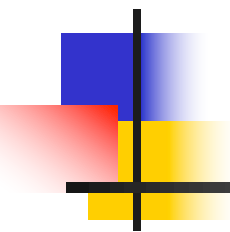


算法设计与分析

Computer Algorithm Design & Analysis

赵峰

zhaof@hust.edu.cn



Chapter 5

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，

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

- **上中位数**： $i = \lceil (n+1)/2 \rceil$ **注：一般取下中位数。**

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。

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

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

从一个由 n 个元素构成的集合中选择第 i 个顺序统计量的问题。

- 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）。

选择问题：

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。

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

设法找元素集合里面的第 k 小元素，该元素为集合的第 k 个顺序统计量。

时间复杂度： $O(n)$

9.1 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元素呢？

$n-1$ 次，时间： $O(n)$

MINIMUM(A)

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

最大值呢？ $n-1$ ，时间： $O(n)$

- “同时” 找集合中的最大值和最小值呢？

同时找集合中的最大值和最小值

- 如果分别独立地找其中的最小值和最大值，则各需做 $n-1$ 次比较，共需 $2n-2$ 次比较。

MAXMIN(A)

```
max ← min ← A(1)  // 设n为奇数
for i = 2 to A.length-1 step by 2
do
    if (A[i] > A[i+1])
        max1 = A[i];
        min1 = A[i+1];
        if max1 > max
            max = max1
        if min1 < min
            min = min1
return max, min
```

- **成对比较**。除了max和min的初始化，其余每对元素元素需**3次比较**即可。
 - 如果n为奇数，共需 $3\lfloor n/2 \rfloor$ 次比较；
 - 如果n是偶数，共需 $3n/2 - 2$ 次比较。
- 总的比较次数至多是 $3\lfloor n/2 \rfloor$

9.2 期望为线性时间的选择算法

■借助QUICKSORT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-PARTITION:

随机化的PARTITION过程

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

PARTITION(1, n) : 在第一次划分后, 主元素 v 被放在位置 $A(j)$ 上, 则有 $j-1$ 个元素小于或等于 $A(j)$, 且有 $n-j$ 个元素大于或等于 $A(j)$ 。此时,

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

- 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 次调用选用的主元素是第 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可以等概率地返回任何元素作为主元，概率是 $1/n$ 。

对所有 $k=1,2,\dots,n$ ，定义指示器随机变量 X_k ：

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

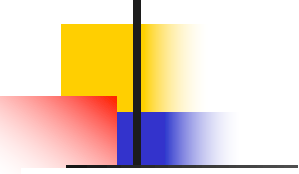
假设 A 中元素是互异的，则有

$$E[X_k] = 1/n$$

■ 期望上界分析

- 一次RANDOMIZED –SELECT调用中，选择 $A[q]$ 作为主元，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。
- 设 $T(n)$ 是单调递增的。设每次划分第 i 个元素总落在元素数较多的一边。
- 对一次给定的RANDOMIZED –SELECT调用，指示器随机变量 X_k 的值在 k 时为1，其他值为0。
- 当 $X_k=1$ 时，如果需要递归，两个子数组的大小分别为 $k-1$ 和 $n-k$ ，算法仅在其中之一、且是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，

➤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，其他各项在总和中出现两次；

则有：

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

- 代换法证明： $E[T(n)] = O(n)$.
- 即证明，存在常数 c ，有 $E[T(n)] \leq cn$ 。

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

注： a 是为去掉 $E[T(n)]$ 中的 $O(n)$ 而引入的常数

■ 这里，为了证明 $E[T(n)] \leq cn$ ，须有 $cn/4 - c/2 - an \geq 0$ 。

- 选取常数 c , 使得 $c > 4a$, $(c/4 - a) > 0$, 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。

- 同时, $n < 2c/(c-4a)$ 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

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。

步骤：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 个元素进行分类并找出其中间元素 m_i ,

$1 \leq i \leq \lfloor n/r \rfloor$, 共得 $\lfloor n/r \rfloor$ 个中间值 (中位数) —— **一次取中**。

