(now);
    }
}

int build(int l,int r){
    if(l>r) return 0;
    int mid=l+r>>1,v=mid-1;
    int now=new_node(v);
    ch[now][0]=build(l,mid-1);
    ch[now][1]=build(mid+1,r);
    update(now);
    return now;
}

void dfs(int x){
```

```
    if(!x) return;
    pushdown(x);
    dfs(ch[x][0]);
    if(val[x]>=1&&val[x]<=n) printf("%d ",val[x]);
    dfs(ch[x][1]);
}

void res(int l,int r){
    int a,b,c,d;
    split(rt,r+1,a,b);
    split(a,l,c,d);
    mrk[d]^=1;
    rt=merge(merge(c,d),b);
}

signed main(){
    srand(time(0));
    scanf("%d%d",&n,&m);
    rt=build(1,n+2);
    for(int l,r,i=1;i<=m;i++){
        l=read();r=read();res(l,r);
    }
    dfs(rt);
    puts("");
    return 0;
}
```

luoguP2042 [NOI2005] 维护数列

综合性比较强的著名毒瘤题，支持区间插入、区间删除以及一些经典操作。

1	**插入**	INSERT $posi\ tot\ c_1\ c_2 \cdots c_{tot}$	**在当前数列的第 posi 个数字后插入 tot 个数字：$c_1, c_2 \cdots c_{tot}$ * c*；若在数列首插入，则 posi为 0**
2	删除	DELETE $posi\ tot$	从当前数列的第 posi个数字开始连续删除 tot个数字
3	修改	MAKE-SAME $posi\ tot\ c$	从当前数列的第 posi个数字开始的连续 tot个数字统一修改为 c
4	翻转	REVERSE $posi\ tot$	取出从当前数列的第 posi个数字开始的 tot个数字，翻转后放入原来的位置
5	求和	GET-SUM $posi\ tot$	计算从当前数列的第 posi个数字开始的 tot 个数字的和并输出
6	求最大子列和	MAX-SUM	求出当前数列中和最大的一段子列，并输出最大和

```
#include<bits/stdc++.h>
using namespace std;
const int N=5e5+7;
const int mod=1e9+7;
```

```
int read(){ int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}

struct FHQ{
    int ls,rs,sz,val,cov; //左儿子，右儿子，树大小，权值和标记值
    bool lzy,tag; //两个标记，翻转区间和区间赋值
    int sum,maxqz,maxhz,maxzd; //区间和,区间最大前缀,区间最大后缀,区间最大子段和
}T[N];

int stk[N],len;  //栈
int idx,rt; //结点个数和树根
char op[10]; //指令

inline int new_node(int v){  //新建结点
    idx=stk[len--]; //数列中最多有5e5个数，所以可以用栈维护所有结点编号，做到重复利用空间
    T[idx].ls=T[idx].rs=T[idx].lzy=T[idx].cov=0;
    T[idx].sz=1;
    T[idx].val=T[idx].sum=v;
    T[idx].maxqz=T[idx].maxhz=max(0,v); //前缀、后缀可以为空
    T[idx].maxzd=v; //字段不能为空
    return idx; //返回节点编号
}

inline void pushup(int x){ //结点更新信息
    if(!x) return;
    T[x].sz=T[T[x].ls].sz+T[T[x].rs].sz+1;  //子树大小
    T[x].sum=T[T[x].ls].sum+T[T[x].rs].sum+T[x].val; //区间和

T[x].maxqz=max(max(T[T[x].ls].maxqz,T[T[x].ls].sum+T[x].val+T[T[x].rs].maxqz),0)
; //区间最大前缀和

T[x].maxhz=max(max(T[T[x].rs].maxhz,T[T[x].rs].sum+T[x].val+T[T[x].ls].maxhz),0)
; //区间最大后缀和
    T[x].maxzd=max(T[x].val,T[x].val+T[T[x].ls].maxhz+T[T[x].rs].maxqz); //区间最
大字段和
    if(T[x].ls) T[x].maxzd=max(T[x].maxzd,T[T[x].ls].maxzd);
    if(T[x].rs) T[x].maxzd=max(T[x].maxzd,T[T[x].rs].maxzd);
}

inline void reverse(int x){ //区间翻转
    if(!x) return;
    swap(T[x].ls,T[x].rs);
    swap(T[x].maxhz,T[x].maxqz); //注意前后缀也要翻转
    T[x].lzy^=1;
}

inline void cover(int x,int v){ //区间修改
    T[x].val=T[x].cov=v;  //给子树贴标记
    T[x].sum=T[x].sz*v; //修改区间(子树)和
    T[x].maxqz=T[x].maxhz=max(0,T[x].sum);
    T[x].maxzd=max(v,T[x].sum);
    T[x].tag=1;
}

