树套树练习

qq_42371755 于 2019-12-06 16:14:06 发布

树套树讲题

T1：二逼平衡树（树套树）

传送门

题解

所谓树套树呢？就是把两棵不同的树套在一起，以达到意想不到的效果。
对于这道题，可以线段树套平衡树，也就是每个点维护一个平衡树，下面我来分别讲解一下每种操作。
查询k在区间内的排名

把区间分解成线段树上的点，查询k在每个点上的排名-1然后累加，最后在加上1。
查询区间内排名为k的值
先二分一个值，然后用拿到这个值的最大排名（我没有把相同的点合并为一个点），然后找到满最大排名大于等于k 的最小的mid，就是答案。
修改某一位值上的数值
暴力在线段树上单点修改，删去原来的数字后加入新给的数字。
查询k在区间内的前驱
先分解区间，然后取每个点找到的前驱中最大的。
查询k在区间内的后继
与上面的操作方式类似。
总时间复杂度O($\mathrm{nlog^3n}$)

代码

```
#include <cstdio>
#include <iostream>
#include <cstdlib>
#include <ctime>
#define ls son[num][0]
#define rs son[num][1]
using namespace std;
const int maxn = 5e4+5;
const int INF = 2147483647;
int readint(){
    int x=0,f=1;char s=getchar();
    #define sc (s=getchar())
    while(s<'0'||s>'9'){
        if(s=='-')
            f=-1;
        sc;
    }
```

```
    while(s>='0'&&s<='9'){
        x=(x<<3)+(x<<1)+(s^48);
        sc;
    }
    #undef sc
    return x*f;
}
int len,n,q;
int a[maxn<<5];
int siz[maxn<<5],key[maxn<<5],son[maxn<<5][2],cnt[maxn<<5],rd[maxn<<5];
struct treap{
    int rt;
    void push_up(int num){
        siz[num]=siz[ls]+siz[rs]+cnt[num];
    }
    void rotate(int &x,int d){
        int root=son[x][d^1];
        son[x][d^1]=son[root][d];
        son[root][d]=x;
        push_up(x);
        push_up(root);
        x=root;
    }
    void insert(int &num,int x){
        if(!num){
            num=++len;
            siz[num]=cnt[num]=1;
            key[num]=x;
            rd[num]=rand();
            return;
        }
        if(key[num]==x){
```

```
            cnt[num]++;
            siz[num]++;
            return;
        }
        int d=(x>key[num]);
        insert(son[num][d],x);
        if(rd[num]<rd[son[num][d]])
            rotate(num,d^1);
        push_up(num);
    }
    void deleted(int &num,int x){
        if(!num)
            return;
        if(x!=key[num])
            deleted(son[num][x>key[num]],x);
        else{
            if(!ls&&!rs){
                cnt[num]--;
                siz[num]--;
                if(cnt[num]==0)
                    num=0;
            }
            else if(ls&&!rs){
                rotate(num,1);
                deleted(rs,x);
            }
            else if(!ls&&rs){
                rotate(num,0);
                deleted(ls,x);
            }
            else{
                int d=rd[ls]>rd[rs];
```

```
                rotate(num,d);
                deleted(son[num][d],x);
            }
        }
        push_up(num);
    }
    int get_rank(int num,int x){
        if(!num)
            return 0;
        if(key[num]==x)
            return siz[ls];
        if(key[num]<x)
            return siz[ls]+cnt[num]+get_rank(rs,x);
        return get_rank(ls,x);
    }
    int find(int num,int x){
        if(!num)
            return 0;
        if(siz[ls]>=x)
            return find(ls,x);
        else if(siz[ls]+cnt[num]<x)
            return find(rs,x-cnt[num]-siz[ls]);
        else
            return key[num];
    }
    int getpre(int num,int x){
        if(!num)
            return -INF;
        if(key[num]>=x)
            return getpre(ls,x);
        else
            return max(key[num],getpre(rs,x));
```

```
    }
    int getback(int num,int x){
        if(!num)
            return INF;
        if(key[num]<=x)
            return getback(rs,x);
        else
            return min(key[num],getback(ls,x));
    }
    int com(int &num,int x){
