# 算法设计与分析

主讲人：吴庭芳

Email: *tfwu@suda.edu.cn*

苏州大学 计算机学院







# 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（互异的）数的集合 A 和一个整数 $i$ （$1\leq i \leq n$）
  - 输出：元素 $x\in$ A，且 A 中恰好有 $i$-1 个其他的元素小于 $x$
- **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)
min = A[1]
for  i = 2 to A.length
     if min > A[i]
           min =  A[i]
return min
```

可以给出上述最小值算法的比较次数的上界: $n$-1 次

- 那 $n$-1 是最少的比较次数吗?

# 最小值和最大值

- 对于确定最小值问题，可以得到其下界就是 $n$-1 次比较
- 锦标赛算法：对于任意一个确定最小值的算法，可以把它看成是在各元素之间进行的一场锦标赛，每次比较都是锦标赛中的一场比赛，两个元素中较小的获胜
  - 除了最终的获胜者之外，其他每个元素都至少要输掉一场比赛
  - 为了得到最终的胜者（最小值），必须要做 $n$-1 次比较
  - 因此，从所执行的比较次数来看，算法 MINIMUM 是最优的
- 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?
  - 如果分别独立寻找其中的最小值和最大值，则各需要做 $n$-1 次比较，共需要 2$n$-2 次比较

# 最小值和最大值

- **同时寻找最小值和最大值**
  - 记录比较过程中遇到的最小值和最大值
  - **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）

```
MAX-MINIMUM (A)
if A.length is odd
        min = A[1]
        max = A[1]
        i = 2
    while i ≤ A.length
            min = MIN(MIN(A[i], A[i+1]), min)
            max = MAX(MAX(A[i], A[i+1]), max)
            i = i+2
    end
else  min = MIN(A[1], A[2])
      max = MAX(A[1], A[2])
      i = 3
      …
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，那么总共做了 $3\lfloor n/2 \rfloor$ 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：$3\lfloor n/2 \rfloor$







