可持久化平衡树

算法简介

一种可以维护历史版本的平衡树，时间复杂度和空间复杂度均为 $O\left(m\log n\right)$

算法思想

使用类似可持久化线段树的方法继承历史版本，但大部分平衡树都自带旋转操作，节点的父子关系会发生，不利于可持久化。

于是考虑使用 fhq treap 进行可持久化。

fhq treap 的核心操作为 split 和 merge所以考虑 split 和 merge 时动态开点即可。

关于 merge 操作是否需要动态开点存在争议。事实上fhq treap 的 split 操作会提前为 merge 操作开点，但考虑下面代码

```
void erase(int &root,int p,int v){
    int lef,mid,rig;
    split(p,lef,rig,v);
    split(lef,lef,mid,v-1);
    merge(mid,node[mid].ch[0],node[mid].ch[1]);
    merge(lef,lef,mid);
    merge(root,lef,rig);
}
```

我们会发现 `merge(mid,node[mid].ch[0],node[mid].ch[1]);` 语句并没有 split 操作为其开点，这将导致错误。

解决方案为所有相同关键字的节点共用一个位置，或者为 merge 操作开点。考虑到前者编程较为复杂，这里选择后者。

算法模板

集合维护版本

洛谷p3835

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <string>
#include <sstream>
```

```
#include <cstring>
#include <cctype>
#include <cmath>
#include <vector>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <ctime>
#include <cassert>
#define _for(i,a,b) for(int i=(a);i<(b);++i)
#define _rep(i,a,b) for(int i=(a);i<=(b);++i)
#define mem(a,b) memset(a,b,sizeof(a))
using namespace std;
typedef long long LL;
inline int read_int(){
    int t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline LL read_LL(){
    LL t=0;bool sign=false;char c=getchar();
    while(!isdigit(c)){sign|=c=='-';c=getchar();}
    while(isdigit(c)){t=(t<<1)+(t<<3)+(c&15);c=getchar();}
    return sign?-t:t;
}
inline char get_char(){
    char c=getchar();
    while(c==' '||c=='\n'||c=='\r')c=getchar();
    return c;
}
inline void write(LL x){
    register char c[21],len=0;
    if(!x)return putchar('0'),void();
    if(x<0)x=-x,putchar('-');
    while(x)c[++len]=x%10,x/=10;
    while(len)putchar(c[len--]+48);
}
inline void space(LL x){write(x),putchar(' ');}
inline void enter(LL x){write(x),putchar('\n');}
const int MAXN=5e5+5,MAXM=40,Inf=0x7fffffff;
int root[MAXN];
struct fhq_treap{
    struct Node{
        int val,r,sz,ch[2];
    }node[MAXN*MAXM];
    int tot;
    int Copy(int k){
        node[++tot]=node[k];
```

```
        return tot;
    }
    int New(int v){
        int x=++tot;
node[x].val=v,node[x].r=rand(),node[x].sz=1,node[x].ch[0]=node[x].ch[1]=0;
        return x;
    }
    void pushup(int
k){node[k].sz=node[node[k].ch[0]].sz+node[node[k].ch[1]].sz+1;}
    void split(int k,int &k1,int &k2,int v){
        if(!k) return k1=k2=0,void();
        if(node[k].val<=v){
            k1=Copy(k);split(node[k].ch[1],node[k1].ch[1],k2,v);pushup(k1);
        }else{
            k2=Copy(k);split(node[k].ch[0],k1,node[k2].ch[0],v);pushup(k2);
        }
    }
    void merge(int &k,int k1,int k2){
        if(!k1||!k2)return k=k1|k2,void();
        if(node[k1].r>node[k2].r){
            k=Copy(k1);merge(node[k].ch[1],node[k1].ch[1],k2);pushup(k);
        }else{
            k=Copy(k2);merge(node[k].ch[0],k1,node[k2].ch[0]);pushup(k);
        }
    }
    void insert(int &root,int p,int v){
        int lef,rig;
        split(p,lef,rig,v);
        merge(lef,lef,New(v));
        merge(root,lef,rig);
    }
    void erase(int &root,int p,int v){
        int lef,mid,rig;
        split(p,lef,rig,v);
        split(lef,lef,mid,v-1);
        merge(mid,node[mid].ch[0],node[mid].ch[1]);
        merge(lef,lef,mid);
        merge(root,lef,rig);
    }
    int rank(int root,int v){
        int pos=root,ans=0;
        while(true){
            if(!pos)return ans+1;
if(node[pos].val<v)ans+=node[node[pos].ch[0]].sz+1,pos=node[pos].ch[1];
            else pos=node[pos].ch[0];
        }
    }
    int kth(int root,int rk){
        int pos=root;
        while(true){
            if(rk==node[node[pos].ch[0]].sz+1)return node[pos].val;
```

```
            else if(rk<node[node[pos].ch[0]].sz+1)pos=node[pos].ch[0];
            else rk-=node[node[pos].ch[0]].sz+1,pos=node[pos].ch[1];
        }
    }
    int pre(int root,int v){
        int pos=root,ans=-Inf;
        while(pos){
            if(node[pos].val>=v)pos=node[pos].ch[0];
            else ans=max(ans,node[pos].val),pos=node[pos].ch[1];
        }
        return ans;
    }
    int suf(int root,int v){
        int pos=root,ans=Inf;
        while(pos){
            if(node[pos].val<=v)pos=node[pos].ch[1];
            else ans=min(ans,node[pos].val),pos=node[pos].ch[0];
        }
        return ans;
    }
}tree;
int main()
{
    int n=read_int(),v,opt,x;
    _rep(i,1,n){
        v=read_int(),opt=read_int(),x=read_int();
        switch(opt){
            case 1:
                tree.insert(root[i],root[v],x);
                break;
            case 2:
                tree.erase(root[i],root[v],x);
                break;
            case 3:
                enter(tree.rank(root[i]=root[v],x));
                break;
            case 4:
                enter(tree.kth(root[i]=root[v],x));
                break;
            case 5:
                enter(tree.pre(root[i]=root[v],x));
                break;
            case 6:
                enter(tree.suf(root[i]=root[v],x));
                break;
        }
    }
    return 0;
}
```

序列维护版本

洛谷p5055