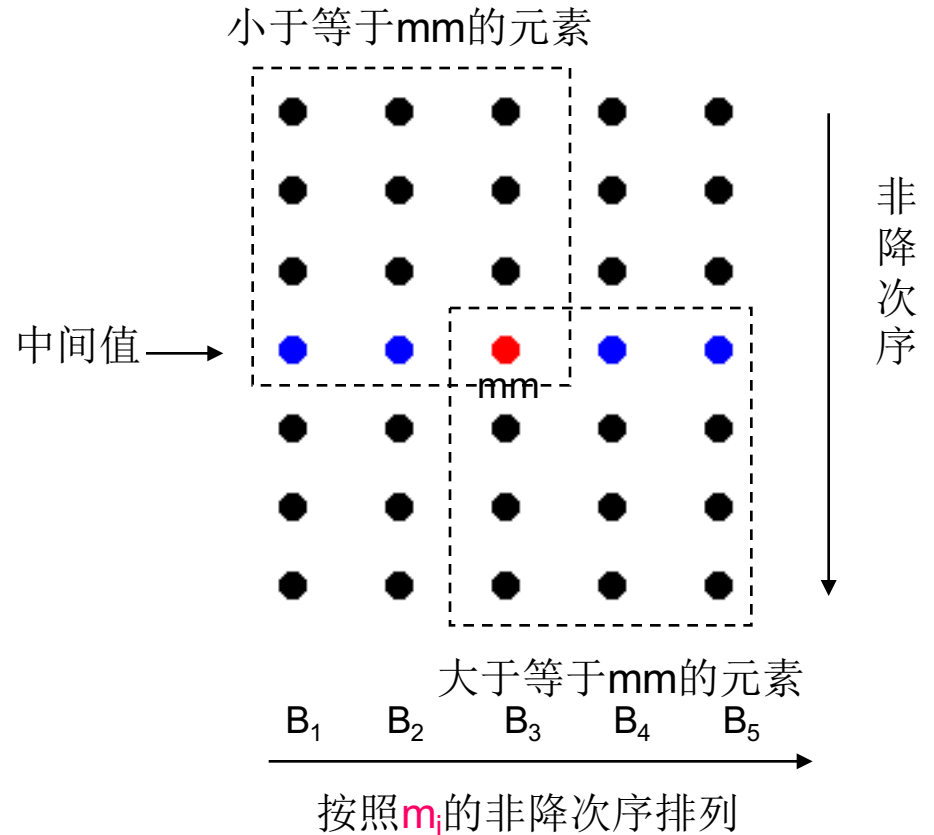
再后，对这 $\lfloor n/r \rfloor$ 个中间值分类，再找出其中间值 m_m (中位数) 。

