



# 算法设计与分析

Computer Algorithm Design & Analysis  
2022.3

何琨

[brooklet60@hust.edu.cn](mailto:brooklet60@hust.edu.cn)

群 号 : 638562638

# 关于排序

排序主要参见数据结构课程和教材第6~8章相关内容。

## 第6章 堆排序

- 1、堆的定义及堆的操作：HEAPIFY、建堆
- 2、堆排序的基本思想和分析
- 3、优先队列（6.5, ★）

**优先队列**（Priority Queue）：是一种用来维护由一组元素构成的集合S的**数据结构**，其中的每一个元素都有一个相关的值，称为**关键字**（key）。优先队列有最大优先队列和最小优先队列。

## 一个最大优先队列支持以下操作：

- INSERT( $S, x$ )：把元素 $x$ 插入集合 $S$ 中，等价于 $S = S \cup \{x\}$ 。
- MAXIMUM( $S$ )：返回 $S$ 中具有最大关键字值的元素。
- EXTRACT-MAX( $S$ )：去掉并返回 $S$ 中的最大关键字值的元素。
- INCREASE-KEY( $S, x, k$ )：将元素 $x$ 的关键字的值增加到 $k$ （这里假设 $k$ 的值不小于 $x$ 的原来的关键字的值）。

## 优先队列的实现：

- 线性队列
- 堆

## 优先队列有很多应用：

- C++类库里的优先队列
- 实际应用场合：作业调度管理等。

# 第8章 线性时间的排序算法

- 计数排序
- 基数排序
- 桶排序

## ◆ 以比较为基础的排序算法的时间下界

**以比较为基础的排序**：只使用比较运算来决定元素之间的大小关系并调整其位置，不做改变元素值大小等其它操作的排序算法

# 以关键字比较为基础的排序算法的时间下界

——最坏情况下的时间下界为： $\Omega(n \log n)$

假设参加排序的 $n$ 个关键字 $A(1), A(2), \dots, A(n)$ 互异。任意两个关键字的比较必导致 $A(i) < A(j)$  或  $A(i) > A(j)$  的结果。

利用**二元比较树**描述元素间的比较过程：

◆ 二元比较树是一个**决策树**，它表示某一特定排序算法对所有元素的比较操作过程。

◆ 内结点代表一对元素的比较。

◆ 叶子结点代表某种特定输入下经过算法排序后的排列。

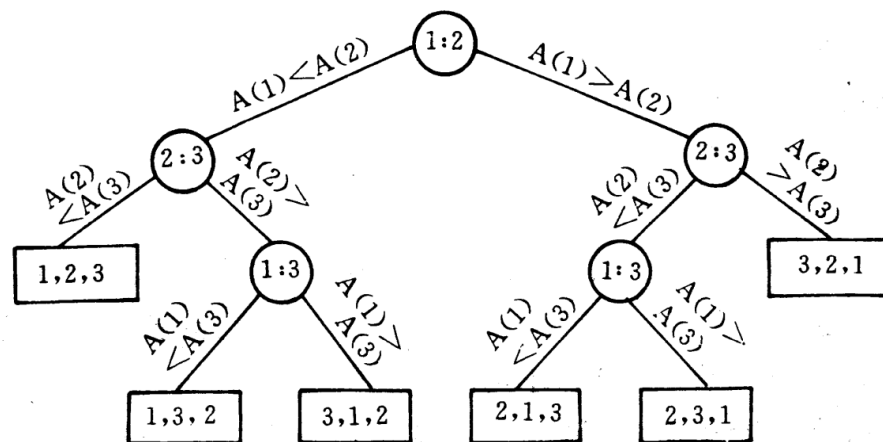
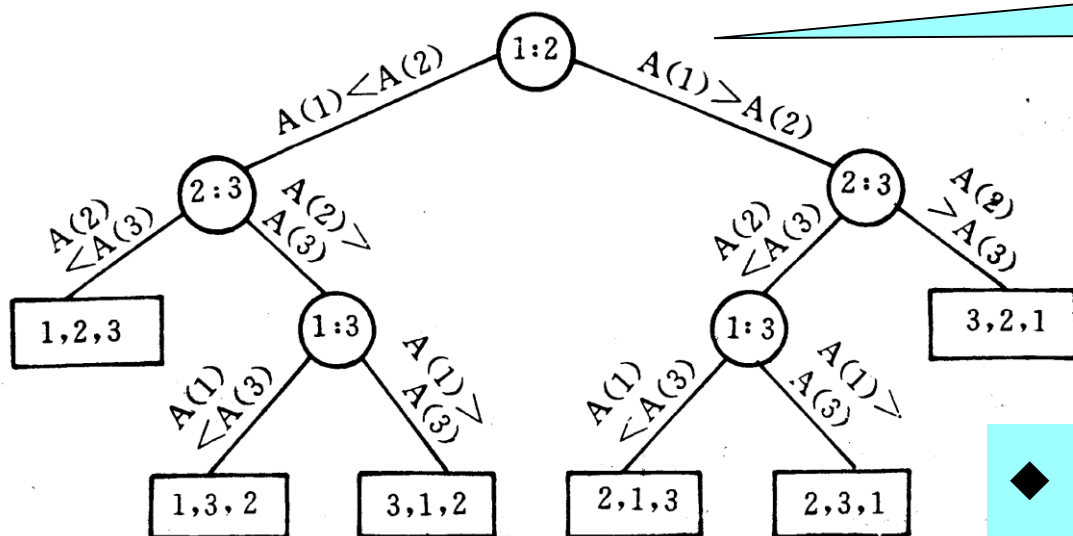


