题意解释

给定一个 $1 \sim n$ 的排列 a .

对于一个整数 $k \in [1,n]$, 将排列中 $\leqslant k$ 的项构成的子序列建**大根笛卡尔树**. 这棵笛卡尔树的所有节点的子树大小之和记为 s_k .

$\forall k \in [1,n]$, 求 s_k .

知识点提炼

线段树beats，笛卡尔树

核心解题思路

思路一：$n \leq 2000$

枚举 k ，$O(n)$ 构建笛卡尔树即可，复杂度$O(n^2)$。

或者思路二中，没有使用线段树beats，暴力实现。期望得分 40 分。

```
#include <bits/stdc++.h>
using namespace std;
#define SZ 666666
int n,a[SZ],r[SZ],p[SZ],q[SZ];
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;++i)
        scanf("%d",a+i),r[a[i]]=i;
    for(int i=1;i<=n;++i)
    {
        int g=r[i];
        //maintain p,q with seg tree beats
        //when some maximum is changed query in another seg
        for(int j=1;j<g;++j) q[j]=min(q[j],g-1);
        for(int j=g+1;j<=n;++j) p[j]=max(p[j],g+1);
        p[g]=1; q[g]=n;
        int ans=0;
        for(int j=1;j<=n;++j) if(a[j]<=i)
            for(int k=p[j];k<=q[j];++k) ans+=a[k]<=i;
        printf("%d\n",ans);
    }
}
```

思路二：无特殊限制

Part 1

笛卡尔树中以 i 为根的子树，对应排列中以 a_i 为最大值的极大区间，记为(l_i, r_i)。其中 l_i 是 a_i 左边第一个大于它的位置，r_i 同理。

那么求子树大小和等价于求区间长度和，也即$s_x = \sum_{a_i \le x}(r_i - l_i - 1)$。这里的 l_i, r_i 是在去掉 $a_i > x$后的子序列上定义的，而不是原序列。

所以说，只要能在从小到大插入 a 的过程中，维护 $\sum r_i$ 和 $\sum l_i$，就可以得到答案，下面考虑如何维护 $\sum r_i$。

Part 2

因为插入是从小到大的，所以新插入的值 x 会成为序列的最大值，设这个位置在原排列中的位置为 p。

考虑新序列的 r 和原序列的 r' 的关系

- $r_p = x + 1$
- 若 $i > p \wedge a_i < x$，那么 $r_i = r'_i + 1$
- 若 $i < p \wedge a_i < x$，那么 $r_i = \min(r'_i, p')$，其中 p' 是 x 在当前子序列中的位置。

也就是说我们需要支持以下操作

- 修改
  - 区间加
  - 单点修改
  - 区间对一个值取 $\min$
- 查询
  - 全局和

这个东西可以通过线段树beats维护。

具体来讲，对线段树的每一个节点维护：最大值v，次大值s，最大值个数len，非最大值的个数$slen$，区间和sum，同时需要维护标记：对最大值的加减标记t，和对非最大值的加减标记st。

时间复杂度 $O(n \log^2 n)$。

Part 3

在求出 $A_x = \sum r$ 之后，可以把原排列reverse之后用同样方法求一次，可得$B_x = \sum(x - l + 1)$，则有 $s_x = \sum(r - l + 1) = A_x + B_x - x(x + 2)$ 。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef unsigned int uint;
typedef unsigned long long ull;
typedef double dou;
typedef long double ld;
typedef pair<int, int> pii;
#define fi first
#define se second
#define mapa make_pair
template<typename _T> inline void read(_T &x){
    x=0; char c=getchar(); bool f=0;
```

