2020-2021 ACM-ICPC, Asia Seoul Regional Contest

比赛链接

I. Stock Analysis

题意

给定一个长度为 n 的序列，接下来 m 个询问，每次询问区间 $[L,R]$ 中的所有连续子串中和不超过 V 的最大值。

题解

离线询问，将询问的 V 和 $O(n^2)$ 个子串和从小到大排序后依次处理，于是题目变为支持单点最值操作和矩形最值查询的模板题。

考虑树套树，时间复杂度 $O\left(n^2\log^2 n+m\log^2 n\right)$但是二维线段树会变卡常，瓶颈是修改操作。

考虑用树状数组代替线段树，对于单点最值操作树状数组时间复杂度 $O(\log n)$区间最值查询操作时间复杂度 $O(\log^2 n)$

ps. 如果是单点修改操作和区间最值查询操作，那树状数组修改和查询时间复杂度均为 $O(\log^2 n)$一般就不如线段树了。

经检验，线段树套树状数组最快，时间复杂度 $O\left(n^2\log^2 n+m\log^3 n\right)$修改和查询操作复杂度比较平衡。

二维树状数组次之，时间复杂度 $O\left(n^2\log^2 n+m\log^4 n\right)$主要原因是虽然修改操作常数更小了但查询操作太慢了。

```
const int MAXN=2005,MAXM=2e5+5,MAXQ=MAXN*MAXN/2+MAXM;
int n,lef[MAXN<<2],rig[MAXN<<2],s[MAXN<<2][MAXN<<2];
void pushup1D(int rt,int k){
    s[rt][k]=max(s[rt][k<<1],s[rt][k<<1|1]);
}
void pushup2D(int rt,int k){
    s[rt][k]=max(s[rt<<1][k],s[rt<<1|1][k]);
}
void build2D(int k,int L,int R){
    lef[k]=L,rig[k]=R;
    if(L==R)
    return;
    int M=L+R>>1;
```

```
    build2D(k<<1,L,M);
    build2D(k<<1|1,M+1,R);
}
void update1D(int rt,int k,int cl,int cr,int pos,int v){
    if(cl==cr){
        if(lef[rt]==rig[rt])
        s[rt][k]=v;
        else
        pushup2D(rt,k);
        return;
    }
    int cm=cl+cr>>1;
    if(pos<=cm)
    update1D(rt,k<<1,cl,cm,pos,v);
    else
    update1D(rt,k<<1|1,cm+1,cr,pos,v);
    pushup1D(rt,k);
}
void update2D(int k,int x,int y,int v){
    if(lef[k]==rig[k])
    return update1D(k,1,1,n,y,v);
    int mid=lef[k]+rig[k]>>1;
    if(x<=mid)
    update2D(k<<1,x,y,v);
    else
    update2D(k<<1|1,x,y,v);
    update1D(k,1,1,n,y,v);
}
int query1D(int rt,int k,int cl,int cr,int ql,int qr){
    if(ql<=cl&&cr<=qr)
    return s[rt][k];
    int cm=cl+cr>>1;
    if(qr<=cm)
    return query1D(rt,k<<1,cl,cm,ql,qr);
    else if(ql>cm)
    return query1D(rt,k<<1|1,cm+1,cr,ql,qr);
    else
    return
max(query1D(rt,k<<1,cl,cm,ql,qr),query1D(rt,k<<1|1,cm+1,cr,ql,qr));
}
int query2D(int k,int lx,int rx,int ly,int ry){
    if(lx<=lef[k]&&rig[k]<=rx)
    return query1D(k,1,1,n,ly,ry);
    int mid=lef[k]+rig[k]>>1;
    if(rx<=mid)
    return query2D(k<<1,lx,rx,ly,ry);
    else if(lx>mid)
    return query2D(k<<1|1,lx,rx,ly,ry);
    else
    return max(query2D(k<<1,lx,rx,ly,ry),query2D(k<<1|1,lx,rx,ly,ry));
```

```
}
LL pre[MAXN],ans[MAXM],ss[MAXQ];
struct Node{
    LL val;
    int lef,rig,idx;
    bool operator < (const Node &b)const{
        return val<b.val||(val==b.val&&idx<b.idx);
    }
}node[MAXQ];
int main()
{
    n=read_int();
    int m=read_int(),q=0,mv=0;
    _rep(i,1,n)pre[i]=read_int();
    _rep(i,1,n)pre[i]+=pre[i-1];
    _rep(i,1,m){
        int ql=read_int(),qr=read_int();
        LL qv=read_LL();
        node[q++]=Node{qv,ql,qr,i};
    }
    _rep(i,1,n)_for(j,0,i){
        node[q++]=Node{pre[i]-pre[j],j+1,i,0};
        ss[++mv]=pre[i]-pre[j];
    }
    sort(node,node+q);
    sort(ss+1,ss+mv+1);
    mv=unique(ss+1,ss+mv+1)-ss;
    build2D(1,1,n);
    _for(i,0,q){
        if(node[i].idx==0)
update2D(1,node[i].lef,node[i].rig,lower_bound(ss+1,ss+mv,node[i].val)-ss);
        else
ans[node[i].idx]=query2D(1,node[i].lef,node[i].rig,node[i].lef,node[i].rig)
;
    }
    _rep(i,1,m){
        if(ans[i]==0)
        puts("NONE");
        else
        enter(ss[ans[i]]);
    }
    return 0;
}
```

```
#define lowbit(x) ((x)&(-x))
const int MAXN=2005,MAXM=2e5+5,MAXQ=MAXN*MAXN/2+MAXM;
int n,lef[MAXN<<2],rig[MAXN<<2],a[MAXN<<2][MAXN],s[MAXN<<2][MAXN];
void build2D(int k,int L,int R){
    lef[k]=L,rig[k]=R;
    if(L==R)
```

```
    return;
    int M=L+R>>1;
    build2D(k<<1,L,M);
    build2D(k<<1|1,M+1,R);
}
void update1D(int rt,int pos,int v){
    a[rt][pos]=max(a[rt][pos],v);
    while(pos<=n){
        s[rt][pos]=max(s[rt][pos],v);
        pos+=lowbit(pos);