最后，将 m_m 作为划分元素执行划分 —— **二次取中**。

■ 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, $1 \leq i \leq 5$

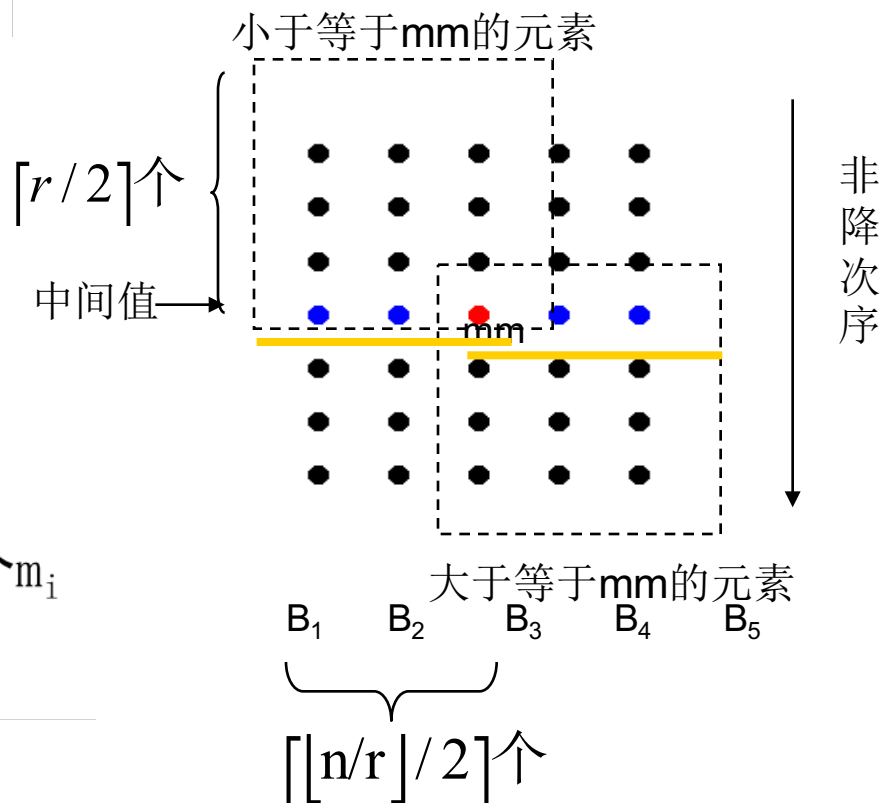
由图所示有：



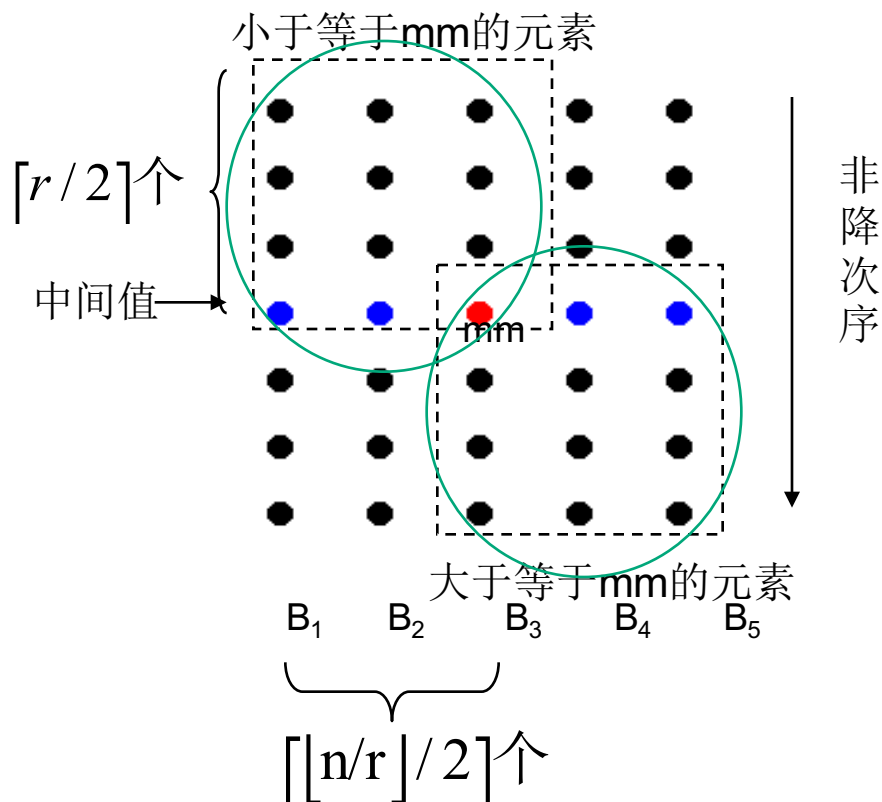
r 个元素的中间值是第 $\lceil r/2 \rceil$ 小元素；

至少有 $\lceil [n/r]/2 \rceil$ 个 m_i 小于或等于 mm ；

也至少有 $\lfloor n/r \rfloor - \lceil [n/r]/2 \rceil + 1 \geq \lceil [n/r]/2 \rceil$ 个 m_i 大于或等于 mm 。



