

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 $O(n^2 \log^2 n + m \log^2 n)$ 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，树状数组时间复杂度变为 $O(\log^2 n)$

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

另外二维树状数组次之，时间复杂度 $O(n^2 \log^2 n + m \log^4 n)$ 修改常数更小了但查询变慢了。

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <algorithm>
#include <string>
#include <sstream>
#include <cstring>
#include <cctype>
#include <cmath>
#include <vector>
#include <set>
#include <bitset>
#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){
    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=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;
}
```

From:
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:jxm2001:contest:2021_buaa_spring_training4&rev=1619427683

Last update: 2021/04/26 17:01