

算法设计与分析

主讲人：吴庭芳

Email: tfwu@suda.edu.cn

苏州大学

计算机科学与技术学院

SCHOOL OF
COMPUTER SCIENCE &
TECHNOLOGY
SOOCHOW UNIVERSITY
计算机科学与技术学院
苏州大学

学院 教师 学生 校友 捐赠 基金会





第六讲 选择算法

内容提要:

- 最小值和最大值
- 期望为线性时间的选择算法
- 最坏情况为线性时间的选择算法



选择问题描述

- 在一个由 n 个元素组成的集合中，第 i 个顺序统计量是指该集合中第 i 小的元素
- **选择问题**：从一个由 n 个互异数值构成的集合中选择第 i 个顺序统计量
 - 输入：一个包含 n 个（互异）数的集合 A 和一个整数 i ($1 \leq i \leq n$)
 - 输出：元素 $x \in A$ ，满足 A 中恰好有 $i-1$ 个元素小于 x
- 选择问题可以在 $O(n \lg n)$ 时间内解决：
 - 用堆排序或归并排序对输入数组进行排序
 - 再在输出数组中根据下标找出第 i 个元素即可
- 还有其他更快的算法吗？



最小值和最大值

- 在一个有 n 个元素的集合中，需要做多少次比较才能确定其最小或最大元素呢？
- 假设集合元素存放在数组 A 中，且 $A.length = n$

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

- 可以给出上述寻找最小值算法的比较次数的上界： $n-1$ 次
- 若需要同时寻找集合中的最大值和最小值，共需要多少次比较呢？
 - 如果分别独立寻找集合的最小值和最大值，则各需要做 $n-1$ 次比较，共需要 $2n-2$ 次比较



最小值和最大值

□ 同时寻找最小值和最大值:

- **成对处理输入元素**: 先比较两个输入元素, 然后将较小的与当前最小值比较, 较大的与当前最大值比较 (每对元素需要 **3 次比较**)
- 记录比较过程中遇到的最小值和最大值

MAX-MINIMUM (A)

```
1  if A.length is odd
2      min = A[1]
3      max = A[1]
4      i = 2
5      while i ≤ A.length
6          min = MIN(MIN(A[i], A[i+1]), min)
7          max = MAX(MAX(A[i], A[i+1]), max)
8          i = i+2
9      end
10 else min = MIN(A[1], A[2])
11      max = MAX(A[1], A[2])
12      i = 3
13      ...
14  return min, max
```




最小值和最大值

□ 如何设定当前最小值和最大值的初始值：依赖于 n 的奇偶性

- 如果 n 是奇数，将最小值和最大值的初始值都设为第一个元素值
- 如果 n 是偶数，就对前两个元素做一次比较，以决定最小值和最大值的初始值

□ 总的比较次数：

- 如果 n 是奇数，总共做了 $3\lfloor n/2 \rfloor$ 次比较
- 如果 n 是偶数，总共做了 $3(n-2)/2+1$ 次比较
- 因此，不管是哪一种情况，总的比较次数至多是： $3\lfloor n/2 \rfloor$



第六讲 顺序统计学

内容提要:

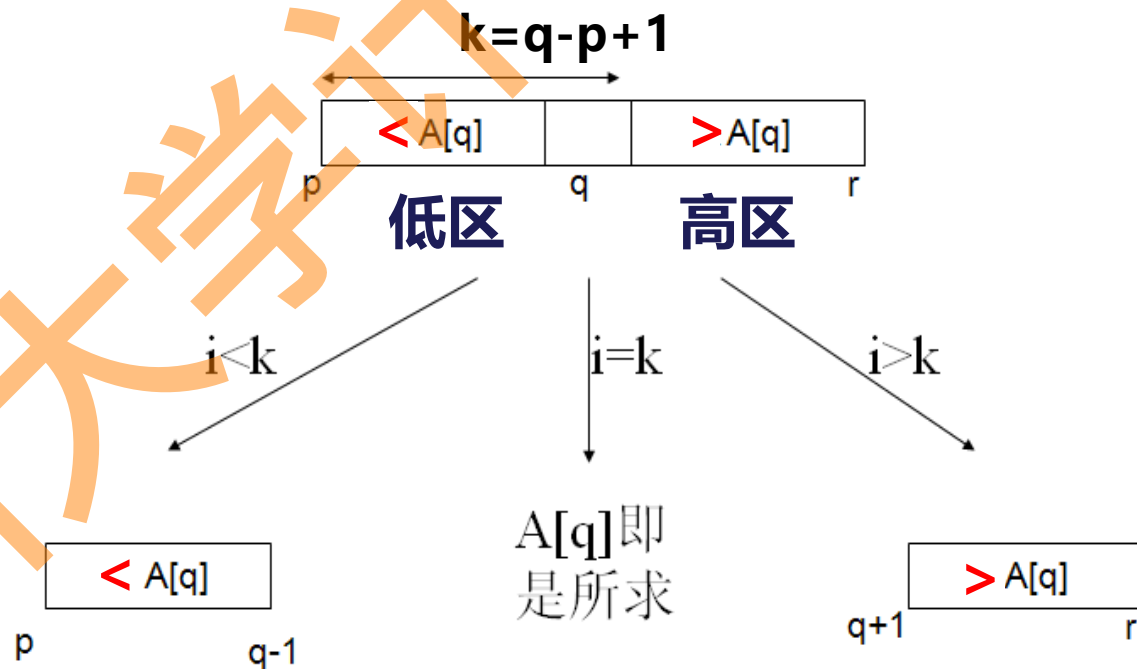
- 最小值和最大值
- 期望为线性时间的选择算法
- 最坏情况为线性时间的选择算法