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



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

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)。

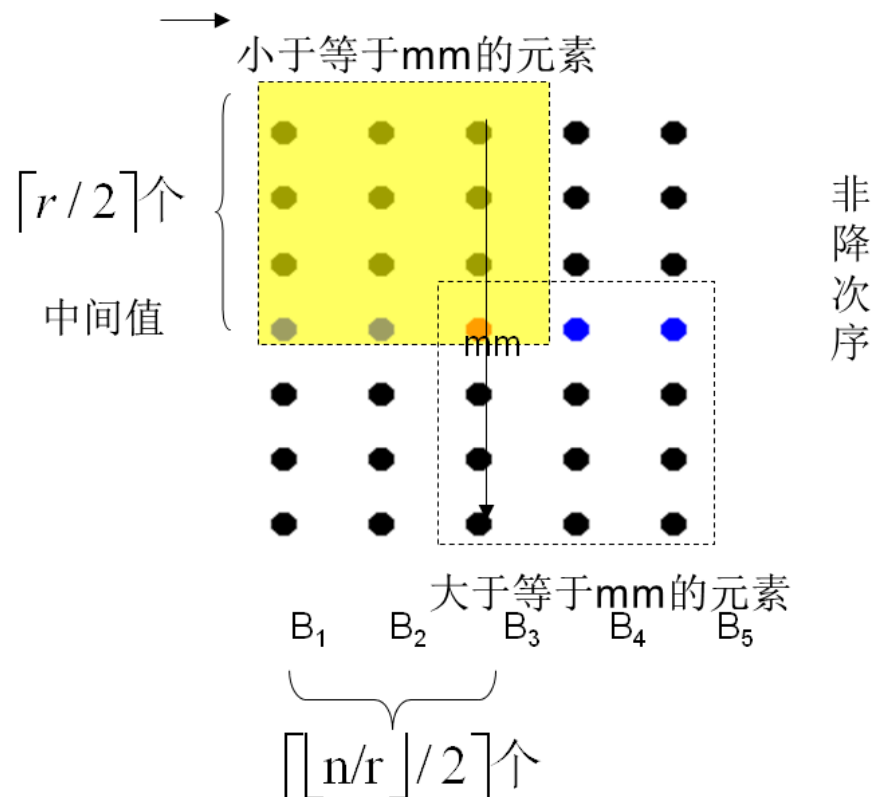
可得到，

- ◆ 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

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

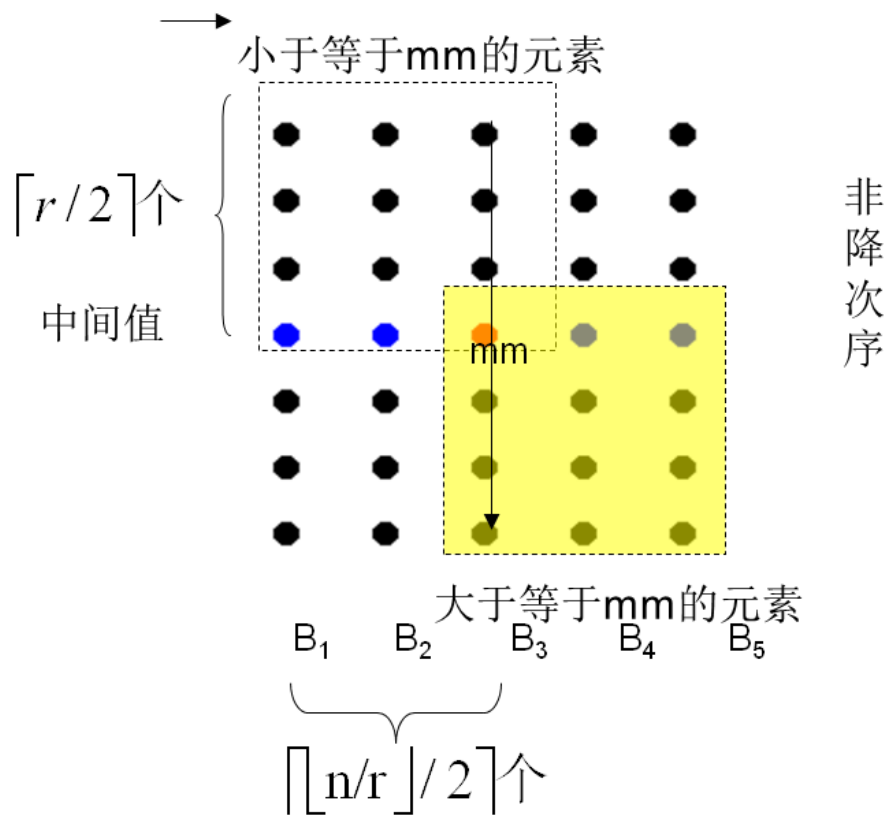
$$\begin{aligned} & n - 1.5\lfloor n/5 \rfloor \\ &\leq n - 1.5(n - 4) / 5 \\ &= 0.7n + 1.2 \\ &\text{注: } \lfloor n/5 \rfloor \geq (n - 4) / 5 \end{aligned}$$

