n 个整数中找出最大的前 m 个数问题

课上我们讨论了从 n 个整数中找出最大的前 m 个数这一问题，有同学提出可以使用堆排序，不考虑建堆（$\mathrm{O(n)}$）的话，可以很容易理解时间复杂度为$\mathrm{O(m*\log(n))}$，这是一个非常好的解决策略。不过，我想到的是了另一个基于只处理一边的快排思路，现在我稍微算了算这个思路的平均时间复杂度。

A：输入的数组
m: 找出最大的前 m 个数的定义
p：PARTITION 开始编号
r：PARTITION 尾部编号

方法描述：
基于第 m 个数来调整，使得比第 m 个数字大的所有数字都位于数组的左边，比第 m 个数字小的所有数字都位于数组的右边。这样调整之后，位数数组中左边的 m 个数字就是最大的 m 个数字，但是这 m 个数字不一定是顺序的)。这种思路时间复杂度为 O(n)。

伪代码分析：
m-LARGE-SELECT(A, m, p, r)
if p == r
    return A
q = RANDOMIZED-PARTITION(A, p, r)   //随机选取 pivot 并划分，将大的数置前，小的数置后（快排分割的函数部分），返回 pivot 的大小
k = q-p+1
if m == k
    return A
else if m<k
    return m-LARGE-SELECT(A, m, p, q-1)
else if m>k
    return m-LARGE-SELECT(A, m-k, p, q+1)

其中，q 是 pivot，返回的数组 A 中前 m 个数就是我们要的最大的 m 个数，只不过这 m 个数不是从大到小排序的
时间复杂度证明：

令$X_k = I\{子数组A[p \ldots\ldots q]有k个元素\}$

$$\mathrm{E}(X_k) = \frac{1}{\mathrm{n}}$$

寻求上界的递归式为：

$$\mathrm{T(n)} \le \sum_{k=1}^{n} X_k * (T(\max(k-1, n-k)) + O(n)$$

$$\mathrm{T(n)} \leq \sum_{k=1}^{n} X_k * T((\max(k-1, n-k)) + O(n)$$

两边取期望，并考虑独立随机变量得：

$$\mathrm{E}\big(\mathrm{T(n)}\big) \leq \sum_{k=1}^{n} E(X_k) * E[T(\max(k-1, n-k))] + O(n)$$

$$\mathrm{E}\big(\mathrm{T(n)}\big) \leq \frac{1}{\mathrm{n}} * \sum_{k=1}^{n} E[T(\max(k-1, n-k))] + O(n)$$

考虑：

$$\max(k-1, n-k) = \begin{cases} k-1 & k > \left\lceil \frac{n}{2} \right\rceil \\ \mathrm{n}-k & k \leq \left\lceil \frac{n}{2} \right\rceil \end{cases}$$

$$\mathrm{E}[T(n)] \leq \frac{2}{\mathrm{n}} * \sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + O(n)$$

使用替代法来证明：

假设对于满足这个递归式初始条件的某个常数 a，有$\mathrm{E}[T(n)] \leq a * n$，假设对于小于某个常数的n，有$\mathrm{T(n)} = \mathrm{O}(1)$，用bn来表示上式的$O(n)$部分。

$$\therefore \mathrm{E[T(n)]} \leq \frac{2}{\mathrm{n}} * \sum_{k=\left\lfloor \frac{n}{2} \right\rfloor}^{n-1} a * k + b * n$$

易得（参考中文版算法导论（第三版）P122）

$$\mathrm{E[T(n)]} \leq \frac{3\mathrm{an}}{4} + \frac{a}{2} + bn = an - \left(\frac{an}{4} - \frac{a}{2} - bn\right)$$

还需要证明当 n 足够大时，最后一个表达式至多为an。

$$\frac{an}{4} - \frac{a}{2} - bn \geq 0$$

$$\mathrm{n}\left(\frac{a}{4} - b\right) \geq \frac{a}{2}$$

如果$\left(\frac{a}{4} - b\right) > 0$，则：

$$\mathrm{n} \geq \frac{2\mathrm{a}}{a - 4b}$$

根据之前的假设，对于小于某个常数的n，有$\mathrm{T(n)} = \mathrm{O}(1)$，当这个常数为$\frac{2\mathrm{a}}{a-4b}$时，对于

$$\mathrm{n} < \frac{2\mathrm{a}}{a - 4b}$$

有：

$$\mathrm{T(n)} = \mathrm{O}(1)$$

那么：

$$\mathrm{E}[T(n)] = O(n)$$

结论：假设所有元素互异，在O(n)时间内，可以找到比某一顺序统计量大的所有数（无序）。