[报告]ZOJ Doraemon's Sweet Bullet

[Source]

http://acm.zju.edu.cn/onlinejudge/showProblem.do?problemCode=3453

[Description]

有 N 个敌人，排成一排编号为 1~n；对于敌人 i 有一个初始血值 value[i]；对于敌人 i，其朋友范围区间[Li,Ri]，i 可能在[Li,Ri]区间内。你每次从右边发射子弹，第 i 颗子弹值为 Ki，打中第一个 value 值大于或等于 Ki 的敌人，该敌人 value 值变为 1，其朋友范围内的敌人 value 值均增加 1；但是，如果没有敌人的 value 值大于或者等于 Ki，则所有敌人 value 值增加 1。求最后敌人中最高的 value 值。

[Solution]

点查找与区间覆盖，典型的线段树。对于这个问题，维护两个域即可：cnt，该区间内最大的血值；a，该区间的子区间需增加的血值，相当于懒惰标记。对于每次操作，如果存在符合的敌人，则由根节点优先向右查找敌人，即满足情况的最右的敌人，然后根据他的朋友范围区间覆盖。

[Code]

```
#include<iostream>
#include<cstdio>
#include<string.h>
#include<string>
#include<math.h>
#include<algorithm>
using namespace std;
#define maxn 100009

struct tree{
    int l,r;
    int cnt;
    int a;
```

```
};

tree t[4*maxn];
int a[maxn],b[maxn],c[maxn];
int m,n,pos,val;

void build(int p,int l,int r){
    t[p].l=l;
    t[p].r=r;
    t[p].a=0;
    t[p].cnt=a[l];
    if(l==r) return ;
    int mid=(l+r)>>1;
    build(p<<1,l,mid);
    build(p<<1|1,mid+1,r);
    t[p].cnt=max(t[p<<1].cnt,t[p<<1|1].cnt);
}

int query(int p){
    if(t[p].l==t[p].r)
        return t[p].cnt;
    if(t[p].a){
        t[p<<1].a+=t[p].a;
        t[p<<1|1].a+=t[p].a;
        t[p<<1].cnt+=t[p].a;
        t[p<<1|1].cnt+=t[p].a;
        t[p].a=0;
    }
    return max(query(p<<1),query(p<<1|1));
}

void update(int p,int l,int r,int a){
    if(t[p].l==l&&t[p].r==r){
        t[p].cnt+=a;
        t[p].a+=a;
        return ;
    }
    if(t[p].a){
        t[p<<1].a+=t[p].a;
        t[p<<1|1].a+=t[p].a;
        t[p<<1].cnt+=t[p].a;
        t[p<<1|1].cnt+=t[p].a;
        t[p].a=0;
    }
```

```
    int mid=(t[p].l+t[p].r)>>1;
    if(r<=mid) update(p<<1,l,r,a);
    else if(l>mid) update(p<<1|1,l,r,a);
    else{
        update(p<<1,l,mid,a);
        update(p<<1|1,mid+1,r,a);
    }
    t[p].cnt=max(t[p<<1].cnt,t[p<<1|1].cnt);
}

void search(int p,int val){
      if (t[p].l==t[p].r){
            t[p].cnt=1;
            t[p].a=0;
            pos=t[p].l;
            return ;
      }
      if (t[p].a){
            t[p<<1].a+=t[p].a;
            t[p<<1|1].a+=t[p].a;
            t[p<<1].cnt+=t[p].a;
            t[p<<1|1].cnt+=t[p].a;
            t[p].a=0;
      }
      int mid=(t[p].l+t[p].r)>>1;
      if (val<=t[p<<1|1].cnt)
            search(p<<1|1,val);
      else
            search(p<<1,val);
      t[p].cnt=max(t[p<<1].cnt,t[p<<1|1].cnt);
}

int main(){
    while(scanf("%d",&n)==1){
        pos=1;
        for(int i=1;i<=n;i++)
            scanf("%d %d %d",&a[i],&b[i],&c[i]);
        build(1,1,n);
        scanf("%d",&m);
        for(int i=0;i<m;i++){
            scanf("%d",&val);
            if(t[1].cnt<val){
                update(1,1,n,1);
                continue;
```

```
            }
            search(1,val);
            update(1,b[pos],c[pos],1);
        }
        printf("%d\n",query(1));
    }
    return 0;
}
```