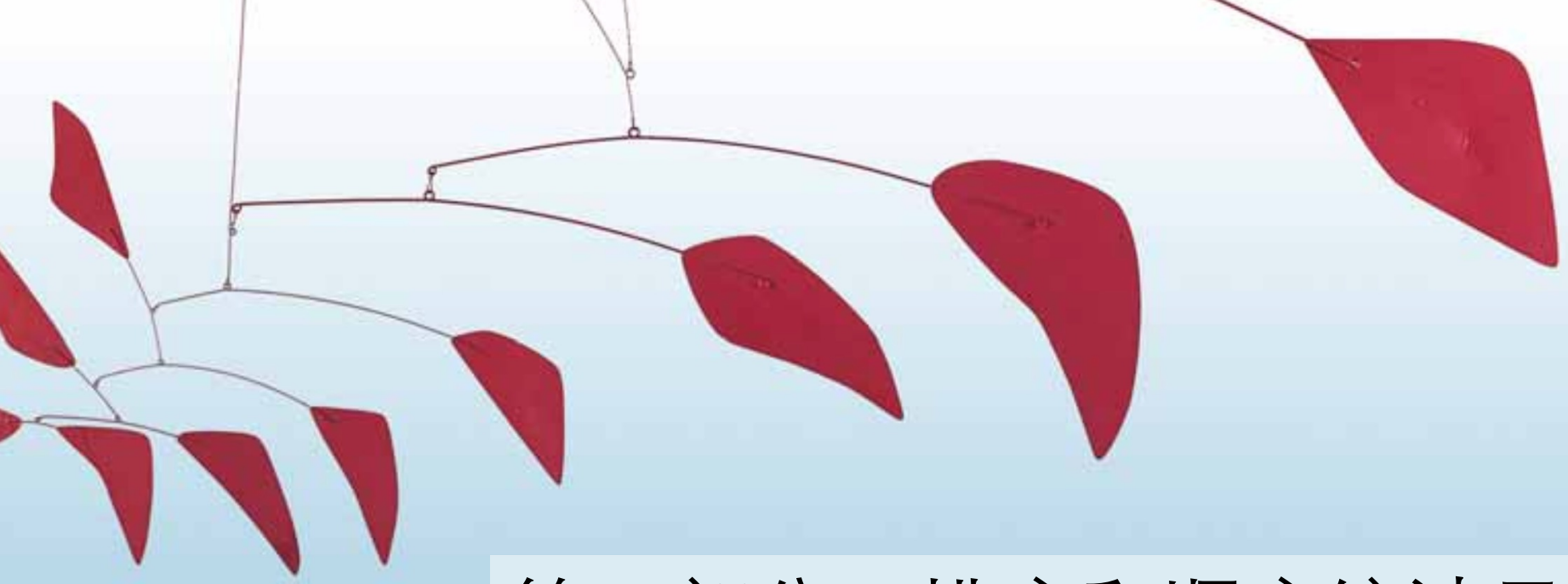


第二部分：排序和顺序统计量

INTRODUCTION TO

ALGORITHMS

THIRD EDITION

中位数和顺序统计量

《算法导论》—— 第8讲

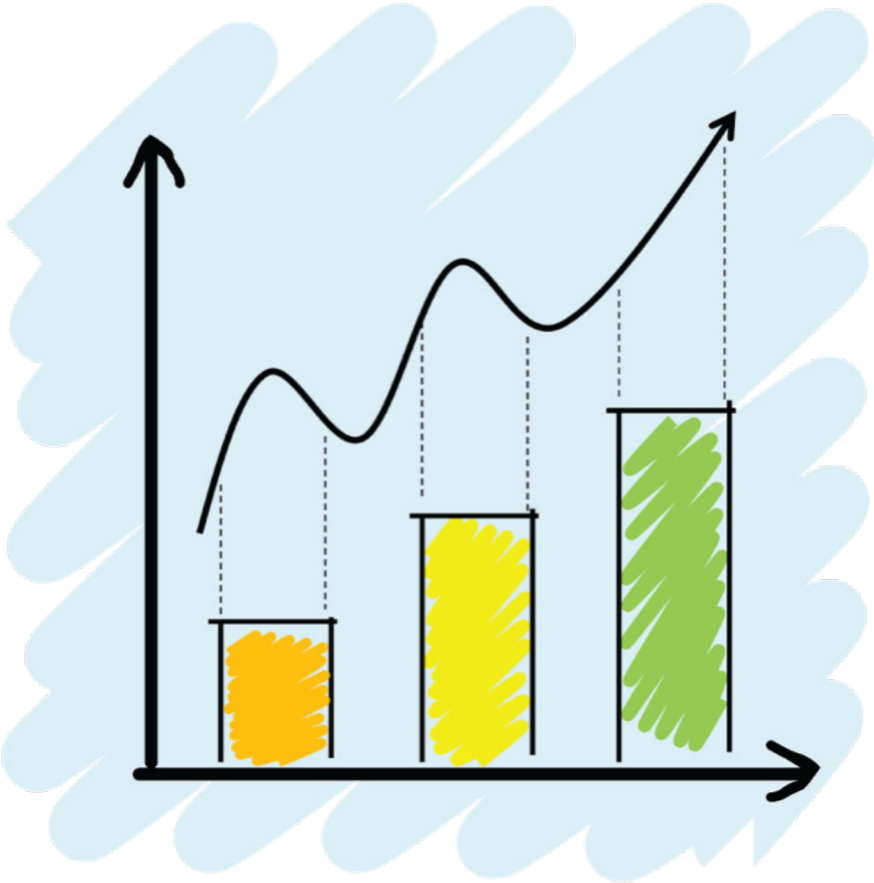
jiacaicui@163.com

内容提要

- 顺序统计量
  - 一些特殊的顺序统计量
  - 选择问题
- 期望为线性时间的选择算法
  - 分治策略 —— 利用 RANDOMIZED-PARTITION
  - 一个实用的随机算法
- 最坏为线性时间的选择算法
  - 分治策略
  - BFPRT 算法
  - 一个更具有理论意义的算法

顺序统计量

选择问题



顺序统计量（order statistics）

- 在一个由 n 个元素组成的集合中，第 i 个顺序统计量是该集合中第 i 小的元素。
  - 最小值是第 1 个顺序统计量（$i = 1$）
  - 最大值是第 n 个顺序统计量（$i = n$）
- 中位数（median）是它所属集合的“中点元素”。
  - 当 n 为奇数时，中位数位于 $i = \frac{n+1}{2}$ 处；
  - 当 n 为偶数时，中位数位于 $i = \frac{n}{2}$ 和 $i = \frac{n}{2} + 1$ 这两处。
  - 不考虑 n 的奇偶性，下中位数位于 $i = \left\lfloor \frac{n+1}{2} \right\rfloor$ 处，上中位数位于 $i = \left\lceil \frac{n+1}{2} \right\rceil$ 处。

选择问题

- 输入：一个包含 n 个数的集合 A 和一个整数 i 。
  - n 个数互异； $1 \leq i \leq n$ 。
- 输出：元素 $x \in A$ ，且 A 中恰好有 $i-1$ 个其他元素小于它。