        int pre=getpre(num,x),nxt=getback(num,x);
        int f1=x-pre,f2=nxt-x;
        if(f1>f2){
            deleted(num,nxt);
            return f2;
        }
        else{
            deleted(num,pre);
            return f1;
        }
    }
}tree[maxn<<3];
namespace segmentTree{
    void build(int num,int l,int r){
        for(int i=l;i<=r;i++)
            tree[num].insert(tree[num].rt,a[i]);
        if(l==r)
            return;
        int mid=(l+r)>>1;
        build(num<<1,l,mid);
        build(num<<1|1,mid+1,r);
    }
```

```
146     void update(int num,int l,int r,int pos,int val){
147         if(pos<l||r<pos)
148             return;
149         tree[num].deleted(tree[num].rt,a[pos]);
150         tree[num].insert(tree[num].rt,val);
151         if(l==r)
152             return;
153         int mid=(l+r)>>1;
154         update(num<<1,l,mid,pos,val);
155         update(num<<1|1,mid+1,r,pos,val);
156     }
157     // zhen de rank = queryrank() + 1
158     int queryrank(int num,int l,int r,int goal_l,int goal_r,int val){
159         if(goal_l>r||goal_r<l)
160             return 0;
161         if(l>=goal_l&&r<=goal_r)
162             return tree[num].get_rank(tree[num].rt,val);
163         int mid=(l+r)>>1;
164         return queryrank(num<<1,l,mid,goal_l,goal_r,val)+queryrank(num<<1|1,mid+1,r,goal_l,goal_r,val);
165     }
166     int queryfind(int goal_l,int goal_r,int rank){
167         int l=0,r=1e8,ans;
168         while(l<=r){
169             int mid=(l+r)>>1;
170             if(queryrank(1,1,n,goal_l,goal_r,mid)+1<=rank)
171                 ans=mid,l=mid+1;
172             else
173                 r=mid-1;
174         }
175         return ans;
176     }
177     int querypre(int num,int l,int r,int goal_l,int goal_r,int val){
```

```
178         if(goal_r<l||goal_l>r)
179             return -INF;
180         if(goal_l<=l&&r<=goal_r)
181             return tree[num].getpre(tree[num].rt,val);
182         int mid=(l+r)>>1;
183         return max(querypre(num<<1,l,mid,goal_l,goal_r,val),querypre(num<<1|1,mid+1,r,goal_l,goal_r,val));
184     }
185     int queryback(int num,int l,int r,int goal_l,int goal_r,int val){
186         if(goal_r<l||goal_l>r)
187             return INF;
188         if(goal_l<=l&&r<=goal_r)
189             return tree[num].getback(tree[num].rt,val);
190         int mid=(l+r)>>1;
191         return min(queryback(num<<1,l,mid,goal_l,goal_r,val),queryback(num<<1|1,mid+1,r,goal_l,goal_r,val));
192     }
193 }
194 int main (){
195     srand(20050301);
196     n=readint(),q=readint();
197     for(int i=1;i<=n;i++)
198         a[i]=readint();
199     segmentTree::build(1,1,n);
200     while(q--){
201         int op=readint();
202         if(op==3){
203             int pos=readint(),val=readint();
204             segmentTree::update(1,1,n,pos,val);
205             a[pos]=val;
206         }
207         else{
208             int l=readint(),r=readint(),k=readint();
209             if(op==1)
```

```
                printf("%d\n",segmentTree::queryrank(1,1,n,l,r,k)+1);
            if(op==2)
                printf("%d\n",segmentTree::queryfind(l,r,k));
            if(op==4)
                printf("%d\n",segmentTree::querypre(1,1,n,l,r,k));
            if(op==5)
                printf("%d\n",segmentTree::queryback(1,1,n,l,r,k));
        }
    }
    return 0;
}
```