    }
}
void update2D(int k,int x,int y,int v){
    if(lef[k]==rig[k])
    return update1D(k,y,v);
    int mid=lef[k]+rig[k]>>1;
    if(x<=mid)
    update2D(k<<1,x,y,v);
    else
    update2D(k<<1|1,x,y,v);
    update1D(k,y,v);
}
int query1D(int rt,int l,int r){
    int ans=0;
    while(l<=r){
        ans=max(ans,a[rt][r]);
        for(--r;r-l>=lowbit(r);r-=lowbit(r))
        ans=max(ans,s[rt][r]);
    }
    return ans;
}
int query2D(int k,int lx,int rx,int ly,int ry){
    if(lx<=lef[k]&&rig[k]<=rx)
    return query1D(k,ly,ry);
    int mid=lef[k]+rig[k]>>1;
    if(rx<=mid)
    return query2D(k<<1,lx,rx,ly,ry);
    else if(lx>mid)
    return query2D(k<<1|1,lx,rx,ly,ry);
    else
    return max(query2D(k<<1,lx,rx,ly,ry),query2D(k<<1|1,lx,rx,ly,ry));
}
LL pre[MAXN],ans[MAXM],ss[MAXQ];
struct Node{
    LL val;
    int lef,rig,idx;
    bool operator < (const Node &b)const{
        return val<b.val||(val==b.val&&idx<b.idx);
    }
}node[MAXQ];
```

```
int main()
{
    n=read_int();
    int m=read_int(),q=0,mv=0;
    _rep(i,1,n)pre[i]=read_int();
    _rep(i,1,n)pre[i]+=pre[i-1];
    _rep(i,1,m){
        int ql=read_int(),qr=read_int();
        LL qv=read_LL();
        node[q++]=Node{qv,ql,qr,i};
    }
    _rep(i,1,n)_for(j,0,i){
        node[q++]=Node{pre[i]-pre[j],j+1,i,0};
        ss[++mv]=pre[i]-pre[j];
    }
    sort(node,node+q);
    sort(ss+1,ss+mv+1);
    mv=unique(ss+1,ss+mv+1)-ss;
    build2D(1,1,n);
    _for(i,0,q){
        if(node[i].idx==0)
update2D(1,node[i].lef,node[i].rig,lower_bound(ss+1,ss+mv,node[i].val)-ss);
        else
ans[node[i].idx]=query2D(1,node[i].lef,node[i].rig,node[i].lef,node[i].rig)
;
    }
    _rep(i,1,m){
        if(ans[i]==0)
        puts("NONE");
        else
        enter(ss[ans[i]]);
    }
    return 0;
}
```

```
#define lowbit(x) ((x)&(-x))
const int MAXN=2005,MAXM=2e5+5,MAXQ=MAXN*MAXN/2+MAXM;
int n;
struct Tree{
    int a[MAXN],s[MAXN];
    void update(int pos,int v){
        a[pos]=max(a[pos],v);
        while(pos<=n){
            s[pos]=max(s[pos],v);
            pos+=lowbit(pos);
        }
    }
    int query(int l,int r){
        int ans=0;
        while(l<=r){
```

```
            ans=max(ans,a[r]);
            for(--r;r-l>=lowbit(r);r-=lowbit(r))
            ans=max(ans,s[r]);
        }
        return ans;
    }
}tree1[MAXN],tree2[MAXN];
void update2D(int x,int y,int v){
    tree1[x].update(y,v);
    while(x<=n){
        tree2[x].update(y,v);
        x+=lowbit(x);
    }
}
int query2D(int lx,int rx,int ly,int ry){
    int ans=0;
    while(lx<=rx){
        ans=max(ans,tree1[rx].query(ly,ry));
        for(--rx;rx-lx>=lowbit(rx);rx-=lowbit(rx))
        ans=max(ans,tree2[rx].query(ly,ry));
    }
    return ans;
}
LL pre[MAXN],ans[MAXM],ss[MAXQ];
struct Node{
    LL val;
    int lef,rig,idx;
    bool operator < (const Node &b)const{
        return val<b.val||(val==b.val&&idx<b.idx);
    }
}node[MAXQ];
int main()
{
    n=read_int();
    int m=read_int(),q=0,mv=0;
    _rep(i,1,n)pre[i]=read_int();
    _rep(i,1,n)pre[i]+=pre[i-1];
    _rep(i,1,m){
        int ql=read_int(),qr=read_int();
        LL qv=read_LL();
        node[q++]=Node{qv,ql,qr,i};
    }
    _rep(i,1,n)_for(j,0,i){
        node[q++]=Node{pre[i]-pre[j],j+1,i,0};
        ss[++mv]=pre[i]-pre[j];
    }
    sort(node,node+q);
    sort(ss+1,ss+mv+1);
    mv=unique(ss+1,ss+mv+1)-ss;
    _for(i,0,q){
```

```
        if(node[i].idx==0)
update2D(node[i].lef,node[i].rig,lower_bound(ss+1,ss+mv,node[i].val)-ss);
        else
ans[node[i].idx]=query2D(node[i].lef,node[i].rig,node[i].lef,node[i].rig);
    }
    _rep(i,1,m){
        if(ans[i]==0)
        puts("NONE");
        else
        enter(ss[ans[i]]);
    }
    return 0;
}
```