知识精炼（九）

主讲人：邓哲也

Codeforces 220E

给你一个长度为 n 的序列 A[1], A[2], …, A[n].

问你有多少组 (l, r) (1 <= l < r <= n) 满足

A[1], A[2], …, A[l], A[r], A[r+1], ..., A[n] 的逆序

对个数不超过 k。

$2 <= n <= 10^5$, $0 <= k <= 10^{18}$

input	input
3 1 1 3 2	5 2 1 3 2 1 7
output	**output**
3	6

Codeforces 220E

对于第二个样例：

5 2

1 3 2 1 7

看一下可行的是哪些 (l, r)：

(1, 3) (1, 4) (1, 5)

(2, 4) (2, 5)

(3, 5)

你有什么发现？

Codeforces 220E

可以发现，如果（l，r）可行，那么（l，r+1）一定也可行。

因此我们枚举 l，只要算出了最小的 r，就可以把 n-r+1

加入答案。

进一步的，随着 l 右移，最小的 r 也会右移！

随着 l 从 1 移到 n， r 也会从 2 移到 n。

每次 l 右移一位， r 会右移若干位。

直接维护 l 和 r 两个指针的移动即可。

Codeforces 220E

初始状态 l = 1, r = 2

建两个树状数组 S 和 T。

S 维护 A[1..l]，T 维护 A[r..n]

一开始先计算逆序对。

如果逆序对 > k，那么 r 要右移。

r 右移一位，逆序对数会发生什么变化。

Codeforces 220E

S 中大于 A[r] 的数 对逆序对的贡献会减一。

T 中小于 A[r] 的数 对逆序对的贡献会减一。

一直右移 r，直到逆序对 <= k。

这个时候就求出了 l 对应的最小的 r，把 n - r + 1 加入答案。

然后 l 右移一位，对逆序对会产生什么影响呢。

S 中大于 A[l] 的数 对逆序对的贡献会加一。

T 中小于 A[l] 的数 对逆序对的贡献会加一。

Codeforces 220E

然后继续维护 r 右移。

依次类推，当 l 移动到 n 或者 A[1..l] 中的逆序对已经大于 k 时算法结束。

时间复杂度：O(n log n).

Codeforces 220E

```
#define N 100010
typedef long long ll;
int a[N], bin[N], n, cnt, S[N], T[N];
ll K;
void add(int *bit, int x, int v){
    for (;x <= cnt; x += x & -x) bit[x] += v;
}

int ask(int *bit, int x){
    int ret = 0;
    for (;x;x -= x & -x) ret += bit[x];
    return ret;
}
```

Codeforces 220E

```
int main(){
    scanf("%d%I64d", &n, &K);
    for(int i = 1;i <= n;i ++){
        scanf("%d", &a[i]);
        bin[++ cnt] = a[i];
    }
    sort(bin + 1, bin + n + 1);
    cnt = unique(bin + 1, bin + n + 1) - bin - 1;
    for(int i = 1;i <= n;i ++)
        a[i] = lower_bound(bin + 1, bin + cnt + 1, a[i]) - bin;
    ll cur = 0;
    for (int i = n;i >= 1;i --){
        add(T, a[i], 1);
        cur += ask(T, a[i] - 1);
    }
    add(T, a[1], -1);
    add(S, a[1], 1);
```

Codeforces 220E

```
    int l = 1, r = 2;
    ll ans = 0;
    while (l < n){
        while ((cur > K && r + 1 <= n) || r <= l) {
            cur -= l - ask(S, a[r]);
            cur -= ask(T, a[r] - 1);
            add(T, a[r], -1);
            r ++;
        }
        if (cur > K) break;
        ans += n - r + 1;
        l ++;
        add(S, a[l], 1);
        cur += l - ask(S, a[l]);
        cur += ask(T, a[l] - 1);
    }
    cout << ans << endl;
    return 0;
}
```

下节课再见