[国家集训队]排队

传送门

题解

还需要考虑a_l,a_r之间的大小关系对全局逆序对的贡献
若$a_l < a_r$
那么交换后会产生 1的贡献
若$a_l > a_r$
那么交换后会产生-1-1的贡献
所以我们需要这样一种数据结构，可以支持：
单点插入
单点删除
区间询问严格的比val小的元素有多少个
区间询问严格的比val大的元素有多少个
我们可以使用：分块或者树套树
我这里使用的是树套树(即线段树套Treap)

代码

```
#include <cstdio>
#include <iostream>
#include <cstdlib>
#include <ctime>
#define ls son[num][0]
#define rs son[num][1]
using namespace std;
const int maxn = 5e4+5;
const int INF = 2147483647;
int readint(){
    int x=0,f=1;char s=getchar();
    #define sc (s=getchar())
    while(s<'0'||s>'9'){
        if(s=='-')
            f=-1;
        sc;
    }
    while(s>='0'&&s<='9'){
        x=(x<<3)+(x<<1)+(s^48);
        sc;
    }
    #undef sc
    return x*f;
}
int len,n,q;
int a[maxn<<5];
int siz[maxn<<5],key[maxn<<5],son[maxn<<5][2],cnt[maxn<<5],rd[maxn<<5];
struct treap{
    int rt;
    void push_up(int num){
```

```
        siz[num]=siz[ls]+siz[rs]+cnt[num];
    }
    void rotate(int &x,int d){
        int root=son[x][d^1];
        son[x][d^1]=son[root][d];
        son[root][d]=x;
        push_up(x);
        push_up(root);
        x=root;
    }
    void insert(int &num,int x){
        if(!num){
            num=++len;
            siz[num]=cnt[num]=1;
            key[num]=x;
            rd[num]=rand();
            return;
        }
        if(key[num]==x){
            cnt[num]++;
            siz[num]++;
            return;
        }
        int d=(x>key[num]);
        insert(son[num][d],x);
        if(rd[num]<rd[son[num][d]])
            rotate(num,d^1);
        push_up(num);
    }
    void deleted(int &num,int x){
        if(!num)
            return;
```

```
63          if(x!=key[num])
64              deleted(son[num][x>key[num]],x);
65          else{
66              if(!ls&&!rs){
67                  cnt[num]--;
68                  siz[num]--;
69                  if(cnt[num]==0)
70                      num=0;
71              }
72              else if(ls&&!rs){
73                  rotate(num,1);
74                  deleted(rs,x);
75              }
76              else if(!ls&&rs){
77                  rotate(num,0);
78                  deleted(ls,x);
79              }
80              else{
81                  int d=rd[ls]>rd[rs];
82                  rotate(num,d);
83                  deleted(son[num][d],x);
84              }
85          }
86          push_up(num);
87      }
88      int get_rank(int num,int x){
89          if(!num)
90              return 0;
91          if(key[num]==x)
92              return siz[ls];
93          if(key[num]<x)
94              return siz[ls]+cnt[num]+get_rank(rs,x);
```

```
95          return get_rank(ls,x);
96      }
97      int get_Rank(int num,int x){
98          if(!num)
99              return 0;
100         if(key[num]==x)
101             return siz[rs];
102         if(key[num]>x)
103             return siz[rs]+cnt[num]+get_Rank(ls,x);
104         return get_Rank(rs,x);
105     }
106     int find(int num,int x){
107         if(!num)
108             return 0;
109         if(siz[ls]>=x)
110             return find(ls,x);
111         else if(siz[ls]+cnt[num]<x)
112             return find(rs,x-cnt[num]-siz[ls]);
113         else
114             return key[num];
115     }
116     int getpre(int num,int x){
117         if(!num)
118             return -INF;
119         if(key[num]>=x)
120             return getpre(ls,x);