图 2.6 对三个关键字分类的比较树



初始集合中的元素因大小不同，将导致排序过程走不同的分支

图 2.6 对三个关键字分类的比较树

- ◆ 若  $A(i) < A(j)$ ，进入下一级的左分支
- ◆ 若  $A(i) > A(j)$ ，进入下一级的右分支

算法在外部结点终止。

**最长的路径**代表算法在最坏情况下的执行情况，该路径的长度即是算法在最坏情况下所作的比较次数。

故，以比较为基础的排序算法的**最坏情况下界**等于二元比较树的最小高度。

① 由于 $n$ 个关键字有 $n!$ 种可能的排列，所以分类的二元比较树中将有 $n!$ 个外部结点。

② 设一棵二元比较树的所有内结点的级数均小于或等于 $k$ ，则该树中最多有 $2^k$ 个外结点。

记算法在最坏情况下所作的比较次数为 $T(n)$ ，则有

$$T(n) = k$$

—— 生成外结点所代表的分类序列所需的比较次数等于该外结点所在的级数-1；

则，根据①和②的分析，有：

$$n! \leq 2^{T(n)}$$

化简：

$$\text{当 } n > 1 \text{ 时，有 } n! \geq n(n-1)(n-2) \cdots \left( \lceil n/2 \rceil \right) \geq (n/2)^{n/2}$$

当  $n \geq 4$  时，两边取对数，有

$$T(n) \geq (n/2) \log(n/2) \geq (n/4) \log n$$

故，任何以比较为基础的排序算法的最坏情况的时间下界  
为： $\Omega(n \log n)$



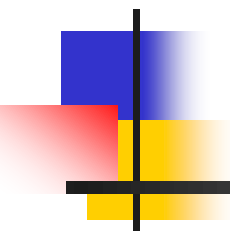
定理 设 $A(1:n)$ 含有  $n(n \geq 1)$  个不同的元素, 且有

$$A(1) < A(2) < \cdots < A(n)。$$

以以比较为基础的算法去判断给定的 $x$ 是否有  $x \in A(1:n)$  ,  
则, 最坏情况下, 任何这样的算法所需的最小比较次数 $FIND(n)$   
有:

$$FIND(n) \geq \lceil \log(n+1) \rceil$$

- 1) 任何一种以比较为基础的有序检索算法, 在最坏情况下的计算时间都不低于  $O(\log n)$ 。因此, 不可能存在最坏情况比二分检索数量级还低的算法。
- 2) 二分检索是解决检索问题的最优的最坏情况算法



# Chapter 9

## Medians and Order Statistics

---

中位数和顺序统计量

# 基本概念：

1) **顺序统计量**：在一个由 $n$ 个元素组成的集合中，第 $i$ 个顺序统计量(order statistic)是该集合中的第 $i$ 小的元素。

如：在一个元素集合中，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 ( $i=n$ )。

2) **中位数**：对一个有 $n$ 个元素的集合，将数据排序后，位置在最中间的数称为该集合的**中位数**。

➤ 当元素数为**奇数**时，中位数出现在 $i=(n+1)/2$ 处；

如：1、2、3、6、7的中位数是3。

➤ 当元素数为**偶数**时，中位数取作第 $n/2$ 个数据与第 $n/2+1$ 个数据的**算术平均值**。

如：1、2、3、5的中位数是2.5。

- 当元素数为偶数时，也可视为存在两个中位数，分别出现在 $i=n/2$  (称为下中位数) 和 $i=n/2+1$  (称为上中位数) 处。

如： 1、2、3、5的下中位数是2，上中位数是3。

- 一般情况下，不管元素数是偶数或奇数，可以用下式计算：

➤ 下中位数：  $i = \lfloor (n+1)/2 \rfloor$  ,

➤ 上中位数：  $i = \lceil (n+1)/2 \rceil$  注：实际中多取下中位数

如： 1) 1、2、3、6、7的中位数是3。

$$\lfloor (5+1)/2 \rfloor = \lceil (5+1)/2 \rceil = 3$$

2) 1、2、3、5的下中位数是2，上中位数是3。

$$\text{下中位数: } \lfloor (4+1)/2 \rfloor = 2$$

$$\text{上中位数: } \lceil (4+1)/2 \rceil = 3$$

选择问题：从 $n$ 个元素的集合中选择第 $i$ 个顺序统计量的问题形式化地归结为“选择问题”。

- 假设集合中的元素是互异的（可推广至包含重复元素的情形）。

输入：一个包含 $n$ 个（互异）元素的集合 $A$ 和一个整数 $i$ ， $1 \leq i \leq n$ 。

输出：元素 $x \in A$ ，且 $A$ 中恰好有 $i-1$ 个其他元素小于它。

# How to do ?

## 1 ) 排序

元素集合排序后，位于第 $i$ 位的元素即为该集合的第 $i$ 个顺序统计量。

时间复杂度： $O(n \log n)$

## 2 ) 选择算法

设法找出元素集合里面的第 $i$ 小元素，该元素为集合的第 $i$ 个顺序统计量。

时间复杂度： $O(n)$

## 9.1 最小值和最大值（略）

## 9.2 期望为线性时间的选择算法

### ■ 借助QUICKSORT的PARTITION过程

PARTITION( $A, p, r$ )