```
    for(; c<'0'||c>'9'; c=getchar()) f|=(c=='-');
    for(; c>='0'&&c<='9'; c=getchar()) x=(x<<1)+(x<<3)+(c^48);
    x=(f)?(-x):x;
}
const int N=2e5+5;
int n;
int a[N], pos[N];
struct node{
    int mx, mx2, cnt, cnt2, tag, tag2;          // segment tree beats的一些信息
    ll sum;
    #define mx(p) tr[p].mx
    #define mx2(p) tr[p].mx2
    #define cnt(p) tr[p].cnt
    #define cnt2(p) tr[p].cnt2
    #define tag(p) tr[p].tag
    #define tag2(p) tr[p].tag2
    #define sum(p) tr[p].sum
}tr[N<<2];
#define ls p<<1
#define rs p<<1|1
inline void build(int p, int l, int r){
    cnt(p)=cnt2(p)=tag(p)=tag2(p)=sum(p)=0;
    mx(p)=-1; mx2(p)=-1e9;
    if(l==r) return ;
    int mid=(l+r)>>1;
    build(p<<1, l, mid); build(p<<1|1, mid+1, r);
}
inline void push_up(int p){
    mx(p)=max(mx(ls), mx(rs));
    sum(p)=sum(ls)+sum(rs);
    cnt(p)=0;
    if(mx(p)==mx(ls)) cnt(p)+=cnt(ls);
    if(mx(p)==mx(rs)) cnt(p)+=cnt(rs);
    cnt2(p)=cnt(ls)+cnt2(ls)+cnt(rs)+cnt2(rs)-cnt(p);
    if(mx(ls)==mx(rs)){
        mx2(p)=max(mx2(ls), mx2(rs));
    }
    else if(mx(p)==mx(ls)){
        mx2(p)=max(mx2(ls), mx(rs));
    }
    else {
        mx2(p)=max(mx(ls), mx2(rs));
    }
}
inline void calc(int p, int v){
    sum(p)+=(ll)v*cnt(p); mx(p)+=v; tag(p)+=v;
}
inline void calc2(int p, int v){
    sum(p)+=(ll)v*cnt2(p); mx2(p)+=v; tag2(p)+=v;
}
inline void push_down(int p){
    if(tag(p)||tag2(p)){
        if(mx(ls)+tag(p)==mx(p)) calc(ls, tag(p)), calc2(ls, tag2(p));
        else calc(ls, tag2(p)), calc2(ls, tag2(p));
        if(mx(rs)+tag(p)==mx(p)) calc(rs, tag(p)), calc2(rs, tag2(p));
```

```
        else calc(rs, tag2(p)), calc2(rs, tag2(p));
        tag(p)=0; tag2(p)=0;
    }
}
inline void ins(int p, int l, int r, int x, int v){
    if(l==r){
        mx(p)=v; cnt(p)=1; sum(p)=v;
        return ;
    }
    int mid=(l+r)>>1;
    push_down(p);
    if(x<=mid) ins(p<<1, l, mid, x, v);
    else ins(p<<1|1, mid+1, r, x, v);
    push_up(p);
}
inline void mdf(int p, int l, int r, int L, int R, int v){
    if(L>R) return ;
    if(mx(p)<=v) return ;
    int mid=(l+r)>>1;
    if(L<=l&&r<=R){
        if(mx2(p)<=v) {
            calc(p, v-mx(p));
            return ;
        }
    }
    push_down(p);
    if(L<=mid) mdf(p<<1, l, mid, L, R, v);
    if(R>mid) mdf(p<<1|1, mid+1, r, L, R, v);
    push_up(p);
}
inline void add(int p, int l, int r, int L, int R, int v){
    if(L>R) return ;
    if(L<=l&&r<=R){
        calc(p, v); calc2(p, v);
        return ;
    }
    int mid=(l+r)>>1;
    push_down(p);
    if(L<=mid) add(p<<1, l, mid, L, R, v);
    if(R>mid) add(p<<1|1, mid+1, r, L, R, v);
    push_up(p);
}
inline int getsz(int p, int l, int r, int L, int R){
    if(L>R) return 0;
    if(L<=l&&r<=R) return cnt(p)+cnt2(p);
    int mid=(l+r)>>1, ret=0;
    push_down(p);
    if(L<=mid) ret=getsz(p<<1, l, mid, L, R);
    if(R>mid) ret+=getsz(p<<1|1, mid+1, r, L, R);
    return ret;
}
ll ans[N];
int main(){
    read(n);
    for(int i=1; i<=n; ++i) read(a[i]), pos[a[i]]=i;
```

```
    for(int i=1; i<=n; ++i) ans[i]-=(ll)i*i;
    for(int _=1; _<=2; ++_){                // 求\sum r 和 \sum l
        build(1, 1, n);
        for(int i=1; i<=n; ++i){
            ins(1, 1, n, pos[i], i);        // 分别进行题解中的三种修改
            add(1, 1, n, pos[i]+1, n, 1);
            mdf(1, 1, n, 1, pos[i]-1, getsz(1, 1, n, 1, pos[i]-1));
            ans[i]+=sum(1);
        }
        for(int i=1; i<=n; ++i) pos[i]=n-pos[i]+1;
    }
    for(int i=1; i<=n; ++i) printf("%lld\n", ans[i]);
    return 0;
}
```

本题易错点

- 要注意到移动左端点和移动右端点对于答案的影响是不同的