121         else
122             return max(key[num],getpre(rs,x));
123     }
124     int getback(int num,int x){
125         if(!num)
126             return INF;
```

```
        if(key[num]<=x)
            return getback(rs,x);
        else
            return min(key[num],getback(ls,x));
    }
    int com(int &num,int x){
        int pre=getpre(num,x),nxt=getback(num,x);
        int f1=x-pre,f2=nxt-x;
        if(f1>f2){
            deleted(num,nxt);
            return f2;
        }
        else{
            deleted(num,pre);
            return f1;
        }
    }
}tree[maxn<<3];
namespace segmentTree{
    void build(int num,int l,int r){
        for(int i=l;i<=r;i++)
            tree[num].insert(tree[num].rt,a[i]);
        if(l==r)
            return;
        int mid=(l+r)>>1;
        build(num<<1,l,mid);
        build(num<<1|1,mid+1,r);
    }
    void update(int num,int l,int r,int pos,int val){
        if(pos<l||r<pos)
            return;
        tree[num].deleted(tree[num].rt,a[pos]);
```

```
        tree[num].insert(tree[num].rt,val);
        if(l==r)
            return;
        int mid=(l+r)>>1;
        update(num<<1,l,mid,pos,val);
        update(num<<1|1,mid+1,r,pos,val);
    }
    // zhen de rank = queryrank() + 1
    int queryrank(int num,int l,int r,int goal_l,int goal_r,int val){
        if(goal_l>r||goal_r<l)
            return 0;
        if(l>=goal_l&&r<=goal_r)
            return tree[num].get_rank(tree[num].rt,val);
        int mid=(l+r)>>1;
        return queryrank(num<<1,l,mid,goal_l,goal_r,val)+queryrank(num<<1|1,mid+1,r,goal_l,goal_r,val);
    }
    int queryRank(int num,int l,int r,int goal_l,int goal_r,int val){
        if(goal_l>r||goal_r<l)
            return 0;
        if(l>=goal_l&&r<=goal_r)
            return tree[num].get_Rank(tree[num].rt,val);
        int mid=(l+r)>>1;
        return queryRank(num<<1,l,mid,goal_l,goal_r,val)+queryRank(num<<1|1,mid+1,r,goal_l,goal_r,val);
    }
    int queryfind(int goal_l,int goal_r,int rank){
        int l=0,r=1e8,ans;
        while(l<=r){
            int mid=(l+r)>>1;
            if(queryrank(1,1,n,goal_l,goal_r,mid)+1<=rank)
                ans=mid,l=mid+1;
            else
                r=mid-1;
```

```
        }
        return ans;
    }
}
int main (){
    //freopen("own.out","w",stdout);
    //freopen("testdata.in","r",stdin);
    //freopen("own.out","w",stdout);
    srand(20050301);
    int n=readint();
    for(int i=1;i<=n;i++)
        a[i]=readint();
    segmentTree::build(1,1,n);
    int ans=0;
    for(int i=1;i<=n;i++)
        ans+=segmentTree::queryRank(1,1,n,1,i-1,a[i]);
    cout<<ans<<endl;
    int q=readint();
    while(q--){
        int x=readint(),y=readint();
        int t1=a[x],t2=a[y];
        ans-=segmentTree::queryRank(1,1,n,1,x-1,a[x]);
        ans-=segmentTree::queryrank(1,1,n,x+1,n,a[x]);
        ans+=segmentTree::queryRank(1,1,n,1,x-1,a[y]);
        ans+=segmentTree::queryrank(1,1,n,x+1,n,a[y]);
        segmentTree::update(1,1,n,x,t2);
        ans-=segmentTree::queryRank(1,1,n,1,y-1,a[y]);
        ans-=segmentTree::queryrank(1,1,n,y+1,n,a[y]);
        ans+=segmentTree::queryRank(1,1,n,1,y-1,a[x]);
        ans+=segmentTree::queryrank(1,1,n,y+1,n,a[x]);
        segmentTree::update(1,1,n,y,t1);
        swap(a[x],a[y]);
```

```
        printf("%d\n",ans);
    }
    return 0;
}
```