期望为线性时间的选择算法

□ 随机选择算法 RANDOMIZED-SELECT 的基本思想:

- 分解: 借鉴快速排序的随机划分过程, 对输入数组进行划分
- 解决: 但是与快速排序算法不同的是, 随机选择算法只需要递归求解划分的一边





期望为线性时间的选择算法

□ RANDOMIZED-SELECT 的伪代码如下：

```
RANDOMIZED-SELECT ( $A, p, r, i$ )
1  if  $p = r$                                 //边界问题处理
2      return  $A[p]$ 
3   $q = \text{RANDOMIZED-PARTITION}(A, p, r)$  //进行划分，返回划分主元下标
4   $k = q - p + 1$                             //主元元素是第  $k$  个顺序统计量
5  if  $i == k$                                 //主元元素就是需要返回的数值
6      return  $A[q]$ 
7  else if  $i < k$                             //第  $i$  个顺序统计量落在划分的低区
8      return RANDOMIZED-SELECT ( $A, p, q-1, i$ )
9  else return RANDOMIZED-SELECT( $A, q+1, r, i-k$ ) //第  $i-k$  小的元素
```



期望为线性时间的选择算法

□ 时间复杂度分析:

- 随机选择算法 RANDOMIZED-SELECT 在最坏情况下运行时间为 $\Theta(n^2)$: 每次划分时极不走运地总是按余下的元素中最大的来进行划分, 即每次都只能去除一个元素:

$$T(n) = T(n-1) + \Theta(n) = \Theta(n^2)$$

- RANDOMIZED-SELECT 随机选择算法保证了在平均情况下非常高效, 因为它避免了最坏情况的出现。通过随机选择主元, 算法能够平衡划分, 使得后续步骤更高效



期望为线性时间的选择算法

- 时间复杂度分析：RANDOMIZED-SELECT 的期望运行时间为 $\Theta(n)$
 - 设算法在一个含有 n 个元素的输入数组 $A[p..r]$ 上的运行时间是一个随机变量，记为 $T(n)$
 - 随机划分过程 RANDOMIZED-PARTITION 等概率地返回任何元素作为主元。因此，对于每一个 k ($1 \leq k \leq n$)，子数组 $A[p..q]$ 有 k 个元素（小于或等于主元）的概率就是 $1/n$ （子数组的大小是等可能分布）
 - 对所有 $k = 1, 2, \dots, n$ ，指示器随机变量 X_k 定义为：
$$X_k = I\{\text{子数组 } A[p..q] \text{ 恰好包含 } k \text{ 个元素}\}$$
假设元素是互异的，有 $E[X_k] = 1/n$



期望为线性时间的选择算法

- 时间复杂度分析：RANDOMIZED-SELECT 的期望运行时间为 $\Theta(n)$
 - 当调用 RANDOMIZED-SELECT 并选择 $A[q]$ 作为主元时，那么或者在子数组 $A[p..q-1]$ 上递归，或者在子数组 $A[q+1..r]$ 上递归
 - 为了评估最大可能的输入数据递归调用所需时间的上界，那么总假定第 i 个元素是在划分中包含较多元素的一边
 - 当 $X_k = 1$ 时，表示需要递归处理的两个子数组的大小分别为 $k-1$ 和 $n-k$ ，得到递归式（Note：指示器随机变量 X_k 恰好在给定的 k 值上取值 1，对其他值都为 0）：

$$\begin{aligned} T(n) &\leq \sum_{k=1}^n X_k \cdot (T(\max(k-1, n-k)) + O(n)) \\ &= \sum_{k=1}^n X_k \cdot T(\max(k-1, n-k)) + O(n) \end{aligned}$$



