# 1 rotinv

## 1.1 30%

$O(n^3)$ 暴力：枚举每个循环状态，再暴力计算逆序对数。

## 1.2 60%

法一：

将原序列复制一份接在自己后面，形成一个长度为 $2n$ 的序列，先暴力计算 $[1,n]$ 的逆序对数，然后将这个区间向右边移动，维护答案。具体来说，就是假如我们算出了 $[i,j]$ 中的逆序对数，我们怎么算 $[i+1,j+1]$ 呢，其实就是加上 $[i+1,j]$ 中比 $a[j+1]$ 大的数的个数再减去 $[i+1,j]$ 中比 $a[i]$ 小的数的个数。所以我们可以暴力算那两个数量，连续 $n$ 个长度为 $n$ 的子区间的逆序数的和就是答案，复杂度 $O(n^2)$。

法二：

计算一个长度为 $n$ 的序列的逆序对的个数可以通过数据结构（树状数组或线段树）优化到 $O(nlogn)$，维护 $[1,i-1]$ 的一个值的分布情况（加入到线段树中），每次查询这些数比 $a[i]$ 大的有多少个，再将 $a[i]$ 加入到线段树中去。所以计算 $n$ 个序列的逆序数只需要 $O(n^2logn)$.

## 1.3 100%

将上面的法一和法二结合一下，用线段树优化法一中那个暴力，使得可以 $O(logn)$ 查询，最后复杂度 $O(nlogn)$.

## 2 rise

### 2.1 30%

暴力 $O(n^2)$.

### 2.2 100%

用线段树解决，每个节点维护：

- 最大值：vmax
- 这个区间对应的答案：c1
- 这个区间的右儿子（如果有的话）去掉小于等于左儿子最大值以后的剩下的那些数对应的答案：c2

每个节点支持一种询问 query(nd,h)：询问去掉 nd 这个节点中小于等于 h 的数之后这个区间对应的答案.

假如我们可以实现上面这个操作的话，我们先将询问区间 $[L, R]$ 对应的那些节点提取出来排好：nd1, nd2, nd3, ... nds.

query(nd1,0) + query( nd2, max(nd1)) + query(nd3,max(nd1,nd2)) + ... 就是答案.

我们考虑怎样实现这个询问过程 query(nd,h):
如果是叶子节点，只需要返回 [h < a]
如果 h < nd 左儿子最大值，则返回 query(nd 左儿子,h) + 右儿子 c2
如果 h >= nd，则返回 query(nd 右儿子,h)

其实可以将 query 看成我用 h 去砍一刀，然后计算答案，上面的过程的正确性还是比较容易理解的，如果我还有什么没有说清楚的可以看一下代码.

容易发现 query 每次最多选两个儿子中的一个节点走下去，所以最多走 $O(logn)$ 个节点，所以复杂度是 $O(logn)$ 的.

每个节点在算 c2 时需要调用一次 query，所以 build 的时间复杂度是 $O(nlogn)$ 的.

询问的时候，最多提取出来 $O(logn)$ 个区间，每个区间进行一次 $O(logn)$ 的询问，所以单次询问是 $O(log^2n)$ 的.

总的复杂度是 $O(nlogn + mlog^2n)$ 的.

# 3 seqmod

## 3.1 30%

随便怎么暴力都可以...

## 3.2 100%

就是有序链剖，细节我课上讲过了，大家看一下代码可以看懂的. 与普通链剖的区别就是这个东西要在乎顺序，所以不能用 swap 那种写法，两个各跳各的，还要注意合并的顺序.