动态逆序对

传送门

题解

按道理来说应该是树套树都能过的，况且可能我的写法太丑了，luogu只有70分

代码

我的：

```
#pragma GCC optimize(fast)
#include <cstdio>
#include <iostream>
#include <cstdlib>
#include <ctime>
#define ls son[num][0]
#define rs son[num][1]
using namespace std;
const int maxn = 1e5+5;
const int INF = 2147483647;
typedef long long LL;
int readint(){
    int x=0,f=1;char s=getchar();
    #define sc (s=getchar())
    while(s<'0'||s>'9'){
```

```
        if(s=='-')
            f=-1;
        sc;
    }
    while(s>='0'&&s<='9'){
        x=(x<<3)+(x<<1)+(s^48);
        sc;
    }
    #undef sc
    return x*f;
}
int len,n,q;
int a[maxn<<5];
int key[maxn<<5],rd[maxn<<5];
int son[maxn<<5][2];
int siz[maxn<<5],cnt[maxn<<5];
struct treap{
    int rt;
    void push_up(int num){
        siz[num]=siz[ls]+siz[rs]+cnt[num];
    }
    void rotate(int &x,int d){
        int root=son[x][d^1];
        son[x][d^1]=son[root][d];
        son[root][d]=x;
        push_up(x);
        push_up(root);
        x=root;
    }
    void insert(int &num,int x){
        if(!num){
            num=++len;
```

```
            siz[num]=cnt[num]=1;
            key[num]=x;
            rd[num]=rand();
            return;
        }
        if(key[num]==x){
            cnt[num]++;
            siz[num]++;
            return;
        }
        int d=(x>key[num]);
        insert(son[num][d],x);
        if(rd[num]<rd[son[num][d]])
            rotate(num,d^1);
        push_up(num);
    }
    void deleted(int &num,int x){
        if(!num)
            return;
        if(x!=key[num])
            deleted(son[num][x>key[num]],x);
        else{
            if(!ls&&!rs){
                cnt[num]--;
                siz[num]--;
                if(cnt[num]==0)
                    num=0;
            }
            else if(ls&&!rs){
                rotate(num,1);
                deleted(rs,x);
            }
```

```
            else if(!ls&&rs){
                rotate(num,0);
                deleted(ls,x);
            }
            else{
                int d=rd[ls]>rd[rs];
                rotate(num,d);
                deleted(son[num][d],x);
            }
        }
        push_up(num);
    }
    LL get_rank(int num,int x){
        if(!num)
            return 0;
        if(key[num]==x)
            return siz[ls];
        if(key[num]<x)
            return siz[ls]+cnt[num]+get_rank(rs,x);
        return get_rank(ls,x);
    }
    LL get_Rank(int num,int x){
        if(!num)
            return 0;
        if(key[num]==x)
            return siz[rs];
        if(key[num]>x)
            return siz[rs]+cnt[num]+get_Rank(ls,x);
        return get_Rank(rs,x);
    }
    int find(int num,int x){
        if(!num)
```

```
112             return 0;
113         if(siz[ls]>=x)
114             return find(ls,x);
115         else if(siz[ls]+cnt[num]<x)
116             return find(rs,x-cnt[num]-siz[ls]);
117         else
118             return key[num];
119     }
120     int getpre(int num,int x){
121         if(!num)
122             return -INF;
123         if(key[num]>=x)
124             return getpre(ls,x);
125         else
