算法基础

主讲人：　庄连生

Email: { *lszhuang@ustc.edu.cn* }

Spring 2018，**USTC**

University of Science and Technology of China

第七讲 顺序统计学

内容提要：

- 最小值和最大值
- 以期望线性时间做选择
- 最坏情况线性时间的选择

问题描述

- 在一个由n个元素组成的集合中，第i个顺序统计量是该集合中第i个小的元素。
- 一个中位数是它所在集合的“中点元素”。
  - ✓ 当n为奇数时，中位数是唯一的，$i=(n+1)/2$；
  - ✓ 当n为偶数时，中位数有两个，即：$i=n/2$(下中位数)和$i=n/2+1$（上中位数）。
- 选择问题：从一个由n个不同值构成的集合中，选择其第i个顺序统计量。
  - ✓ **输入**：一个包含n个（不同的）数的集合A和一个数i, $1 \le i \le n$
  - ✓ **输出**：元素$x \# A$，它恰大于A中其它i-1个元素。

第七讲 顺序统计学

内容提要：

- 最小值和最大值
- 以期望线性时间做选择
- 最坏情况线性时间的选择

最小值和最大值

- **最小/最大值**：最坏情形进行n-1次比较，时间复杂度为θ(n)；
- 假设集合放在数组*A*中，且*length*[A] = *n*。

```
MINIMUM ( A )
    min ← A[1];
    for  i ← 2 to length[A]
       do if min > A[i]
              then min ←  A[i]
    return min
```

总共比较了$n-1$次，时间复杂度：$O(n)$

最小值和最大值

同时找最小值和最大值

1）记录比较过程中遇到的最小值和最大值；

2）成对处理元素，比较当前两个元素，把较小者与最小值比较，较大者与最大值比较；

```
MAX-MINIMUM ( A )
if  length[A] is odd
      then  min ← A[1];  max ← min;
      else min ← MIN( A[1], A[2] ), max ← MAX( A[1], A[2] );
i ++;
while  i ≤ length[A]
       min ← MIN( MIN(A[i], A[i+1]), min )
       max ← MAX( MAX(A[i], A[i+1]), max )
       i ← i+2;
end
return min, max
```

最小值和最大值

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 $3n/2-2$ 次比较。

时间复杂度为：O(n)