算法导论: $0.7n+6$



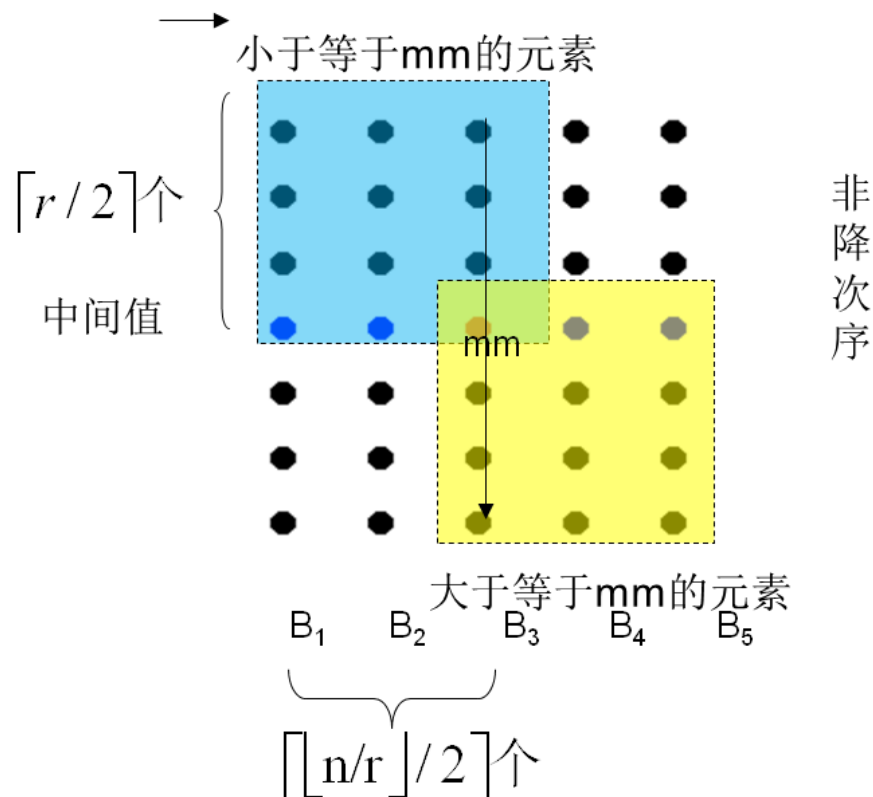
同理,

- ◆ 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 $n - 1.5\lfloor n/5 \rfloor \leq 0.7n + 1.2$ 个元素小于或等于 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，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,k,n)

//在集合A中找第k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k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 $\lfloor n/r \rfloor$ 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

:k=j: return(v)

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

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

endcase

end SELECT2

3) SELECT2的时间分析

记 $T(n)$ 是SELECT2所需的最坏情况时间

对特定的 r 分析SELECT2：选取 $r=5$ 。

假定 A 中的元素各不相同，则有

将个数固定的 r 个元素的
排序时间视为常数时间

- ① 若 $n \leq r$ ，则采用插入法直接对 A 分类并返回第 k 小元素 $\rightarrow O(1)$
 - 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 $\rightarrow O(n)$
 - ③ 设 $M = \{m_1, m_2, \dots, m_{\lfloor n/r \rfloor}\}$ 是 $\lfloor n/r \rfloor$ 子集合的中间值集合 $\rightarrow O(n)$
 - ④ $v \leftarrow \text{SELECT2}(M, \lceil \lfloor n/r \rfloor / 2 \rceil, \lfloor n/r \rfloor)$ $\rightarrow T(n/5)$
 - ⑤ $j \leftarrow \text{PARTITION}(A, v)$ $\rightarrow O(n)$
 - ⑥ case $\rightarrow T(3n/4)$ ，当 $n \geq 24$
 - : $k=j$: return(v)
 - : $k < j$: 设 S 是 $A(1:j-1)$ 中元素的集合; return($\text{SELECT2}(S, k, j-1)$)
 - :else: 设 R 是 $A(j+1:n)$ 中元素的集合; return($\text{SELECT2}(R, k-j, n-j)$)
- endcase
- end SELECT2