期望为线性时间的选择算法

- 两边取期望值，得到：

$$\begin{aligned} E[T(n)] &\leq E\left[\sum_{k=1}^n X_k \cdot T(\max(k-1, n-k)) + O(n)\right] \\ &= \sum_{k=1}^n E[X_k \cdot T(\max(k-1, n-k))] + O(n) \quad (\text{by linearity of expectation}) \\ &= \sum_{k=1}^n E[X_k] \cdot E[T(\max(k-1, n-k))] + O(n) \quad (\text{by equation (C.24)}) \\ &= \sum_{k=1}^n \frac{1}{n} \cdot E[T(\max(k-1, n-k))] + O(n) \quad (\text{by equation (9.1)}) . \end{aligned}$$

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

- 如果 n 是偶数，则从 $T(\lceil n/2 \rceil)$ 到 $T(n-1)$ 的每一项在总和中恰好出现两次；
如果 n 是奇数，除了 $T(\lceil n/2 \rceil)$ 出现一次外，其他这些项都会出现两次

- 所以有：
$$E[T(n)] \leq \frac{2}{n} \sum_{k=\lceil n/2 \rceil}^{n-1} E[T(k)] + O(n) .$$



期望为线性时间的选择算法

- 利用代入法证明，猜测： $E[T(n)] = O(n)$
- 假设对满足递归式初始条件的某个常数 c ，有 $E[T(n)] \leq cn$

$$\begin{aligned} E[T(n)] &\leq \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + an \\ &= \frac{2c}{n} \left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lfloor n/2 \rfloor - 1} k \right) + an \\ &= \frac{2c}{n} \left(\frac{(n-1)n}{2} - \frac{(\lfloor n/2 \rfloor - 1)\lfloor n/2 \rfloor}{2} \right) + an \\ &\leq \frac{2c}{n} \left(\frac{(n-1)n}{2} - \frac{(n/2 - 2)(n/2 - 1)}{2} \right) + an \\ &= \frac{2c}{n} \left(\frac{n^2 - n}{2} - \frac{n^2/4 - 3n/2 + 2}{2} \right) + an \\ &= \frac{c}{n} \left(\frac{3n^2}{4} + \frac{n}{2} - 2 \right) + an \\ &= c \left(\frac{3n}{4} + \frac{1}{2} - \frac{2}{n} \right) + an \\ &\leq \frac{3cn}{4} + \frac{c}{2} + an \\ &= cn - \left(\frac{cn}{4} - \frac{c}{2} - an \right). \end{aligned}$$

- 上述最后一个不等式成立需要满足 $(cn/4 - c/2) \geq an$ ，得到 $n(c/4 - a) \geq c/2$ ，即 $c \geq 4a$
- 因此，可以得到以下结论：假设所有元素是互异的，在期望线性时间内，可以找到任意顺序的统计量



第六讲 顺序统计学

内容提要:

- 最小值和最大值
- 期望为线性时间的选择算法
- 最坏情况为线性时间的选择算法



最坏情况为线性时间的选择算法

- 基本思想：类似 RANDOMIZED-SELECT 算法，最坏情况为线性时间的选择算法 SELECT 通过对输入数组来进行递归划分寻找元素，但是 SELECT 算法保证每次对数组的划分是个比较平衡的划分
- SELECT 算法的主要步骤：
 - ① 将输入数组的 n 个元素划分为 $\lfloor n/5 \rfloor$ 组，每组 5 个元素，且至多只有一组由剩下的 $(n \bmod 5)$ 个元素组成
 - ② 寻找 $\lfloor n/5 \rfloor$ 组中每一组的中位数：首先对每一组元素进行插入排序，然后确定每组有序元素的中位数
 - 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 (如果有偶数个中位数，约定 x 是较小的中位数)



最坏情况为线性时间的选择算法

□ SELECT 算法的主要步骤:

- ④ 利用修改过的 PARTITION 版本, 以第③步中找到的中位数的中位数 x 作为主元对输入数组进行划分 (Note: 主元作为 PARTITION 过程的输入参数)。让 k 比划分的低区中的元素数目多 1, 因此主元 x 是第 k 小的元素, 并且有 $n-k$ 个元素在划分的高区
- ⑤ 如果 $i = k$, 则返回 x ; 如果 $i < k$, 则在低区递归调用 SELECT 来找出第 i 小的元素; 如果 $i > k$, 则在高区递归查找第 $i-k$ 小的元素