- 算法：
  - $O(n \log n)$ — 使用堆排序或归并排序对输入数据进行排序，然后在输出数组中根据下标找出第 i 个元素即可。
  - 平均情况 $O(n)$ — 朴素的分治策略，利用 RANDOMIZED-PARTITION 过程。
  - 最坏情况 $O(n)$ — 特殊的分治策略，BFPRT 算法。

最小值和最大值

```
MINIMUM(A)
1  min = A[1]
2  for i = 2 to A.length
3      if min > A[i]
4          min = A[i]
5  return min
```

- 上述算法说明寻找最小值最多需要 $n-1$ 次比较。
- 寻找最小值最少需要多少次比较？
  - 可以将寻找最小值的算法看作是个元素之间的一场锦标赛，每次比较都是锦标赛中的一场。
  - 除了最终的胜者之外，每个元素都至少输掉一场比赛，共 $n-1$ 场。
- 综上，确定最小值，必须要做 $n-1$ 次比较， MINIMUM 算法是最优的。

同时找到最小值和最大值

- 朴素算法：分别找最小值和最大值，需要 $2n-2$ 次比较。
- 实际上只需要最多 $3\left\lfloor \frac{n}{2} \right\rfloor$ 次比较就可以同时找到最小值和最大值。
  - 思路：对输入元素成对地处理。
    1. 将一对输入元素相互进行比较。
    2. 把较小的元素与当前的最小值进行比较，并更新最小值。
    3. 把较大的元素与当前的最大值进行比较，并更新最大值。
  - 不难看出，上述算法可以正确地同时求出最大值和最小值，并且最多需要$3\left\lfloor \frac{n}{2} \right\rfloor$ 次比较操作。
  - 注：向下取整是因为初始化的时候可以任意选一对元素作为最小值和最大值，这一步不需要比较。