126             return max(key[num],getpre(rs,x));
127     }
128     int getback(int num,int x){
129         if(!num)
130             return INF;
131         if(key[num]<=x)
132             return getback(rs,x);
133         else
134             return min(key[num],getback(ls,x));
135     }
136     int com(int &num,int x){
137         int pre=getpre(num,x),nxt=getback(num,x);
138         int f1=x-pre,f2=nxt-x;
139         if(f1>f2){
140             deleted(num,nxt);
141             return f2;
142         }
143         else{
```

```
            deleted(num,pre);
            return f1;
        }
    }
}tree[maxn<<3];
namespace segmentTree{
    void build(int num,int l,int r){
        for(int i=l;i<=r;i++)
            tree[num].insert(tree[num].rt,a[i]);
        if(l==r)
            return;
        int mid=(l+r)>>1;
        build(num<<1,l,mid);
        build(num<<1|1,mid+1,r);
    }
    void update(int num,int l,int r,int pos){
        if(pos<l||r<pos)
            return;
        tree[num].deleted(tree[num].rt,a[pos]);
    //  tree[num].insert(tree[num].rt,val);
        if(l==r)
            return;
        int mid=(l+r)>>1;
        update(num<<1,l,mid,pos);
        update(num<<1|1,mid+1,r,pos);
    }
    // zhen de rank = queryrank() + 1
    LL queryrank(int num,int l,int r,int goal_l,int goal_r,int val){
        if(goal_l>r||goal_r<l)
            return 0;
        if(l>=goal_l&&r<=goal_r)
            return tree[num].get_rank(tree[num].rt,val);
```

```
176         int mid=(l+r)>>1;
177         return querynum(num<<1,l,mid,goal_l,goal_r,val)+querynum(num<<1|1,mid+1,r,goal_l,goal_r,val);
178     }
179     LL queryRank(int num,int l,int r,int goal_l,int goal_r,int val){
180         if(goal_l>r||goal_r<l)
181             return 0;
182         if(l>=goal_l&&r<=goal_r)
183             return tree[num].get_Rank(tree[num].rt,val);
184         int mid=(l+r)>>1;
185         return queryRank(num<<1,l,mid,goal_l,goal_r,val)+queryRank(num<<1|1,mid+1,r,goal_l,goal_r,val);
186     }
187     int queryfind(int goal_l,int goal_r,int rank){
188         int l=0,r=1e8,ans;
189         while(l<=r){
190             int mid=(l+r)>>1;
191             if(querynum(1,1,n,goal_l,goal_r,mid)+1<=rank)
192                 ans=mid,l=mid+1;
193             else
194                 r=mid-1;
195         }
196         return ans;
197     }
198 }
199 int pos[maxn];
200 int main (){
201     //freopen("own.out","w",stdout);
202     //freopen("testdata.in","r",stdin);
203     //freopen("own.out","w",stdout);
204     srand(20050301);
205     int n=readint(),q=readint();
206     for(int i=1;i<=n;i++)
207         a[i]=readint(),pos[a[i]]=i;
```

```
    segmentTree::build(1,1,n);
    LL ans=0;
    for(int i=1;i<=n;i++)
        ans+=segmentTree::queryRank(1,1,n,1,i-1,a[i]);
//  cout<<ans<<endl;
    while(q--){
        //cout<<"Fuck"<<endl;
        printf("%lld\n",ans);
        int x=readint();
        ans-=segmentTree::queryRank(1,1,n,1,pos[x]-1,x);
        ans-=segmentTree::queryrank(1,1,n,pos[x]+1,n,x);
        segmentTree::update(1,1,n,pos[x]);
    }
    return 0;
}
```