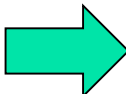
```
1  $x = A[r]$ 
2  $i = p - 1$ 
3 for  $j = p$  to  $r - 1$ 
4     if  $A[j] \leq x$ 
5          $i = i + 1$ 
6     exchange  $A[i]$  with  $A[j]$ 
7 exchange  $A[i + 1]$  with  $A[r]$ 
8 return  $i + 1$ 
```

QUICKSORT( $A, p, r$ )

```
1 if  $p < r$ 
2      $q = \text{PARTITION}(A, p, r)$ 
3     QUICKSORT( $A, p, q - 1$ )
4     QUICKSORT( $A, q + 1, r$ )
```

随机化的PARTITION过程

RANDOMIZED-PARTITION( $A, p, r$ )



```
1  $i = \text{RANDOM}(p, r)$ 
2 exchange  $A[r]$  with  $A[i]$ 
3 return PARTITION( $A, p, r$ )
```

RANDOMIZED-QUICKSORT( $A, p, r$ )

```
1 if  $p < r$ 
2      $q = \text{RANDOMIZED-PARTITION}(A, p, r)$ 
3     RANDOMIZED-QUICKSORT( $A, p, q - 1$ )
4     RANDOMIZED-QUICKSORT( $A, q + 1, r$ )
```



2) 利用RANDOMIZED-PARTITION设计一个较低时间复杂度的算法找集合中的第*i*小元素

**PARTITION(1,n)** : 设主元素*v*被放在位置A(*j*)上。

此时,

- 若*i=j*, 则A(*j*)即是第*i*小元素; 否则,
- 若*i<j*, 则A(1:n)中的第*i*小元素将出现在A(1:*j*-1)中;
- 若*i>j*, 则A(1:n)中的第*i*小元素将出现在A(*j*+1:n)中。

# 利用RANDOMIZED-PARTITION实现选择算法

在 $A[p, r]$ 中找第 $i$ 小元素的算法:

```
RANDOMIZED-SELECT( $A, p, r, i$ )
1  if  $p == r$ 
2      return  $A[p]$ 
3   $q = \text{RANDOMIZED-PARTITION}(A, p, r)$ 
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( $A, q + 1, r, \underline{i - k}$ )
```

■ RANDOMIZED-SELECT的最坏情况运行时间是 $O(n^2)$

➤ **最坏情况下的特例**: 输入 $A$ 恰好使对RANDOMIZED-PARTITION的第 $j$ 次调用选中的主元素是第 $j$ 小元素, 而 $i=n$ 。

- RANDOMIZED-SELECT的期望运行时间是 $O(n)$ 。

证明：

设算法的运行时间是一个随机变量，记为 $T(n)$ 。

设RANDOMIZED-PARTITION( $A, p, r$ )可以等概率地返回任何元素作为主元。即，对每个 $k$  ( $1 \leq k \leq n$ )，划分后区间 $A[p, q]$ 恰好有 $k$ 个元素（全部小于或等于主元）的概率是 $1/(r-p+1)$ 。

对所有 $k=1, 2, \dots, n$ ，定义指示器随机变量 $X_k$ ：

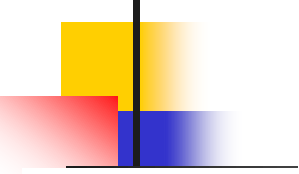
$$X_k = I\{\text{子数组 } A[p..q] \text{ 正好包含 } k \text{ 个元素}\}$$

假设 $A$ 中元素是互异的，则有  $E[X_k] = 1/n$

## ■ 期望上界分析

- RANDOMIZED - SELECT当前处理中， $A[q]$ 是主元。若 $i=q$ ，则得到正确答案，结束过程。否则在 $A[p, q-1]$ 或 $A[q+1, r]$ 上递归。
- 对一次给定的RANDOMIZED - SELECT调用，若主元素恰好落在给定的 $k$ 值，则指示器随机变量 $X_k$ 值为1，否则为0。
- 设 $T(n)$ 是单调递增的。
  - ▣ 为了分析递归调用所需时间的上界，我们设每次划分都有：(很不幸地)第 $i$ 个元素总落在元素数较多的一边。
  - ▣ 当 $X_k=1$ 时，若需递归，两个子数组的大小分别为 $k-1$ 和 $n-k$ ，算法只在其中之一、并设是在较大的子数组上递归执行。

则有以下递归式：



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

■ 两边取期望：

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

注：公式C. 24的应用依赖于 $X_k$ 和 $T(\max(k-1, n-k))$ 是独立的随机变量。见习题9. 2-2

这里，

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

在 $k=1 \sim n$ 的区间里，表达式 $T(\max(k-1, n-k))$ 有：

- 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，则 $T(\lceil n/2 \rceil)$ 出现一次，从 $T(\lceil \frac{n}{2} \rceil + 1)$ 到 $T(n-1)$ 各项在总和中出现两次；