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。

$$T(n) = O(n)$$

方法二：选取其他的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 。

相等元素的一半

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， $T(n) = O(n)$

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(A, i, j)直接对每组的 r 个元素分类，在分类好的序列中，中间元素即为中间下标位置所对应的元素。

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。从而可方便使用递归调用的方式对这些中间值进行排序并找出中间值的中间值（即，二次取中）。

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

算法3.12 SELECT2算法实现的递归程序描述

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

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

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

: $j-m+1 > k$: return SEL(A, m, j-1, k)

: else: return SEL(A, j+1, p, k-(j-m+1))

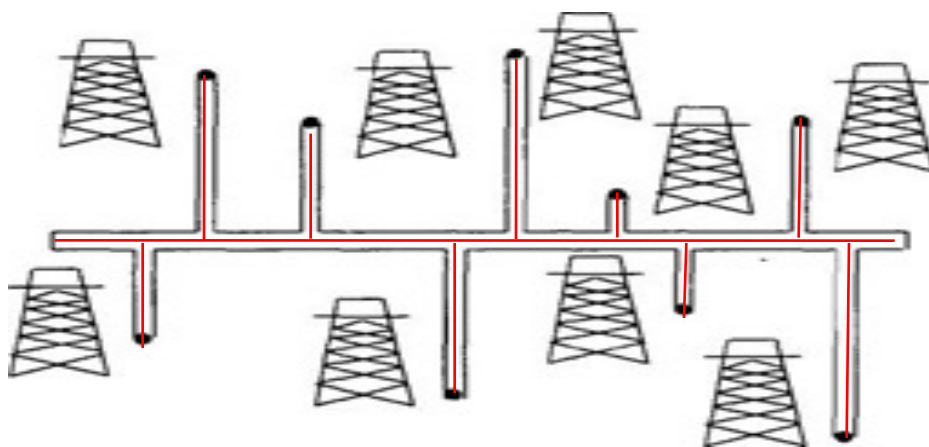
endcase

end SEL

9.4 中位数问题

例4.1：石油管的最优位置

Olay教授正在为一家石油公司咨询，公司正在计划建造一条由东向西的**大型管道**。该管道要穿过一个有 n 口井的油田。从每口井中都有一条**喷油管**沿最短路径与主管道直接相连(或南或北)，如图所示



问题：给定各口井的 x 坐标和 y 坐标。问，Olay教授如何选择主管道的最优位置，使得喷管长度总和最小？

算法分析：

1) 由于主管道是由东向西的，因此要使相连油井与主管道的喷油管最短，喷油管方向必须南北相连，与主管道垂直，即主管道的最优位置应为一条 $y = y_k$ 的水平线。

即，问题的解是求最优位置 y_k 。

2) 为了使 y_k 与各油井的 y 坐标 y_1, \dots, y_n 间的距离和最短，我们将 y_1, \dots, y_n 由小到大排序，选择最中间的那个点作为 y_k 。

即，确定主管道的最优位置，就是求这 n 个油井的 y 坐标的中位数。

主管道最优位置(**y坐标的中位数**)：

- 若油井数为奇数，则第 $(n + 1)/2$ 小的y坐标作为 y_k ；
- 若油井数为偶数，则第 $n/2$ 小的y坐标值与第 $(n/2 + 1)$ 小的y坐标值的平均数作为 y_k 的值。

问题：

证明：按照上述策略设计的主管道位置是最优的吗？

证明：该最优位置可在线性时间内确定吗？

1. 带权中位数

对分别具有正的权重 $\omega_1, \omega_2, \dots, \omega_n$ 且 $\sum_{i=1}^n \omega_i = 1$ 的 n 个不同元素 x_1, x_2, \dots, x_n ，带权中位数是满足如下条件的元素 x_k ：

和

$$\sum_{x_i < x_k} \omega_i < \frac{1}{2}$$
$$\sum_{x_i > x_k} \omega_i \leq \frac{1}{2}$$