inline void pushdown(int x){ //下传标记
    if(!x) return;
    if(T[x].lzy){
```

```
        if(T[x].ls) reverse(T[x].ls);
        if(T[x].rs) reverse(T[x].rs);
        T[x].lzy=0;
    }
    if(T[x].tag){
        if(T[x].ls) cover(T[x].ls,T[x].cov);
        if(T[x].rs) cover(T[x].rs,T[x].cov);
        T[x].cov=T[x].tag=0;
    }
}

inline void del(int x){ //删除操作
    if(!x) return;
    stk[++len]=x; //用栈维护删掉的数以重复利用可以解决空间问题
    if(T[x].ls) del(T[x].ls);
    if(T[x].rs) del(T[x].rs);
}

inline void split(int x,int &L,int &R,int K){ //分离操作
    if(!x){L=R=0;return;}
    pushdown(x);
    if(T[T[x].ls].sz+1<=K) L=x,split(T[x].rs,T[L].rs,R,K-T[T[x].ls].sz-1);
    else R=x,split(T[x].ls,L,T[R].ls,K);
    pushup(x);
}

inline void merge(int x,int y,int &M){ //合并操作
    if(!x||!y){M=x+y;return;}
    if(mod%(T[x].sz+T[y].sz)<T[x].sz)
pushdown(x),M=x,merge(T[x].rs,y,T[M].rs),pushup(x);//等同于随机权值
    else pushdown(y),M=y,merge(x,T[y].ls,T[M].ls),pushup(y);
}

int n,m,a[N];

inline int build(int l,int r){  //将一个序列构建成树的形式
    if(l==r) return new_node(a[l]);
    int x,mid=((l+r)>>1);
    merge(build(l,mid),build(mid+1,r),x);
    return x;
}

signed main(){
    n=read();m=read();
    for(int i=1;i<=500000;i++) stk[++len]=i;
    for(int i=1;i<=n;i++) a[i]=read();
    merge(rt,build(1,n),rt);
    while(m--){
        scanf("%s",op);
        if(op[0]=='I'){ //区间插入操作
            int pos=read(),tot=read(),x,y;
            split(rt,x,y,pos); //以pos位置拆开再进行三棵树的合并
            for(int i=1;i<=tot;i++) a[i]=read();
            merge(x,build(1,tot),x);
            merge(x,y,rt);
        }else if(op[0]=='D'){ //删除操作
            int pos=read(),tot=read(),x,y,z;
            split(rt,x,y,pos-1); //以pos-1位置拆开
```

```
            split(y,y,z,tot); //再数tot个数才开
            del(y);
            merge(x,z,rt); //进行合并
        }else if(op[0]=='M'&&op[2]=='K'){ //区间修改
            int pos=read(),tot=read(),val=read(),x,y,z;
            split(rt,x,y,pos-1);
            split(y,y,z,tot);
            cover(y,val);  //将区间分离出一棵树来然后进行修改
            merge(x,y,x);
            merge(x,z,rt);
        }else if(op[0]=='R'){ //翻转区间
            int pos=read(),tot=read(),x,y,z;
            split(rt,x,y,pos-1);
            split(y,y,z,tot);
            reverse(y);
            merge(x,y,y);
            merge(y,z,rt);
        }else if(op[0]=='G'){ //区间求和
            int pos=read(),tot=read(),x,y,z;
            split(rt,x,y,pos-1);
            split(y,y,z,tot);
            printf("%d\n",T[y].sum);
            merge(x,y,y);
            merge(y,z,rt);
        }else if(op[0]=='M'&&op[2]=='X'){ //求最大子序列和
            printf("%d\n",T[rt].maxzd);
        }
    }
    return 0;
}
```

luoguP2596 [ZJOI2006]书架

支持以下操作:

- 若 op为 `Top`，则后有一个整数 s，表示把编号为 s的书放在最上面。
- 若 op 为 `Bottom`，则后有一个整数 s，表示把编号为 s的书放在最下面。
- 若 op 为 `Insert`，则后有两个整数 s, t，表示若编号为 s的书上面有 x本书，则放回这本书时他的上面有 x + t本书。
- 若 op为 `Ask`，则后面有一个整数 s，表示询问编号为 s的书上面有几本书。
- 若 op 为 `Query`，则后面有一个整数 s，询问从上面起第 s本书的编号。

```
#include<bits/stdc++.h>
#define int long long
#define ls(x) (T[x].ch[0])
#define rs(x) (T[x].ch[1])
#define sz(x) (T[x].sz)
#define val(x) (T[x].val)
#define pri(x) (T[x].pri)
#define fa(x) (T[x].fa)
using namespace std;
const int N=1e5+7;
```