则有：

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

## 代换法证明：E[T(n)]=O(n).

- 即证明：存在常数c，使得E[T(n)] ≤ cn。
- 将上述猜测代入推论证明阶段有：

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

- 这里，为了证明E[T(n)] ≤ cn，须有cn/4 - c/2 - an ≥ 0。
- 什么样的c能满足？

■ (续:  $cn/4 - c/2 - an \geq 0$  何时成立? )

即要求有:  $n(c/4 - a) \geq c/2$

选取常数  $c$ , 使得  $(c/4 - a) > 0$ , 两边同除  $(c/4 - a)$ , 则有

$$n \geq \frac{c/2}{c/4 - a} = \frac{2c}{c - 4a}.$$

因此, 当  $n \geq 2c/(c - 4a)$  时, 对任意的  $n$  有  $E[T(n)] \leq cn$ , 即

$E[T(n)] = O(n)$  成立。

➤  $n < 2c/(c - 4a)$  时, 可假设  $T(n) = O(1)$ 。

**结论:** 若所有元素互异, 则可在线性期望时间内, 找到任意顺序统计量。



## 9.3 最坏情况是 $O(n)$ 的选择算法

- 1) 造成最坏情况是 $O(n^2)$ 的原因分析：类似快速排序的最坏情况
- 2) 采用两次取中间值的规则精心选取划分元素

**目标**：精心选择划分元素，避免随机选取可能出现的极端情况。

**分三步**：

首先，将参加划分的 $n$ 个元素分成 $\lfloor n/r \rfloor$ 组，每组有 $r$ 个元素( $r \geq 1$ )。

(多余的  $n - r\lfloor n/r \rfloor$  个元素忽略不计)

然后，对这  $\lfloor n/r \rfloor$  组每组的 $r$ 个元素进行排序并找出其中间元素 $m_i$ ,

$1 \leq i \leq \lfloor n/r \rfloor$ ，共得 $\lfloor n/r \rfloor$ 个中间值（中位数）。

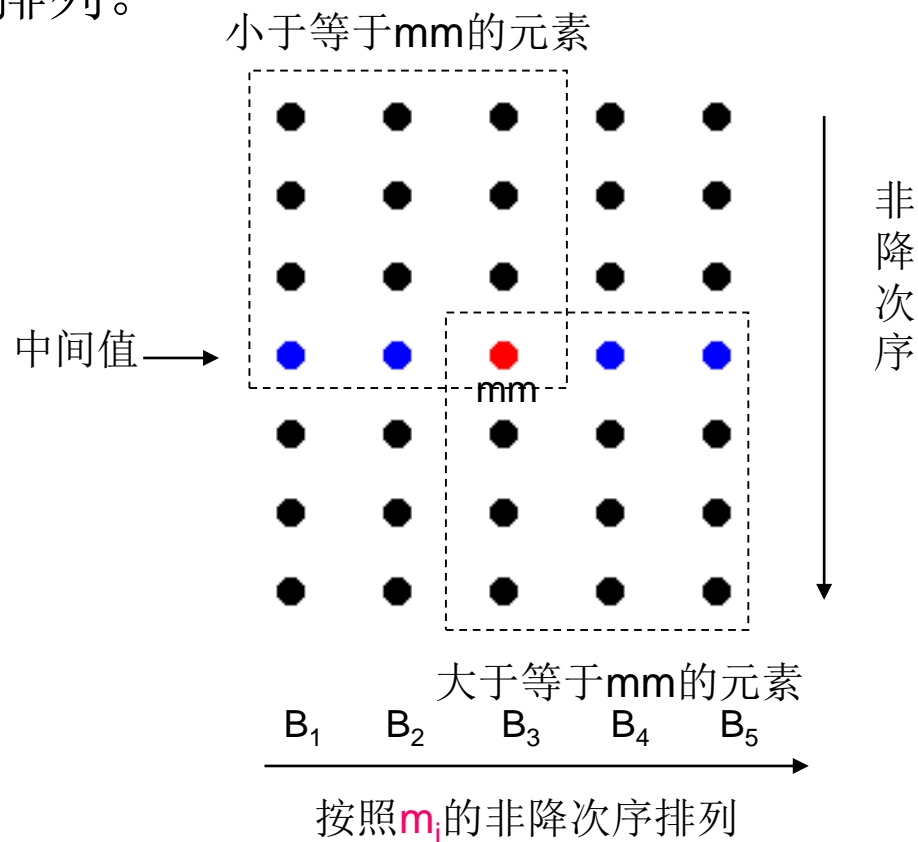
再后，对这  $\lfloor n/r \rfloor$  个中间值查找，再找出其中间值 $mm$ （中位数）。