所有小于 x_k 的元素

所有大于 x_k 的元素

隐含有序

2. 带权中位数应用

(1) 一维空间上的问题

一条直线上有若干个带权的点 p_1, p_2, \dots, p_n ，它们的权重分别是 $\omega_1, \omega_2, \dots, \omega_n$ ，在该直线上寻找一个点 p ，使得

$$\sum_{i=1}^n \omega_i d(p, p_i)$$

最小，其中 $d(a, b)$ 表示点 a 与 b 之间的距离 $d(a, b) = |a - b|$

——称点 p 为该 n 个点的一维**带权中位数**。

分析：由于各点被赋了权，因此上述的带权中位数 p 不一定是按
 递增排序后的 p_1, p_2, \dots, p_n 中处于中间位置的那个点（甚至
 p 不一定是 p_1, p_2, \dots, p_n 中的一个），而是满足下述条件的
 点 p_k ：

在递增序列 $p_1, \dots, p_{k-1}, p_k, p_{k+1}, \dots, p_n$ 中，子序列 p_1, \dots, p_{k-1} 各点
 的权的和小于等于 $1/2$ ，并且子序列 p_{k+1}, \dots, p_n 各点的权
 的和也小于等于 $1/2$ （这里 $\sum_{i=1}^n \omega_i = 1$ ），即

$$\sum_{x_i < x_k} \omega_i \leq \frac{1}{2} \quad \text{和} \quad \sum_{x_i > x_k} \omega_i \leq \frac{1}{2}$$

(试比较上述定义和前面的带权中位数的定义)

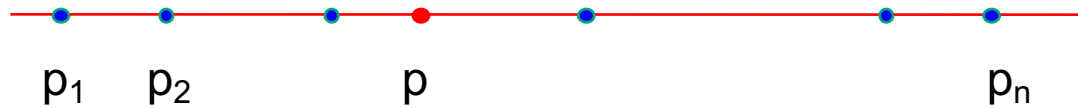
例 一维邮局位置问题

已知 n 个邮局分布在一条直线上，坐标点分别为 p_1, p_2, \dots, p_n 。一邮递员每天需要多次到这些邮局取邮件，设邮递员所处位置为点 p 。由于时间不一致，邮递员每次到一个邮局取件后需要先回到 p 点，然后再去下一个邮局。设邮递员每天到这些邮局的次数分别 $\omega_1, \omega_2, \dots, \omega_n$ 。

问， p 设在哪里可使得邮递员每天到各个邮局走的总里程最短？

分析：

1) 图示



2) 权重：邮递员每天需要到邮局的取件次数即为该问题的权重，可换算成为 **[0..1]** 值。

3) 里程：对邮局 i ，邮递员从 p 处出发到 p_i 处，每天的里程数为 $\omega_i d(p, p_i)$ ，这里， **$d(p, p_i) = |p - p_i|$** ，代表 p 到 p_i 的距离

(注：这里只考虑单向)；

所以，该问题即是求 $\sum_{i=1}^n \omega_i d(p, p_i)$ 的最小值

——带权中位数问题。

(2) 二维空间上的问题

设二维平面上分布着 n 个点 p_1, p_2, \dots, p_n ，点 p_i 的坐标用 (x_i, y_i) 表示，每个点附有一个权重 ω_i ， $\sum_{i=1}^n \omega_i = 1$ 。

定义点 $p_1(x_1, y_1)$ 与点 $p_2(x_2, y_2)$ 之间的距离是

$$d(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2| \quad (\text{称为Manhattan距离})$$

问题：在二维平面上找一个点 $p(x, y)$ ，使得 $\sum_{i=1}^n \omega_i d(p, p_i)$ 最小，则称 p 为该二维平面上 n 个点的带权中位数。

二维空间点的带权中位数求解：

由于 $d(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2|$ ，故可将问题转换为在 x 与 y 两个方向上分别求解带权中位数的问题，从而将二维问题转化为一维问题。

设最佳点为 p ，则满足：

$$\sum_{p_i < p} \omega_i \leq \frac{1}{2} \quad \text{和} \quad \sum_{p_i > p} \omega_i \leq \frac{1}{2}$$