```
int read(){ int x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')
f=f*-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return
x*f;}

struct FHQ{
    int ch[2],val,pri,sz,fa;
}T[N];

int n,m,root=0,tot,id[N];

void update(int x){
    T[x].sz=sz(ls(x))+sz(rs(x))+1;
}

bool get(int x){
    return rs(fa(x))==x;
}

int cre(int v){ //新开辟一个结点
    sz(++tot)=1;
    val(tot)=v;
    pri(tot)=rand();
    id[v]=tot;
    return tot;
}

int merge(int x,int y){ // 合并操作
    if(!x||!y) return x+y; //x和y中必定有一个是0
    if(pri(x)<pri(y)){ //比较两棵树根的随机权值
        rs(x)=merge(rs(x),y);  //把x加到左边的树上
        fa(rs(x))=x;
        update(x); //更新大小
        return x;
    }else{
        ls(y)=merge(x,ls(y));
        fa(ls(y))=y;
        update(y);
        return y;
    }
}

void split(int x,int k,int &a,int &b,int faa=0,int fab=0){
    if(x==0){a=b=0;return;}
    if(k<=sz(ls(x)))  fa(x)=fab,b=x,split(ls(x),k,a,ls(x),faa,x);
    else fa(x)=faa,a=x,split(rs(x),k-sz(ls(x))-1,rs(x),b,x,fab);
    update(x);
}

int find(int k){
    int node=k,res=sz(ls(k))+1;
    while(node!=root&&k){
        if(get(k)) res+=sz(ls(fa(k)))+1;
        k=fa(k);
    }
    return res;
}

signed main(){
```

```
    srand(time(0));
    n=read();m=read();
    int x,y,z,w,s,t,k;
    string opt;
    for(int i=1;i<=n;i++){ //插入
        s=read();
        split(root,i-1,x,y);
        root=merge(merge(x,cre(s)),y);
    }
    for(int i=1;i<=m;i++){
        cin>>opt;s=read();
        if(opt=="Top"){ //将x放到最上面
            k=find(id[s]);
            split(root,k,x,z);
            split(x,k-1,x,y);
            root=merge(y,merge(x,z));
        }else if(opt=="Bottom"){ //将x放到最下面
            k=find(id[s]);
            split(root,k,x,z,0);
            split(x,k-1,x,y,0);
            root=merge(x,merge(z,y));
        }else if(opt=="Insert"){
            t=read();
            k=find(id[s]);
            if(t>0){ //与前驱交换
                split(root,k+1,z,w);
                split(z,k,y,z);
                split(y,k-1,x,y);
                root=(merge(x,merge(z,merge(y,w))));
            }
            if(t<0){ //与后继交换
                split(root,k,z,w);
                split(z,k-1,y,z);
                split(y,k-2,x,y);
                root=merge(x,merge(z,merge(y,w)));
            }
        }else if(opt=="Ask"){
            k=find(id[s]);
            printf("%lld\n",k-1);
        }else if(opt=="Query"){
            split(root,s,x,y);
            int node=x;
            while(rs(node)) node=rs(node);
            printf("%lld\n",val(node));
            root=merge(x,y);
        }
    }
    return 0;
}
```

P4130 [NOI2007] 项链工厂

（PS：听说这是线段树水题）给定序列和所有位置的颜色，支持以下操作：

- F：翻转①号位后的序列
- R k：将序列顺时针旋转k个位置
- S i j：交换i号位和j号位的颜色
- P i j x：从i出发顺时针到j的一段染成颜色x
- C：统计整个序列的颜色数
- CS i j：查询从i出发顺时针到j的一段的颜色数

```
#include<bits/stdc++.h>
#define sz(x) t[x].sz
#define ls(x) t[x].ch[0]
#define rs(x) t[x].ch[1]
#define rnd(x) t[x].rnd
#define val(x) t[x].val
#define cov(x) t[x].cov
#define pre(x) t[x].pre
#define suf(x) t[x].suf
#define sum(x) t[x].sum
#define tg(x) t[x].tg
using namespace std;
const int N=1e6+7;

int n,ad[N],q,c,rt,tot;
char op[20];

struct FHQ{
    int sz,ch[2],val,rnd,cov,pre,suf,sum;
    bool tg;
}t[N];

int newnode(int v){
    int x=++tot;
    sz(x)=1;rnd(x)=rand();sum(x)=1;
    pre(x)=suf(x)=val(x)=v;
    cov(x)=tg(x)=ls(x)=rs(x)=0;
    return x;
}

void reverse(int x){
    swap(ls(x),rs(x));
    swap(pre(x),suf(x));
    tg(x)^=1;
}

void update(int x){
    sz(x)=sz(ls(x))+sz(rs(x))+1;
    if(ls(x)) pre(x)=pre(ls(x));
    else pre(x)=val(x);
    if(rs(x)) suf(x)=suf(rs(x));
    else suf(x)=val(x);
    sum(x)=1;
    if(suf(ls(x))!=val(x)) sum(x)+=sum(ls(x));
    else sum(x)+=sum(ls(x))-1;
    if(pre(rs(x))!=val(x)) sum(x)+=sum(rs(x));
```

