牛客练习赛66

比赛链接

E 骚区间

题意

给定一个长度为n的全排列，问存在多少个闭区间满足区间左端点恰为次小值，区间右端点恰为次大值。

题解 1

用权值线段树维护每个结点作为区间右端点且恰为次大值的左端点选区范围和每个结点作为区间左端点且恰为次小值的右端点选区范围。

枚举区间右端点，用树状数组维护合法左端点集合。时间复杂度 $O(n\log n)$

```
const int MAXN=1e6+5;
struct Tree{
    int lef[MAXN<<2],rig[MAXN<<2],w[MAXN<<2];
    int (*merge)(int,int);
    void Push_up(int k){w[k]=merge(w[k<<1],w[k<<1|1]);}
    void build(int k,int L,int R){
        lef[k]=L,rig[k]=R,w[k]=w[0];
        if(L==R)
        return;
        int M=L+R>>1;
        build(k<<1,L,M);
        build(k<<1|1,M+1,R);
    }
    void build(int n,int w_0){
        w[0]=w_0;
        build(1,1,n);
    }
    void update(int k,int p,int v){
        if(lef[k]==rig[k])
        return w[k]=v,void();
        int mid=lef[k]+rig[k]>>1;
        if(p<=mid)
        update(k<<1,p,v);
        else
        update(k<<1|1,p,v);
        Push_up(k);
    }
    void update_2(int k,int p,int v){
        w[k]+=v;
```

```
        if(lef[k]==rig[k])
        return;
        int mid=lef[k]+rig[k]>>1;
        if(p<=mid)
        update_2(k<<1,p,v);
        else
        update_2(k<<1|1,p,v);
    }
    int query(int k,int L,int R){
        if(L>R)
        return w[0];
        if(L<=lef[k]&&rig[k]<=R)
        return w[k];
        int mid=lef[k]+rig[k]>>1;
        if(R<=mid)
        return query(k<<1,L,R);
        else if(L>mid)
        return query(k<<1|1,L,R);
        return merge(query(k<<1,L,R),query(k<<1|1,L,R));
    }
}tree;
int Max(int a,int b){return a>b?a:b;}
int Min(int a,int b){return a<b?a:b;}
int n,t,a[MAXN],l1[MAXN],r1[MAXN],l2[MAXN],r2[MAXN];
vector<pair<int,int> >opt[MAXN];
#define lowbit(x) ((x)&(-x))
int c[MAXN];
void add(int pos,int v){
    while(pos<=n)
    c[pos]+=v,pos+=lowbit(pos);
}
int query(int L,int R){
    int ans=0,pos;
    pos=R;
    while(pos)
    ans+=c[pos],pos-=lowbit(pos);
    pos=L-1;
    while(pos)
    ans-=c[pos],pos-=lowbit(pos);
    return ans;
}
int main()
{
    n=read_int();
    _rep(i,1,n)
    a[i]=read_int();
    tree.merge=Max;
    tree.build(n,0);
    _rep(i,1,n){
        t=tree.query(1,a[i]+1,n);
```

```
        if(t){
            r1[i]=t;
l1[i]=max(tree.query(1,a[i]+1,a[t]-1),tree.query(1,a[t]+1,n))+1;
        }
        tree.update(1,a[i],i);
    }
    tree.merge=Min;
    tree.build(n,n+1);
    for(int i=n;i>=1;i--){
        t=tree.query(1,1,a[i]-1);
        if(t!=n+1){
            l2[i]=t;
r2[i]=min(tree.query(1,1,a[t]-1),tree.query(1,a[t]+1,a[i]-1))-1;
        }
        tree.update(1,a[i],i);
    }
    _rep(i,1,n) if(l2[i]){
        opt[l2[i]].push_back(make_pair(i,1));
        opt[r2[i]+1].push_back(make_pair(i,-1));
    }
    LL ans=0;
    _rep(i,1,n){
        _for(j,0,opt[i].size())
        add(opt[i][j].first,opt[i][j].second);
        if(r1[i])
        ans+=query(l1[i],r1[i]);
    }
    cout<<ans;
    return 0;
}
```

题解 2

可以用 set 维护端点的合法范围，按权值大小将每个数的下标插入 set□这里 copy 了一份别人的代码。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef unsigned long long ull;
#define pii pair<int,int>
#define pll pair<ll, ll>
#define pb push_back
#define X first
#define Y second
inline ll gcd(ll a, ll b) { while (b != 0) { ll c = a % b; a = b; b = c;
}return a < 0 ? -a : a; }
inline ll lowbit(ll x) { return x & (-x); }
const double PI = 3.14159265358979323846;
```

```
const int inf = 0x3f3f3f3f;
const ll INF = 0x3f3f3f3f3f3f3f3f;
const ll mod = 998244353;
inline ll rd() {
    ll x = 0, f = 1; char ch = getchar();
    while (ch<'0' || ch>'9') { if (ch == '-')f = -1; ch = getchar(); }
    while (ch >= '0' && ch <= '9') { x = (x << 3) + (x << 1) + (ch ^ 48);
ch = getchar(); }
    return x * f;
}
const double eps = 1e-6;
const int M = 1e6 + 10;
const int N = 1e6 + 10;
set<int>S;
int a[N], pos[N];
int n;
vector<int>L[N], R[N];
int ql[N], qr[N];
int c[N];
int query(int x) {
    int sum = 0;
    while (x) {
        sum += c[x];
        x -= lowbit(x);
    }
    return sum;
}
void add(int x, int y) {
    while (x <= n) {
        c[x] += y;
        x += lowbit(x);
    }
}
int main() {
    n = rd();
    for (int i = 1; i <= n; i++) {
        a[i] = rd();
        pos[a[i]] = i;
    }
    for (int i = 1; i <= n; i++) {
        int p = pos[i];
        S.insert(p);
        auto it = S.upper_bound(p);
        if (it != S.end()) {
            //l1[i] = *it;
            L[*it].pb(p);
            //it = S.upper_bound(*it);
            ++it;
            if (it == S.end())
                R[n].pb(p);
```

```
            else
                //r1[i] = *it - 1;
                R[*it - 1].pb(p);
        }
    }
    S.clear();
    for (int i = n; i >= 1; i--) {
        int p = pos[i];
        S.insert(-p);
        auto it = S.upper_bound(-p);
        if (it != S.end()) {
            qr[p] = -(*it);
            ++it;
            if (it == S.end()) {
                ql[p] = 1;
            }
            else {
                ql[p] = -(*it) + 1;
            }
        }
    }
    ll ans = 0;
    for (int i = 1; i <= n; i++) {
        for (auto& v : L[i])add(v, 1);
        if (ql[i] && qr[i])ans += 1ll * query(qr[i]) - query(ql[i] - 1);
        for (auto& v : R[i])add(v, -1);
    }
    cout << ans << endl;
}
```