最后，将 $mm$ 作为划分元素执行划分。

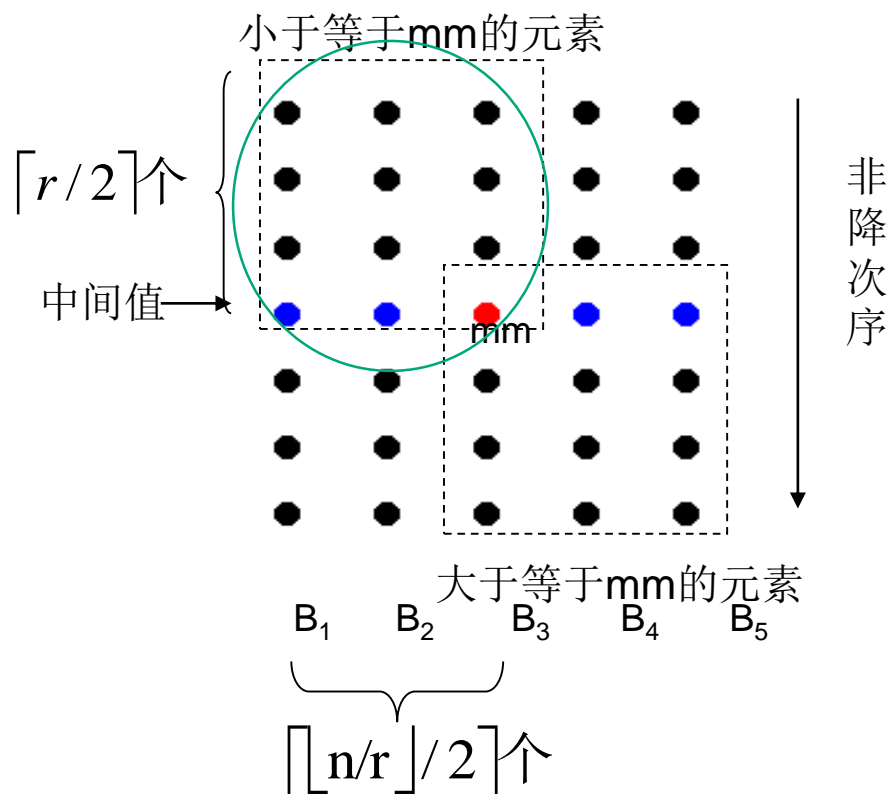
例：设  $n=35$ ,  $r=7$ 。

- 分为  $n/r = 5$  个元素组： $B_1, B_2, B_3, B_4, B_5$ ；
- 每组有7个元素。
- $B_1$ - $B_5$ 按照各组的 $m_i$ 的非降次序排列。
- $mm = m_i$ 的中间值,  $1 \leq i \leq 5$