```
    else sum(x)+=sum(rs(x))-1;
    if(sz(x)==n&&pre(x)==suf(x)&&sum(x)!=1) sum(x)--;
}

void covered(int x,int d){
    cov(x)=val(x)=d;
    pre(x)=suf(x)=d;
    sum(x)=1;
}

void downdata(int x){
    if(tg(x)){
        if(ls(x)) reverse(ls(x));
        if(rs(x)) reverse(rs(x));
        tg(x)^=1;
    }
    if(cov(x)){
        if(ls(x)) covered(ls(x),cov(x));
        if(rs(x)) covered(rs(x),cov(x));
        cov(x)=0;
    }
}

void build(int &p,int l,int r){
    if(l>r) return;
    int mid=((l+r)>>1);
    p=newnode(ad[mid]);
    build(ls(p),l,mid-1);
    build(rs(p),mid+1,r);
    update(p);
}

int merge(int a,int b){
    if(!a||!b) return a+b;
    downdata(a);downdata(b);
    if(rnd(a)<rnd(b)){
        rs(a)=merge(rs(a),b);
        update(a);
        return a;
    }else{
        ls(b)=merge(a,ls(b));
        update(b);
        return b;
    }
}

void split(int now,int k,int &x,int &y){
    if(!now){
        x=y=0;
        return;
    }
    downdata(now);
    if(k<=sz(ls(now))) y=now,split(ls(now),k,x,ls(now));
    else x=now,split(rs(now),k-sz(ls(now))-1,rs(now),y);
    update(now);
}

signed main(){
```

```
    ios::sync_with_stdio(0);
    cin.tie(0);cout.tie(0);
    srand(time(0));
    cin>>n>>c;
    for(int i=1;i<=n;++i) cin>>ad[i];
    build(rt,1,n);
    cin>>q;
    for(int i=1;i<=q;++i){
        cin>>op;
        if(op[0]=='F'){  //1号位不动，翻转2~n号位
            int xa,ya;
            split(rt,1,xa,ya);
            reverse(ya);
            rt=merge(xa,ya);
        }else if(op[0]=='R'){  //所有珠子顺时针旋转k个位置
            int k,xa,ya;
            cin>>k;
            split(rt,n-k,xa,ya);
            rt=merge(ya,xa);
        }else if(op[0]=='S'){  //交换x和y号位的珠子
            int x,y,xa,xb,ya,yb,xc,yc,xd,yd;
            cin>>x>>y;
            if(x>y) swap(x,y);
            if(x==y) continue;
            split(rt,y,xa,ya);split(xa,y-1,xb,yb);
            split(xb,x,xc,yc);split(xc,x-1,xd,yd);
            rt=merge(merge(merge(xd,yb),yc),merge(yd,ya));
        }else if(op[0]=='P'){ //将[x,y]区间染成颜色z
            int x,y,z;
            cin>>x>>y>>z;
            if(x<=y){
                int xa,ya,xb,yb;
                split(rt,y,xa,ya);split(xa,x-1,xb,yb);
                covered(yb,z);
                rt=merge(merge(xb,yb),ya);
            }else{
                int xa,ya,xb,yb;
                split(rt,x-1,xa,ya);split(xa,y,xb,yb);
                covered(xb,z);covered(ya,z);
                rt=merge(merge(xb,yb),ya);
            }
        }else if(op[0]=='C'&&op[1]=='S'){ //查询[x,y]中有多少段颜色
            int x,y,xa,ya,xb,yb;
            cin>>x>>y;
            if(x<=y){
                if(x!=1||y!=n){
                    split(rt,y,xa,ya);
                    split(xa,x-1,xb,yb);
                    cout<<sum(yb)<<"\n";
                    rt=merge(merge(xb,yb),ya);
                }
            }else{
                split(rt,x-1,xa,ya);
                split(xa,y,xb,yb);
                if(suf(ya)!=pre(xb)) cout<<sum(ya)+sum(xb)<<"\n";
                else cout<<sum(ya)+sum(xb)-1<<"\n";
                rt=merge(merge(xb,yb),ya);
            }
```

```
        }else cout<<sum(rt)<<"\n"; //查询整个项链由多少部分组成
    }
    return 0;
}
```

参考资料

http://www.yhzq-blog.cc/fhqtreapzongjie/

https://www.luogu.com.cn/blog/acking/solution-FHQTreap

https://www.cnblogs.com/zwfymqz/p/7151959.html

https://www.luogu.com.cn/blog/zyxxs/solution-p2042