最坏情况为线性时间的选择算法

- 修改过的 PARTITION 版本：SELECT 算法使用快速排序的确定性划分过程 PARTITION，但是将划分的主元也作为输入参数

```
PARTITION( $A, p, r, x$ )
1  $i = p - 1$ 
2 for  $j = p$  to  $r - 1$ 
3   if  $A[j] \leq x$ 
4      $i = i + 1$ 
5   exchange  $A[i]$  with  $A[j]$ 
6 exchange  $A[i + 1]$  with  $A[r]$ 
7 return  $i + 1$ 
```



最坏情况为线性时间的选择算法

SELECT 算法的伪代码:

```
SELECT(A, p, r, i)
1  if p == r
2      return A[p]
3
4  // 步骤1: 将数组划分为  $\lceil n/5 \rceil$  组, 每组5个元素, 并找到每组的中位数
5  n = r - p + 1
6  将 A[p..r] 划分为  $\lceil n/5 \rceil$  组, 每组5个元素 (最后一组可能少于5个)
7  创建新数组 M // 用于存储各组的中位数
8  for j = 1 to  $\lceil n/5 \rceil$ 
9      start = p + (j-1)*5
10     end = min(p + j*5 - 1, r)
11     对 A[start..end] 进行排序 (例如使用插入排序)
12     M[j] = A[start +  $\lfloor (end-start)/2 \rfloor$ ] // 取中位数
13
14 // 步骤2: 递归调用 SELECT 找到中位数的中位数 x
15 x = SELECT(M, 1,  $\lceil n/5 \rceil$ ,  $\lfloor \lceil n/5 \rceil / 2 \rfloor$ )
16
17 // 步骤3: 使用 x 作为主元进行划分
18 q = PARTITION-AROUND-PIVOT(A, p, r, x)
19 k = q - p + 1 // 计算主元在数组中的顺序统计量
20
21 // 步骤4: 根据比较结果决定递归方向
22 if i == k
23     return A[q] // 找到了第 i 小的元素
24 else if i < k
25     return SELECT(A, p, q-1, i) // 在左半部分递归查找
26 else
27     return SELECT(A, q+1, r, i-k) // 在右半部分递归查找
```



最坏情况为线性时间的选择算法

□ 时间复杂度分析:

- SELECT 算法中大于划分主元 x 的元素个数的下界: 在 $\lceil n/5 \rceil$ 个组中, 除了当 n 不能被 5 整除时产生的所含元素少于 5 的那个组和包含 x 的那个组之外, 至少有一半的组中有 3 个元素大于 x 。

记作:

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

- 同理, 至少有 $3n/10-6$ 的元素小于 x
- 如果 PARTITION 过程 $i \neq k$, 在最坏情况下, 在第 5 步中 SELECT 算法递归调用最多作用于 $7n/10+6$ 个元素





最坏情况为线性时间的选择算法

□ 时间复杂度分析:

- SELECT 算法中步骤 ①、② 和 ④ 需要 $O(n)$ 时间。步骤 ③ 所需时间为 $T(\lceil n/5 \rceil)$, 步骤 ⑤ 所需时间至多为 $T(7n/10+6)$:

$$T(n) \leq \begin{cases} O(1) & \text{若 } n < 140 \\ T(\lceil n/5 \rceil) + T(7n/10+6) + O(n) & \text{若 } n \geq 140 \end{cases}$$

- 假设任何少于 140 个元素的输入需要 $O(1)$ 时间
- 用代入法证明这个运行时间是线性的 $T(n)=O(n)$, 即 $T(n) \leq cn$

$$\begin{aligned} T(n) &\leq c\lceil n/5 \rceil + c(7n/10+6) + an \\ &\leq cn/5 + c + c7n/10 + 6c + an \\ &= 9cn/10 + 7c + an \\ &= cn + (-cn/10 + 7c + an) \\ &\leq cn, \quad \text{if } -cn/10 + 7c + an \leq 0! \end{aligned}$$

- 上述不等式等价于 $c \geq 10a(n/(n-70))$ 。假设 $n \geq 140$ 时, $n/(n-70) \leq 2$ 。因此, 选择 $c \geq 20a$, 上式就可以成立!



最坏情况为线性时间的选择算法

□ 总结:

- RANDOMIZED-SELECT 算法和 SELECT 算法也是通过元素间的比较来确定它们之间的相对次序的
- 在基于比较运算的模型中，在最坏情况下，排序算法需要 $\Omega(n \lg n)$ 时间
- 本章中的线性时间选择算法不需要任何关于输入的假设，也不受限于 $\Omega(n \lg n)$ 的下界约束，因为它们没有使用排序就解决了选择问题



谢谢!