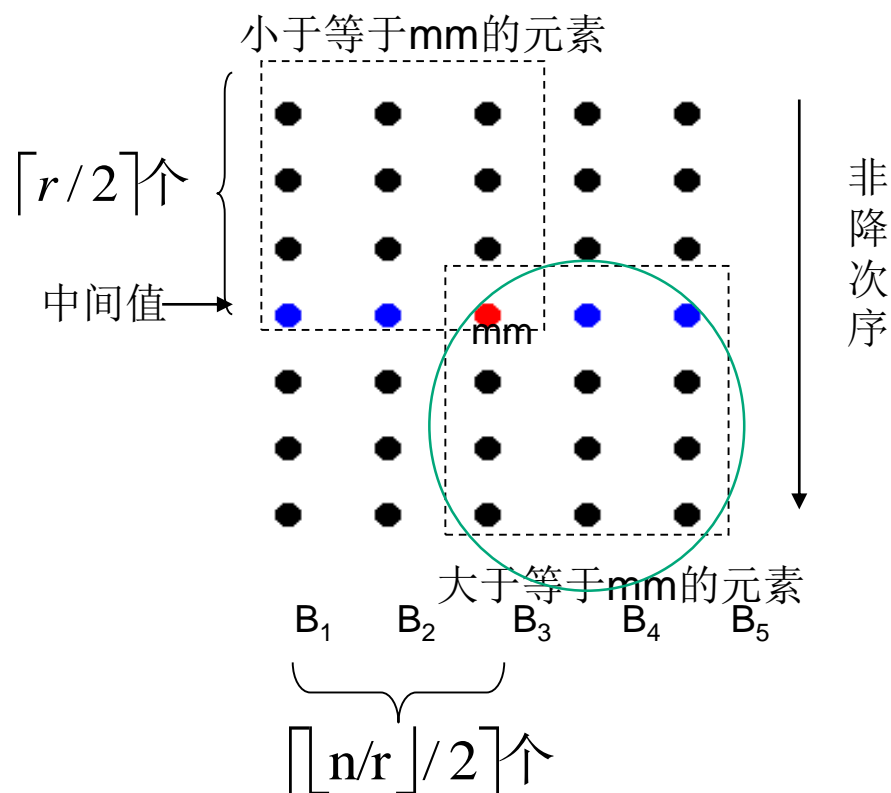
由图所示有：



故，至少有  $\lceil r/2 \rceil \lfloor n/r \rfloor / 2$  个元素小于或等于  $mm$ 。



同理，也至少有  $\lceil r/2 \rceil \lfloor n/r \rfloor / 2$  个元素大于或等于  $mm$ 。



以 $r=5$ 为例。使用两次取中间值规则来选择划分元素 $v$  (即 $mm$ )。

可得到,

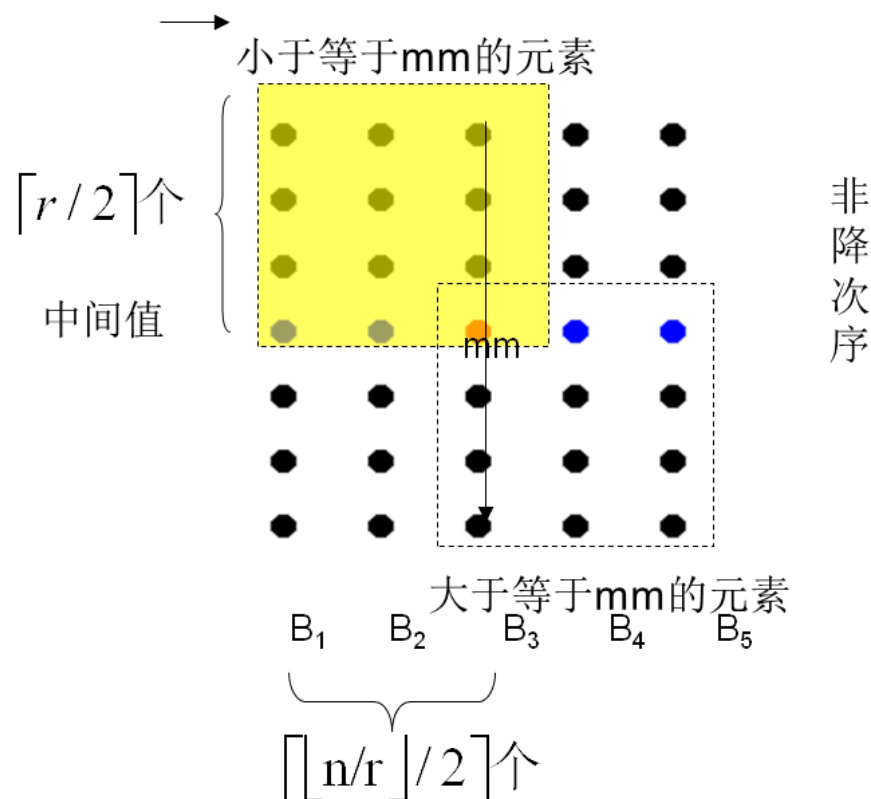
- ◆ 至少有  $1.5\lfloor n/5 \rfloor$  个元素小于或等于选择元素 $v$
- ◆ 且至多有  $n - 1.5\lfloor n/5 \rfloor \leq 0.7n + 1.2$  个元素大于等于 $v$

$$\begin{aligned} & \lceil r/2 \rceil \lfloor \lfloor n/r \rfloor / 2 \rfloor \\ &= \lceil 5/2 \rceil \lfloor \lfloor n/5 \rfloor / 2 \rfloor \\ &\geq 3 \lfloor n/5 \rfloor / 2 = 1.5 \lfloor n/5 \rfloor \end{aligned}$$

$$\begin{aligned} & n - 1.5 \lfloor n/5 \rfloor \\ &\leq n - 1.5(n - 4) / 5 \\ &= 0.7n + 1.2 \end{aligned}$$

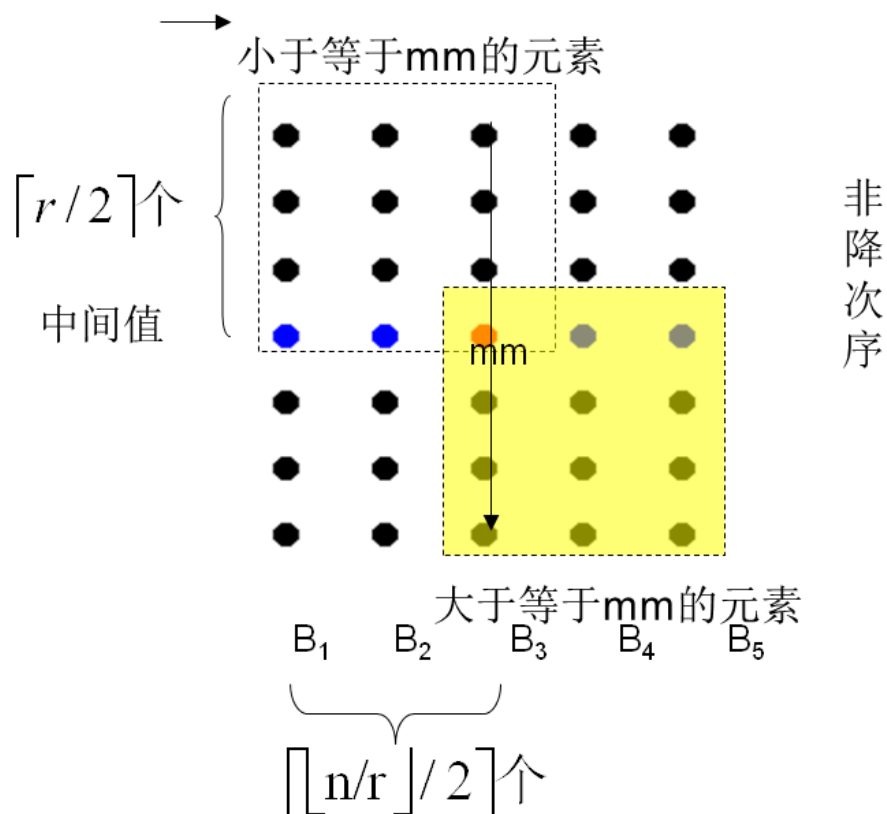
$$\text{注: } \lfloor n/5 \rfloor \geq (n - 4) / 5$$