第七讲 顺序统计学

内容提要：

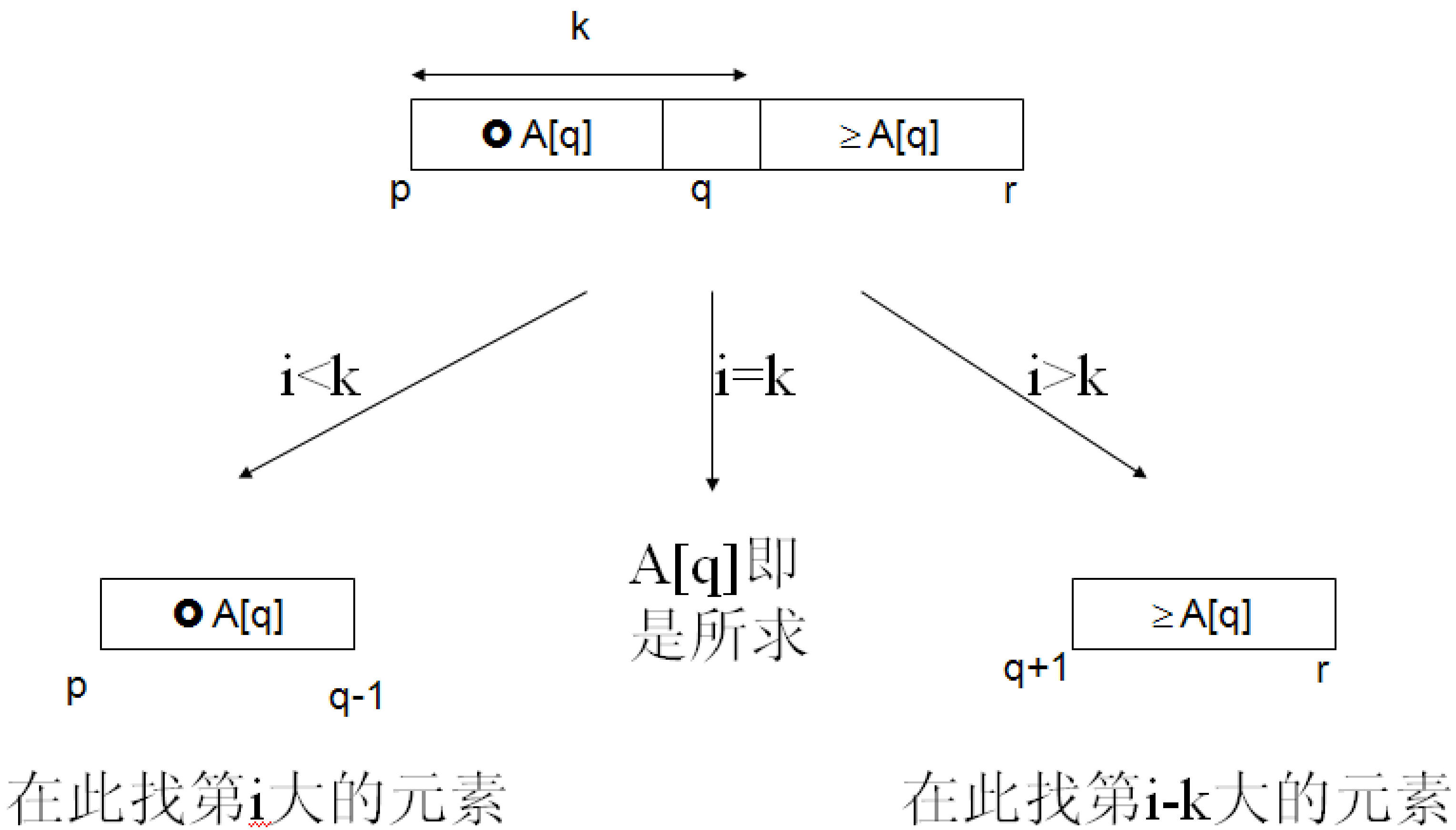
- 最小值和最大值
- 以期望线性时间做选择
- 最坏情况线性时间的选择

以期望线性时间做选择

- **基本思想**：采用分治策略，借鉴快速排序的随机划分法，对输入数组进行递归划分，但是只处理划分的一边。

```
RANDOMIZED-SELECT ( A, p, r, i )
    if  p = r                                          // 临界问题处理
        then  return A[p]
    q ← RANDOMIZED-PARTITION( A, p, r )   //进行划分，返回划分元下标
    k ← q – p + 1
    if  i = k
        then return A[q];
    else if i < k
         then return RANDOMIZED-SELECT ( A, p, q - 1, i )
    else
          return RANDOMIZED-SELECT( A, q+1, r, i – k )
```

以期望线性时间做选择



以期望线性时间做选择

时间复杂度分析：

- 幸运的例子：每次都能去除十分之一以上。

  $T(n)=T(9n/10)+\Theta(n)=O(n)$

  可利用主方法计算出。

  $n^{\log_{10/9} 1}=n^0=1.$

- 运气不好的例子：每次都只能去除一个元素。

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

以期望线性时间做选择

平均情况:

为了求上限方便起见，假定第i小的元素总是掉在较大的Partition中。

- 对任一k=1…n，A[$p..q$]恰有k个元素的几率为1/n。
- 令X_k=I{A[$p..q$]恰有k个元素}，---Indicator random variable。
- E[X_k]=1/n。

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

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

以期望线性时间做选择

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

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

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

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

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

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

以期望线性时间做选择

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

利用代换法：假定E[T(n)] ○ cn。

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

可以取足够大的*c*使得*c*(*n*/4-1/2)大於*an*使得最后一个不等式成立。

第七讲 顺序统计学

内容提要：

- 最小值和最大值
- 以期望线性时间做选择
- 最坏情况线性时间的选择

最坏情况线性时间的选择

- **基本思想**：类似RandomizedSelect算法，通过对输入数组进行递归划分来找出所求元素，但是算法保证每次对划分是个好划分。

- **主要步骤**：

Step 1. 将n个元素分成5个1组，共ceiling(n/5)组。其中，最后1组有n mod 5个元素；

Step 2. 用插入排序对每组排序，取其中值。若最后1组有偶数个元素，取较小的中值；

Step 3. 递归地使用本算法寻找ceiling(n/5)个中位数的中值x;