期望为线性时间的选择算法

实用的求顺序统计量的方法

选择问题的分治算法

```
RANDOMIZED-SELECT(A, p, r, i)
1  if p == r
2      return A[p]
3  q = RANDOMIZED-PARTITION(A, p, r)
4  k = q − p + 1
5  if i == k            // the pivot value is the answer
6      return A[q]
7  elseif i < k
8      return RANDOMIZED-SELECT(A, p, q − 1, i)
9  else return RANDOMIZED-SELECT(A, q + 1, r, i − k)
```

- 分：将输入数组 A 划分成前 $k-1$ 小的元素，第 k 小的元素和后 $n-k$ 小的元素。
- 治：若 $i=k$，则 $A[k]$ 即为所求。若 $i<k$，则递归地到前 k 小的元素中选择第 i 小的元素。若 $i>k$，则递归地到后 $n-k$ 小的元素中寻找第 $i-k$ 小的元素。

RANDOMIZED-SELECT 的运行时间

- 最坏情况运行时间：$O(n^2)$，每次划分都极不平衡的情况。
- 期望运行时间：$E[T(n)] = O(n)$。
  - 记算法第 3 行结束后子数组 $A[p..q]$ 正好包含 k 个元素为事件 H_k，根据 RANDOMIZED- PARTITION 算法的过程，有

$$\Pr\{H_k\} = \Pr\{q = p + k - 1\} = \frac{1}{r - p + 1} = \frac{1}{n}$$

  - 记 $X_k = I\{H_k\}$，则 $E[X_k] = \Pr\{H_k\} = \frac{1}{n}$。考虑运行时间的上界，有

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

  - 其中 X_k 和 $T(\max(k-1, n-k))$ 是独立的（详见*练习8-1问题3*），且

$$\max(k-1, n-k) = \begin{cases} k-1, & k > \lceil n/2 \rceil \\ n-k, & k \le \lceil n/2 \rceil \end{cases}$$

RANDOMIZED-SELECT 期望运行时间

$$E[T(n)] \le E\left[\sum_{k=1}^{n} X_k \cdot T(\max(k-1, n-k)) + O(n)\right]$$

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

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

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

- 使用代入法来证明 $E[T(n)] = O(n)$，将 $E[T(n)] \le cn$ 代入上式，有

$$E[T(n)] \le \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + an \le \frac{2c}{n} \cdot \frac{\left(\frac{n}{2} - 1 + n - 1\right) \cdot \left(\frac{n}{2} + 1\right)}{2} + an$$

$$= c\left(\frac{3n}{4} + \frac{1}{2} - \frac{2}{n}\right) + an \le \frac{3cn}{4} + \frac{c}{2} + an = cn - \left(\frac{cn}{4} - \frac{c}{2} - an\right) \le cn$$

- 其中，最后一步需要 $c > 4a$ 且 $n \ge 2c/(c-4a)$。

最坏情况为线性时间的选择算法

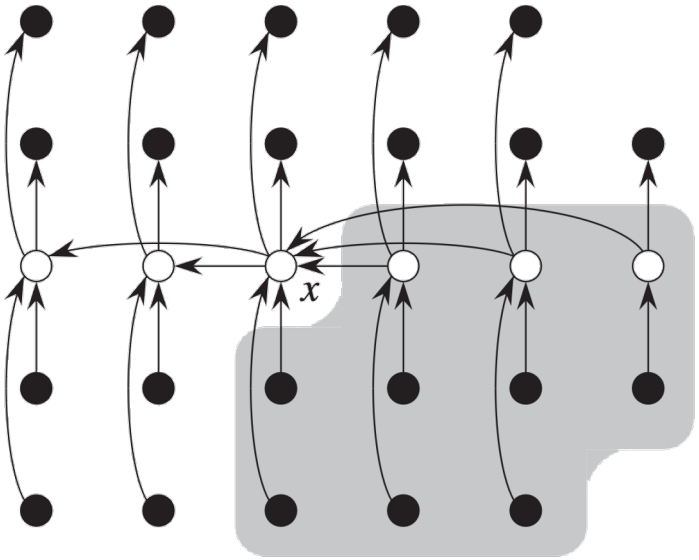
给出尽可能平衡的划分

BFPRT 的 SELECT 算法步骤

- SELECT 算法基本思路依旧是分治，只不过它能够保证得到一个好的划分。
- 通过执行下列步骤，算法 SELECT 可以确定一个有 $n > 1$ 个不同元素的输入数组中第 i 小的元素。（$n = 1$时 SELECT 直接返回唯一的元素即可）