线段树套树状数组（这个代码能过）

```
#include<iostream>
#include<algorithm>
#include<climits>
#include<cstdio>
using namespace std;
struct node
{
    int lson;
    int rson;
    int siz;
}tre[10000000];
void read(int &x)
{
    x=0;
```

```
    int f=1;
    char c=getchar();
    while('0'>c||c>'9')
    {
        if(c=='-')
            f=-1;
        c=getchar();
    }
    while('0'<=c&&c<='9')
    {
        x=(x<<3)+(x<<1)+c-'0';
        c=getchar();
    }
    x*=f;
}
void write(long long x)
{
    if(x<0)
    {
        putchar('-');
        write(-x);
        return;
    }
    if(x>9)
        write(x/10);
    putchar(x%10+'0');
}
int n,m;
int tot;
int cnt;
long long ans;
int q;
```

```
int p[200005];
int a[100005];
int b[200005];
int rt[200005];
int lowbit(int x)
{
    return x&(-x);
}
void insert(int &x,int pos,int val,int l=1,int r=cnt)
{
    if(!x)
        x=++tot;
    tre[x].siz+=val;
    if(l==r)
        return;
    int mid=(l+r)>>1;
    if(pos<=mid)
        insert(tre[x].lson,pos,val,l,mid);
    else
        insert(tre[x].rson,pos,val,mid+1,r);
}
void modify(int x,int v)
{
    int k=a[x];
    for(int i=x;i<=n;i+=lowbit(i))
        insert(rt[i],k,v);
}
int solve_rank(int x,int val,int l=1,int r=cnt)
{
    if(!x)
        return 0;
    if(l==r)
```

```
    {
        return 0;
    }
    int mid=(l+r)>>1;
    if(val<=mid)
        return solve_rank(tre[x].lson,val,l,mid);
    else
        return tre[tre[x].lson].siz+solve_rank(tre[x].rson,val,mid+1,r);
}
int Solve_rank(int l,int r,int val)
{
    if(l>r)
        return 0;
    int s=0, x=val;
    for(int i=l-1;i>0;i-=lowbit(i))
        s-=solve_rank(rt[i],x);
    for(int i=r;i>0;i-=lowbit(i))
        s+=solve_rank(rt[i],x);
    return s;
}
int solve_size(int l,int r)
{
    if(l>r)
        return 0;
    int s=0;
    for(int i=l-1;i>0;i-=lowbit(i))
        s-=tre[rt[i]].siz;
    for(int i=r;i>0;i-=lowbit(i))
        s+=tre[rt[i]].siz;
    return s;
}
void efsort(int l,int r)
```

```
{
    if(l == r) return ;
    int m = (l+r)>>1;
    efsort(l,m);
    efsort(m+1,r);
    int k=l,t1=l,t2=m+1;
    while(t1<=m&&t2<=r)
    {
        if(b[t1]<=b[t2])
        {
            p[k++]=b[t1++];
        }
        else
        {
            ans+=m-t1+1;
            p[k++]=b[t2++];
        }
    }
    while(t1<=m)
        p[k++]=b[t1++];
    while(t2<=r)
        p[k++]=b[t2++];
    for(int i=l;i<=r;i++)
        b[i]=p[i];
}
int main()
{
    read(n);
    read(m);
    for(int i=1;i<=n;i++)
    {
        read(a[i]);
```

```
        a[i]*=2;
        b[i]=a[i];
    }
    cnt=2*n;
    efsort(1,n);
    for(int i=1;i<=n;i++)
        b[a[i]]=i;
    for(int i=1;i<=n;i++)
        modify(i,1);
    for(int i=1;i<=m;i++)
    {
        //cout<<"ans:"<<ans<<'\n';
        write(ans);
        putchar('\n');
        read(q);
         q = b[q<<1];
        int t1=solve_size(1,q-1);
        ans=ans-(t1-Solve_rank(1,q-1,a[q]+1));
        //cout << "LEFT:";
        ans=ans-Solve_rank(q+1,n,a[q]-1);
        modify(q,-1);
    }
    return 0;
}
```