算法导论:  $0.7n+6$



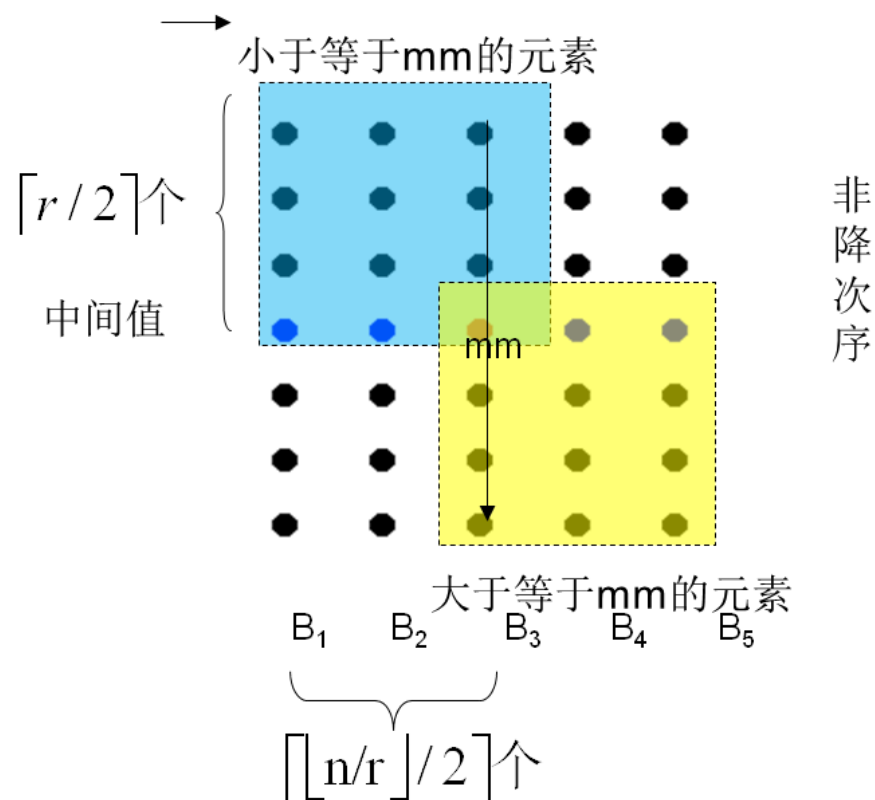
同理,

- ◆ 至少有  $1.5\lfloor n/5 \rfloor$  个元素大于或等于选择元素  $v$
- ◆ 且至多有  $n - 1.5\lfloor n/5 \rfloor \leq 0.7n + 1.2$  个元素小于等于  $v$



故,

这样的 $v$ 可较好地划分 $A$ 中的 $n$ 个元素：比**足够多的元素大**，也比**足够多的元素小**。则，不论落在那个区域，总可以在下一步查找前舍去足够多的元素，而在剩下的“较小”范围内继续查找。



## 2) 算法描述

算法 使用二次取中规则的选择算法的说明性描述

Procedure SELECT2(A, i, n)

//在集合A中找第i小元素

① 若 $n \leq r$ , 则采用插入排序法直接对A分类并返回第i小元素。否则

② 把A分成大小为r的 $\lfloor n/r \rfloor$ 个子集合, 忽略多余的元素

③ 设 $M = \{m_1, m_2, \dots, m_{\lfloor n/r \rfloor}\}$ 是 $\lfloor n/r \rfloor$ 个子集合的中间值集合

④  $v \leftarrow \text{SELECT2}(M, \lceil \lfloor n/r \rfloor / 2 \rceil, \lfloor n/r \rfloor)$

⑤  $j \leftarrow \text{PARTITION}(A, v)$

⑥ case

:i=j: return(v)

:i<j: 设S是A(1:j-1)中元素的集合; return(SELECT2(S, i, j-1))

:else: 设R是A(j+1:n)中元素的集合; return(SELECT2(R, i-j, n-j))

endcase

end SELECT2



SELECT2的时间分析：注，由于r为定值，所以这里视对r个元素的直接排序的时间为“定值”  $O(1)$ 。

故有，

$$T(n) = \begin{cases} cn & n < 24, \\ T(n/5) + T(3n/4) + cn & n \geq 24 \end{cases}$$

用归纳法(代入法)可证：

$$T(n) \leq 20cn$$

故，在r=5地情况下，求解n个不同元素选择问题的算法

SELECT2的最坏情况时间是  $O(n)$ 。

进一步分析:

若A中有相同的元素时, 上述结论 $T(n) = O(n)$ 可能不成立。

原因:

步骤⑤经PARTITION调用所产生的S和R两个子集合中可能存在一些元素等于划分元素 $v$ , 可能导致 $|S|$ 或 $|R|$ 大于 $0.7n+1.2$ , 从而影响到算法的效率。