1. 将输入数组的 n 个元素划分为每组 5 个元素的 $\lfloor n/5 \rfloor$ 组，以及由剩下的 $n \bmod 5$ 个元素组成的一组（可能没有）。
2. 寻找这 $\lfloor n/5 \rfloor$ 组中每一组的中位数。
3. 对第 2 步中找出的 $\lfloor n/5 \rfloor$ 个中位数，递归调用 SELECT 以找出其中位数 x 。
4. 以 x 为主元对数组进行划分，记划分后 x 的下标为 k 。
5. 若 $i = k$，则返回 x 。若 $i < k$，则递归调用 SELECT 来寻找前 $k - 1$ 个数中的第 i 小的数。若 $i > k$，则递归调用 SELECT 来寻找后 $n - k$ 个数中第 $i - k$ 小的数。

BFPRT 算法的子问题规模



- 如上图所示，除了 x 所在的组和不能整除所剩下的组之外，至少有一半的组中有 3 个元素大于 x，所以大于 x 的元素个数至少为：

$$3\left(\left\lceil\frac{1}{2}\left\lceil\frac{n}{5}\right\rceil\right\rceil-2\right)\geq\frac{3n}{10}-6$$

- 总元素个数为 n，因而至多有 $7n/10+6$ 个元素小于 x；同理，大于 x 的元素最多也是 $7n/10+6$ 个，从而算法第 5 步递归调用的子问题规模最大为 $7n/10+6$。

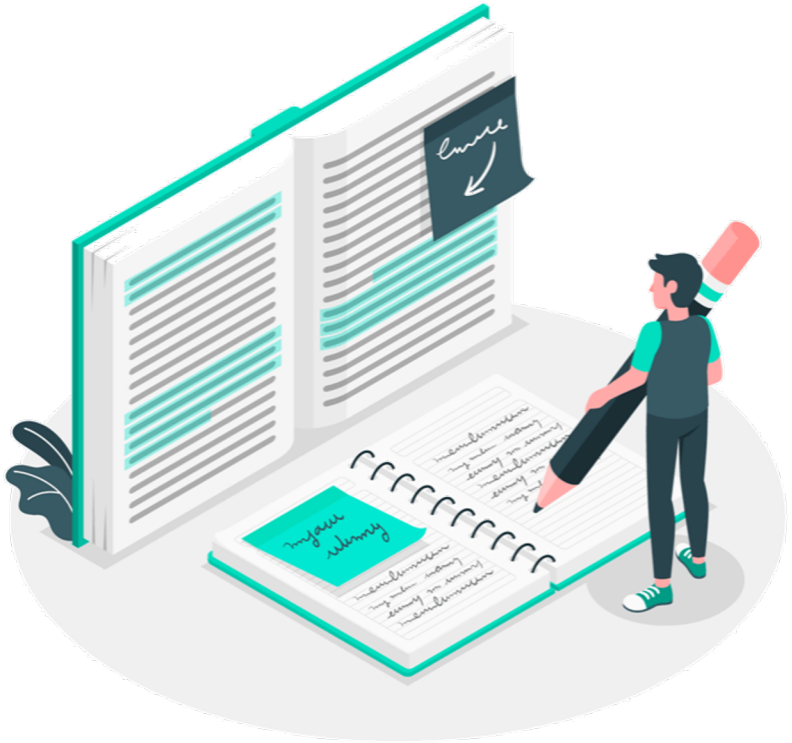
BFPRT 算法的最坏情况运行时间

- 记算法的运行时间为 $T(n)$，其中
  - 步骤 1 只是一种下标看法，不需特别做些什么。
  - 步骤 2 取 $\lceil n/5 \rceil$ 个长度为 $O(1)$ 的子数组的中位数，需要 $O(n)$。
  - 步骤 3 递归调用 SELECT 找 $\lceil n/5 \rceil$ 个中位数的中位数 x，需要 $T(\lceil n/5 \rceil)$。
  - 步骤 4 以 x 为主元划分数组，需要 $O(n)$。
  - 步骤 5 递归对划分子数组递归调用 SELECT，子数组规模最大为 $7n/10 + 6$，最多需要 $T(7n/10 + 6)$。
- 综上，我们有 $T(n) \le T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n)$，不难用代入法证明 $T(n) \le cn$：

$$T(n) \le c(n/5 + 1) + c(7n/10 + 6) + an = cn - \big((c/10 - a)n - 7c\big) \le cn$$

- 最后一步需要 $c > 10a$ 且 $n > 7c/(c/10 - a)$。因此 $T(n) = O(n)$。

本讲小结



内容提要

- 顺序统计量
  - 一些特殊的顺序统计量
  - 选择问题
- 期望为线性时间的选择算法
  - 分治策略 —— 利用 RANDOMIZED-PARTITION
  - 一个实用的随机算法
- 最坏为线性时间的选择算法
  - 分治策略
  - BFPRT 算法
  - 一个更具有理论意义的算法

The End!