排序

一. 考察内容：

线段树 权值线段树 二分答案

二. 题目分析：

[题目大意]

给出一个长度为n的序列，和m个排序操作，每个排序操作类似:op l r，当op=0时，将l~r之间的元素升序排序，当op=1时，将l~r的元素降序排序，求出最后第k为的元素是几。

[写题思路]

首先我们二分一个答案，设我们二分的答案为第k为的元素是ans，然后我们将所有小于等于ans的节点全部设为0，大于等于ans的节点全部设为1，这时我们再用权值线段树维护每个区间0的个数，这样我们可以在logn内对一个区间的0和1进行升序或者降序排序，最后判断一下第k位是不是0，如果是0，则缩小答案，如果不是0，则扩大答案即可。

另外该题有一个线段树合并的写法，复杂度也是nlog^2n，但是我并没有想到怎么做。

三. 代码实现：

```
#define _CRT_SECURE_NO_DEPRECATE
/*************************
*创建时间：2018 08 30
*文件类型：源代码文件
*题目来源：COGS
*当前状态：已通过
*备忘录：线段树 二分答案
*作者：HtBest
*************************/
#include <stdio.h>
#include <iostream>
#include <string>
#include <stdlib.h>
#include <string.h>
#include <algorithm>
#include <queue>
// #include <sys/wait.h>
// #include <sys/types.h>
// #include <unistd.h>
using namespace std;
#define MAXN 100001
int n,m,a[MAXN],mid,q;
struct OP
{
    int op,l,r;
}op[MAXN];
struct NODE
{
    int lson,rson,v,lazy,len;//v:区间中-1个数
    void reset(){lson=rson=v=lazy=len=0;}
    void merge(NODE &a,NODE &b)
    {
        v=a.v+b.v;
        len=a.len+b.len;
    }
    void addl(int &x)
    {
        lazy=x;
        if(lazy==-1)v=len;//区间赋-1
```

```
        else v=0;//区间赋1
    }
}o[2*MAXN];
struct segmentTree
{
    int cnt,root;
    void downl(NODE &x)
    {
        if(x.lazy)
        {
            o[x.lson].addl(x.lazy);
            o[x.rson].addl(x.lazy);
        }
        x.lazy=0;
    }
    void build(int &x,int l,int r,int v)
    {
        x=++cnt;
        o[x].reset();
        if(l==r)o[x].v=(a[l]>v?0:1),o[x].len=1;
        else
        {
            int m=(l+r)>>1;
            build(o[x].lson,l,m,v);
            build(o[x].rson,m+1,r,v);
            o[x].merge(o[o[x].lson],o[o[x].rson]);
        }
    }
    void update(int x,int l,int r,int ul,int ur,int v)
    {
        if(l>=ul&&r<=ur)o[x].addl(v);
        else
        {
            int m=(l+r)>>1;
            downl(o[x]);
            if(ul<=m)update(o[x].lson,l,m,ul,ur,v);
            if(ur>m)update(o[x].rson,m+1,r,ul,ur,v);
            o[x].merge(o[o[x].lson],o[o[x].rson]);
        }
    }
    void query(int x,int l,int r,int ql,int qr,NODE &q)
    {
        if(l>=ql&&r<=qr)q=o[x];
        else
        {
            int m=(l+r)>>1;
            downl(o[x]);
            if(qr<=m)query(o[x].lson,l,m,ql,qr,q);
            else if(ql>m)query(o[x].rson,m+1,r,ql,qr,q);
            else
            {
                NODE a,b;
                query(o[x].lson,l,m,ql,qr,a);
                query(o[x].rson,m+1,r,ql,qr,b);
                q.merge(a,b);
            }
        }
    }
    void build(int x){cnt=0,build(root,1,n,x);}
    void update(int l,int r,int v){if(l<=r)update(root,1,n,l,r,v);}
    int query(int l,int r)
    {
        NODE ans;
        query(root,1,n,l,r,ans);
        return ans.v;
    }
}seg;
/* Variable explain:

*/
void read()
{
```

```
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;++i)scanf("%d",&a[i]);
    for(int i=1;i<=m;++i)scanf("%d%d%d",&op[i].op,&op[i].l,&op[i].r);
    scanf("%d",&q);
    return;
}
bool check(int x)
{
    seg.build(x);
    for(int i=1;i<=m;++i)
    {
        int v=seg.query(op[i].l,op[i].r);
        // printf("%d %d\n",v,op[i].op);
        // for(int j=1;j<=n;++j)
        // {
        //  printf("%d ",seg.query(j,j));
        // }
        // puts("");
        if(op[i].op==0)//升序
        {
            seg.update(op[i].l,op[i].l+v-1,-1);
            seg.update(op[i].l+v,op[i].r,1);
        }
        else //降序
        {
            seg.update(op[i].l,op[i].r-v,1);
            seg.update(op[i].r-v+1,op[i].r,-1);
        }
    }
    return seg.query(q,q);
}
void solve()
{
    int l=0,r=n,ans;
    while(l<=r)
    {
        int m=(l+r)>>1;
        // printf("#%d#\n",m);
        if(check(m))ans=m,r=m-1;
        else l=m+1;
    }
    printf("%d\n",ans);
}
int main()
{
    // freopen(".in","r",stdin);
    // freopen(".out","w",stdout);
    read();
    // printf("%d\n",check(3));
    solve();
    return 0;
}
```

<题目跳转> <查看代码>