例如: 设 $r=5$ , 且A中有相同元素。不妨假设其中有 $0.7n+1.2$ 个元素比 $v$ 小, 而其余的元素都等于 $v$ 。

则, 经过PARTITION, 这些等于 $v$ 的元素中至多有一半可能在落在S中, 故 $|S| \leq 0.7n+1.2 + (0.3n-1.2)/2 = 0.85n+0.6$ 。

同理,  $|R| \leq 0.85n+0.6$ 。

可得, 此时步骤④和⑥所处理的元素总数将是

$$T(n/5) + T(0.85n+0.6) \approx 1.05n+0.6 > n$$

不再是线性关系。故有 $T(n) \neq O(n)$

改进：

方法一：将A集合分成3个子集合U, S和R，其中U是由A中所有与v相同的元素组成，S是由A中所有比v小的元素组成，R则是A中所有比v大的元素组成。

同时步骤⑥更改：

```
case
  :  $|S| \geq k$ : return (SELECT2(S, k, |S|))
  :  $|S| + |U| \geq k$ : return (v)
  : else: return (SELECT2(R,  $k - |S| - |U|$ , |R|))
endcase
```

从而保证  $|S|$  和  $|R| \leq 0.7n + 1.2$  成立，故关于T(n)的分析仍然成立。

即  $T(n) = O(n)$

## 方法二：选取其它的r值进行计算

取 $r=9$ 。重新计算可得，此时将有 $2.5\lfloor n/9 \rfloor$ 个元素小于或等于 $v$ ，同时至少有 $2.5\lfloor n/9 \rfloor$ 大于或等于 $v$ 。

则 当 $n \geq 90$ 时， $|S|$ 和 $|R|$ 都至多为

$$n - 2.5\lfloor n/9 \rfloor + \frac{1}{2}(2.5\lfloor n/9 \rfloor) = n - 1.25\lfloor n/9 \rfloor \leq 31n/36 + 1.25 \leq 63n/72$$

基于上述分析，有新的递推式：

$$T(n) = \begin{cases} c_1 n & n < 90 \\ T(n/9) + T(63n/72) + c_1 n & n \geq 90 \end{cases}$$

用归纳法可证：

$$T(n) \leq 72c_1 n$$

即， $T(n) = O(n)$ 成立

## 4 ) SELECT2的实现

### 算法中需要解决的两个问题

1) 如何求子集合的中间值？

当 $r$ 较小时，采用INSERTIONSORT直接对每组的 $r$ 个元素排序，在排序好的序列中，中间下标位置所对应的元素即为本组中间元素。

2) 如何保存 $\left\lfloor \frac{n}{r} \right\rfloor$ 个子集合的中间值？

在各组找到中间元素后，将其调整到数组A的前部，按子集合的顺序关系连续保存。从而可方便用递归调用的方式对这些中间值进行二次取中，找出中间值的中间值。

### 算法3.11 SELECT2算法的实现

procedure SEL(A, m, p, k)

//返回一个i, 使得 $i \in [m, p]$ , 且A(i)是A(m:p)中第k小元素, r是一个全程变量, 其取值为大于1的整数

global r; integer n, i, j

loop

if  $p-m+1 \leq r$  then call INSERTIONSORT(A, m, p); return (m+k-1); endif

$n \leftarrow p-m+1$  //元素数//

for  $i \leftarrow 1$  to  $\lfloor n/r \rfloor$  do //计算中间值//

call INSERTIONSORT(A,  $m+(i-1)*r$ ,  $m+i*r-1$ ) //将中间值收集到A(m:p)的前部//

call INTERCHANGE(A( $m+i-1$ ), A( $m+(i-1)r + \lfloor r/2 \rfloor - 1$ )))

repeat

$j \leftarrow \text{SEL}(A, m, m + \lfloor n/r \rfloor - 1, \lceil \lfloor n/r \rfloor / 2 \rceil)$  //mm//

call INTERCHANGE(A(m), A(j)) //产生划分元素, 将之调整到第一个元素//

$j \leftarrow p+1$

call PARTITION(m, j)

case

:  $j-m+1=k$ : return(j)

:  $j-m+1 > k$ :  $p \leftarrow j-1$

: else:  $k \leftarrow k - (j-m+1)$ ;  $m \leftarrow j+1$

endcase

repeat

end SEL

## 第三次作业：

习题：9.1-1、9.3-5、9-2

### 思考题

( 1 ) 9.3-1

( 2 ) 9.3-9 ( 见课件 )

( 3 ) 分金币 ( Spreading the Wealth, UVa 11300 )

圆桌旁坐着 $n$ 个人，每人有一定数量的金币，金币总数能被 $n$ 整除。每个人可以给他左右相邻的人一些金币，最终使得每个人的金币数目相等。你的任务是求出被转手的金币数量的最小值。比如， $n=4$ ，且4个人的金币数量分别为1,2,5,4时，只需转移4枚金币（第3个人给第2个人两枚金币，第2个人和第4个人分别给第1个人1枚金币）即可实现每人手中的金币数目相等。

OJ题目（中位数）：POJ 1723、POJ 3579

本次作业截止时间10.14,提交的文件：学号\_姓名\_1014.doc