即带权中位数问题

例 二维邮局位置问题


一维邮局问题的推广：设这些邮局分布在二维平面上，邮局 p_i 的坐标记为 (x_i, y_i) 。最佳点记为 $p(x, y)$ 。点之间的距离取 Manhattan 距离，即，

$$d(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2|$$

问， p 设在哪里可使得邮递员每天到各个邮局走的总里程最短？

分析：该问题即是求 $\sum_{i=1}^n \omega_i d(p, p_i)$ 的最小值

——二维带权中位数问题。



证明：满足 $\sum_{i=1}^n \omega_i d(p, p_i)$ 最小的点

$$\sum_{p_i < p} \omega_i \leq \frac{1}{2} \text{ 和 } \sum_{p_i > p} \omega_i < \frac{1}{2}$$

记 $d(i,j)$ 是点 i 到点 j 的距离，令 $d(i,j)=|\text{num}_i-\text{num}_j|$ ，且有
 $d(i,j)=d(j,i)$

若最优点在 t ，则到其它点的带权距离均应大于等于到 t 的带权距离。

- t 右边的点：不失一般性，取点 $t+1$ ，则有：

$$\sum_{i \neq t} \omega_i d(i, t) \leq \sum_{i \neq t+1} \omega_i d(i, t+1)$$

上式可转化为：

$$\begin{aligned} & \sum_{l < t} \omega_l d(l, t) + \sum_{t+1 < r} \omega_r d(r, t) + \omega_{t+1} d(t+1, t) \\ & \leq \sum_{l < t} \omega_l d(l, t+1) + \sum_{t+1 < r} \omega_r d(r, t+1) + \omega_t d(t, t+1) \end{aligned}$$

整理一下：

$$\begin{aligned} & \sum_{t+1 \leq r} \omega_r d(r, t) - \sum_{t+1 \leq r} \omega_r d(r, t+1) + \omega_{t+1} d(t, t+1) \\ & \leq \sum_{l < t} \omega_l d(l, t+1) - \sum_{l < t} \omega_l d(l, t) + \omega_t d(t+1, t) \end{aligned}$$

进一步有：

$$\begin{aligned} & \sum_{t+1 \leq r} \omega_r (d(r, t) - d(r, t+1)) + \omega_{t+1} d(t, t+1) \\ & \leq \sum_{l < t} \omega_l (d(l, t+1) - d(l, t)) + \omega_t d(t+1, t) \end{aligned}$$

而，

$$d(l, t+1) - d(l, t) = d(t, t+1)$$

$$d(r, t) - d(r, t+1) = d(t+1, t)$$

$$\text{且, } d(t, t+1) = d(t+1, t)$$



因此：

$$\sum_{l \leq t} \omega_l \geq \sum_{t+1 \leq r} \omega_r$$

即：

$$\sum_{l < t} \omega_l + \omega_t \geq \sum_{t < r} \omega_r$$

因此，若t是最优点，则其左边的权值之和加上 ω_t 后大于右边的权值之和。

同理，t左边的点t-1，其右边的权值之和加上 ω_t 后也大于左边的权值之和，即

$$\sum_{t < r} \omega_r + \omega_t \geq \sum_{l < t} \omega_l$$

满足的条件。

而进一步有：

因为左边的权值之和 + $\omega_t \geq$ 右边的权值之和，所以：

$$\sum_{l < t} \omega_l + \omega_t \geq \sum_{t < r} \omega_r = \sum_{i=1}^n \omega_i - (\sum_{l < t} \omega_l + \omega_t)$$

$$\Rightarrow 2 * (\sum_{l < t} \omega_l + \omega_t) \geq \sum_{i=1}^n \omega_i$$

$$\Rightarrow 2 * \sum_{t < r} \omega_r \leq \sum_{i=1}^n \omega_i \quad \sum_{t < r} \omega_r < \frac{1}{2}$$

同理可得：

$$2 * \sum_{l < t} \omega_l \leq \sum_{i=1}^n \omega_i \quad \sum_{l < t} \omega_l < \frac{1}{2}$$

证毕。（这正是带权中位数所具备的性质）

$$\text{令 } \sum_{i=1}^n \omega_i = 1$$