k大数查询

传送门

题解（by zxy）

先%一波zxy
这道题的思路特别巧妙，树套树不一定要用区间线段树套权值线段树，还可以反过来套。
我们维护一个动态开点的权值线段树，每个点代表权值$[l,r]$ 在整个区间的出现情况，套上一个动态开点的区间线段树，操作1对权值线段树单点修改，然后对每个点的$[a,b]$区间修改。
操作2 22先算右子树根的$[a,b]$和，如果答案在里面，找右子树，否则减掉之后找左子树，最后跑到叶子就得到了答案，时间复杂度$O(n\log^2)$,空间复杂度$O(n\log^2)$
这道题是需要卡常的，sum需要开unsigned int，输入的c cc要开long_long long_long。

代码

```cpp
#pragma GCC optimize(2)
#include <cstdio>
#define uint unsigned int
#define LL long long
const int MAXN = 400005;
int read()
{
    int x=0,flag=1;char c;
    while((c=getchar())<'0' || c>'9') if(c=='-') flag=-1;
    while(c>='0' && c<='9') x=(x<<3)+(x<<1)+(c^48),c=getchar();
    return x*flag;
}
int n,m,a,b,rt,cnt1,cnt2;LL c;
struct node1
{
    int ls,rs,lazy;
    uint sum;
}tr[MAXN*40];
struct node2
{
    int ls,rs,rt;
```

```
}Tr[MAXN*20];
void modify(int x,int l,int r,int v)
{
    tr[x].sum+=v*(r-l+1);
    tr[x].lazy+=v;
}
void up(int x)
{
    tr[x].sum=tr[tr[x].ls].sum+tr[tr[x].rs].sum;
}
void down(int x,int l,int r)
{
    if(!tr[x].lazy) return ;
    int mid=(l+r)>>1;
    if(!tr[x].ls) tr[x].ls=++cnt1;
    if(!tr[x].rs) tr[x].rs=++cnt1;
    modify(tr[x].ls,l,mid,tr[x].lazy);
    modify(tr[x].rs,mid+1,r,tr[x].lazy);
    tr[x].lazy=0;
}
void add(int &x,int l,int r,int L,int R)
{
    if(l>R || L>r) return ;
    if(!x) x=++cnt1;
    if(L<=l && r<=R)
    {
        modify(x,l,r,1);
        return ;
    }
    down(x,l,r);
    int mid=(l+r)>>1;
    add(tr[x].ls,l,mid,L,R);
```

```
    add(tr[x].rs,mid+1,r,L,R);
    up(x);
}
uint query(int x,int l,int r,int L,int R)
{
    if(!x || l>R || L>r) return 0;
    if(L<=l && r<=R)
        return tr[x].sum;
    down(x,l,r);
    int mid=(l+r)>>1;
    return query(tr[x].ls,l,mid,L,R)+query(tr[x].rs,mid+1,r,L,R);
}
void Modify(int &x,int l,int r,int id)
{
    if(!x) x=++cnt2;
    add(Tr[x].rt,1,n,a,b);
    if(l==r) return ;
    int mid=(l+r)>>1;
    if(mid>=id)
        Modify(Tr[x].ls,l,mid,id);
    else
        Modify(Tr[x].rs,mid+1,r,id);
}
int Query(int &x,int l,int r,int s)
{
    if(!x) x=++cnt2;
    if(l==r) return l;
    int mid=(l+r)>>1;uint t=query(Tr[Tr[x].rs].rt,1,n,a,b);
    //printf("%d %d %d %d %d\n",l,r,t,a,b);
    if(t>=s)
        return Query(Tr[x].rs,mid+1,r,s);
    return Query(Tr[x].ls,l,mid,s-t);
```

```
}
int main()
{
    n=read();m=read();
    while(m--)
    {
        int op=read();a=read();b=read();scanf("%lld",&c);
        if(op==1)
        {
            Modify(rt,-n,n,c);
        }
        if(op==2)
        {
            printf("%d\n",Query(rt,-n,n,c));
        }
    }
}
```