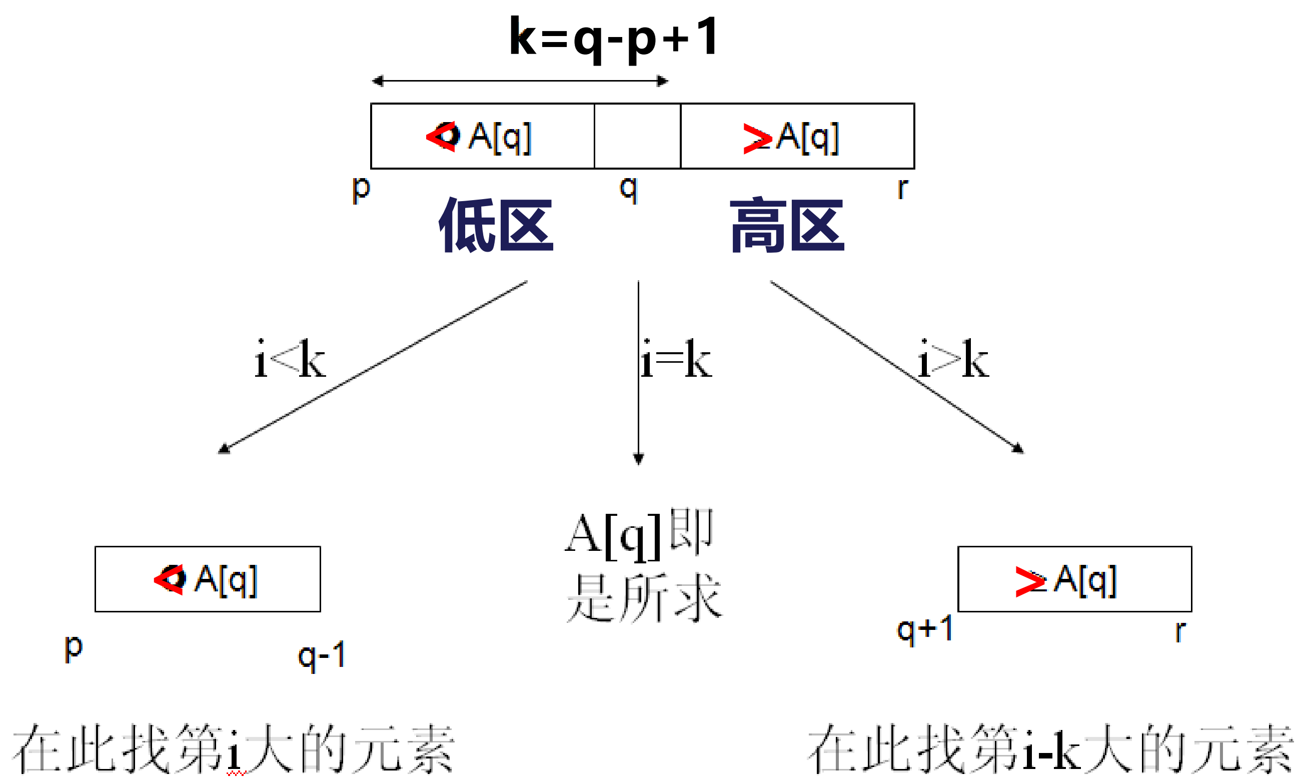
# 第六讲 顺序统计学

## 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，对输入数组进行递归划分**
  - **解决：但是与快速排序算法不同的是，而随机选择算法只递归处理划分的一边**



# 期望为线性时间的选择算法

- **RANDOMIZED-SELECT** 利用 RANDOMIZED-PARTITION 过程，随机选择算法的部分行为是由随机数生成器的输出决定的。RANDOMIZED-SELECT 的伪代码如下：

```
RANDOMIZED-SELECT (A, p, r, i)
  if  p = r                                  //边界问题处理
      return A[p]
   q = RANDOMIZED-PARTITION(A, p, r)   // 进行划分，返回划分主元下标
   k = q – p + 1                 //主元元素是第 k 个顺序统计量
   if  i == k                    //主元元素就是需要返回的数值
       return A[q]
  else if  i < k                 //第 i 个顺序统计量落在划分的低区
      return RANDOMIZED-SELECT (A, p, q-1, i)
  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 的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：

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

  - 因为 RANDOMIZED-SELECT 是随机化的，所以不存在一个特定的输入数据会导致其最坏情况发生

# 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 $k$ ($1 \le k \le n$)，子数组 A[$p..q$] 有 $k$ 个元素（全部小于或等于主元）的概率就是 $1/n$
  - 对所有 $k = 1$，2，…，$n$，指示器随机变量 $X_k$ 为：

$$X_k = \mathrm{I}\{子数组 A[p..q] 恰好包含\ k\ 个元素\}$$

假设元素是互异的，有 $\mathrm{E}[X_k] = 1/n$

# 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
  - 评估最大可能的输入数据递归调用所需时间，给出递归调用所需时间的上界，假定第 *i* 个元素总是在划分中包含较多元素的一边
  - 当 $X_k = 1$ 时，可能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（N：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）：

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

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

# 期望为线性时间的选择算法

- 两边取期望值，得到：

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

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

- 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($\lfloor n/2 \rfloor$) 出现一次外，其他这些项都会出现两次

- 所以有： $\mathrm{E}[T(n)] \le \dfrac{2}{n} \displaystyle\sum_{k=\lfloor n/2 \rfloor}^{n-1} \mathrm{E}[T(k)] + O(n)$ .

# 期望为线性时间的选择算法

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

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

- 上述最后一个不等式成立需要满足 $(cn/4-c/2) \ge an$，得到 $n(c/4-a) \ge c/2$，即 $c \ge 4a$
- **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找出所求元素，但是 SELECT 算法保证每次对数组的划分是个较好的划分**
- **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**
- **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 $\lceil n/5 \rceil$ 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

# 最坏情况为线性时间的选择算法

- **主要步骤：**

③ 对第 2 步中找出的 $\lceil n/5 \rceil$ 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，为了方便，约定 $x$ 是较小的中位数）

④ 利用修改过的 PARTITION 版本，以中位数的中位数 $x$ 作为主元对输入数组进行划分（这里，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 $x$ 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

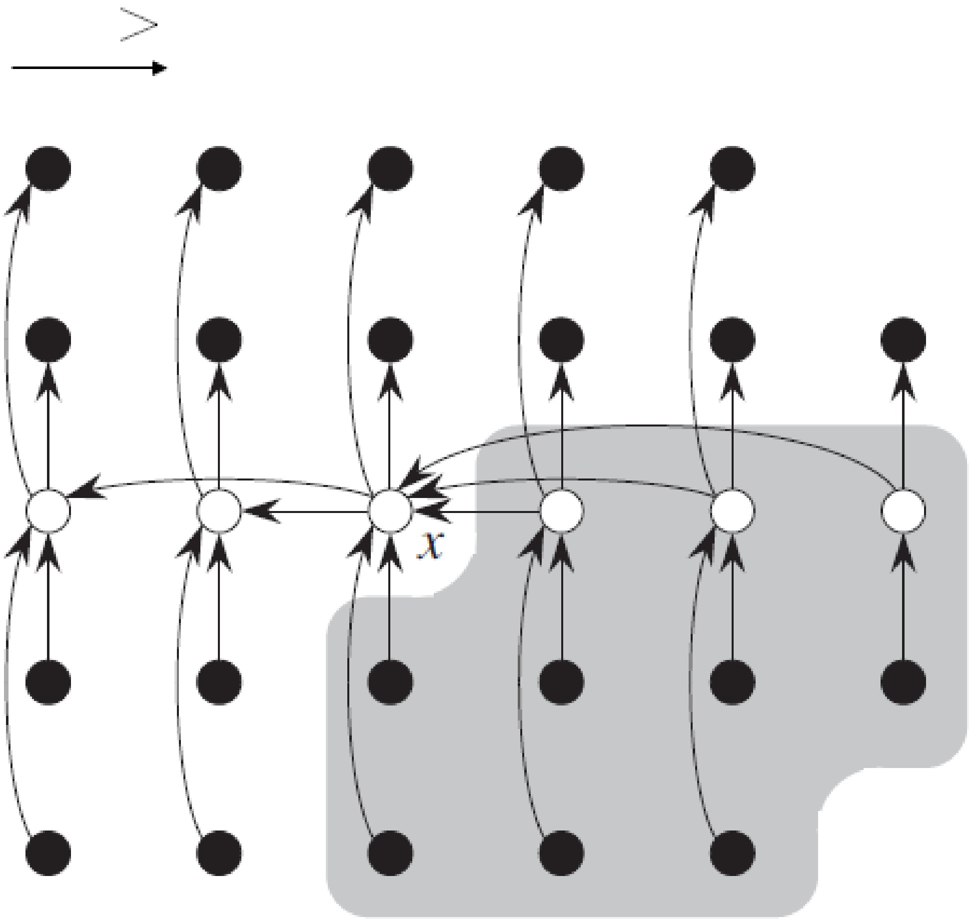
# 最坏情况为线性时间的选择算法

- **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：

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

  - 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



# 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(⌈$n$/5⌉)，步骤 5 所需时间至多为 T(7$n$/10+6)

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

  - 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，即 T($n$) ≤ $cn$

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

  - 上述不等式等价于 $c \ge 10a(n/(n\text{-}70))$。假设 $n \ge 140$ 时，$n/(n\text{-}70) \le 2$。因此，选择 $c \ge 20a$，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，而线性时间排序算法则需要在输入上做一些假设
  - 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！

## Q & A

**作业：9.1-1**

**9.2-2**

**9.3-1**