//第一次递归调用本身

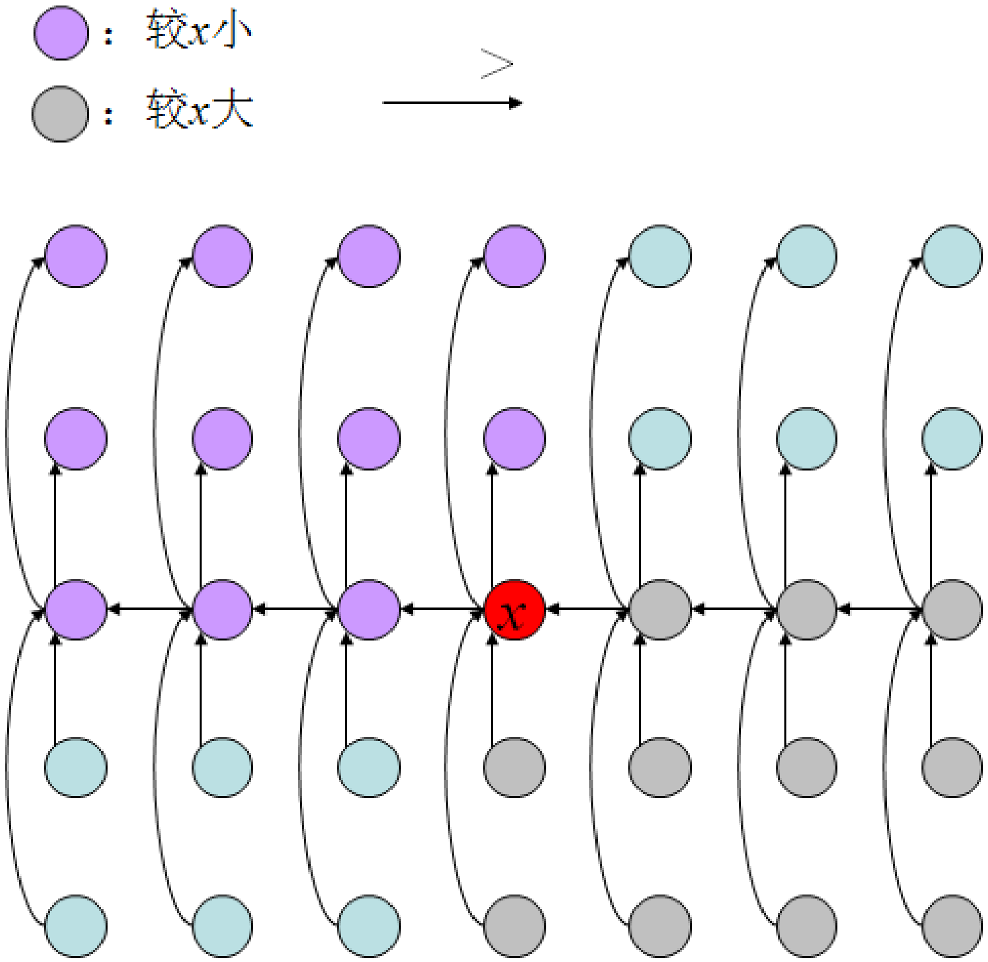
Step 4. 用x作为划分元对数组A进行划分，并设x是第k个最小元；

Step 5. if i = k then return x;

else if i < k then 找左区间的第i个最小元；//第二次递归调用本身

else 找右区间的第i-k个最小元

最坏情况线性时间的选择



最坏情况线性时间的选择

时间复杂度分析：

- 由上页图示，可知至少有 $3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5}\right\rceil \right\rceil - 2\right) \geq \frac{3n}{10} - 6$ 的元素较x来得大。
- 同理，至少有$3n/10$ -6 的元素较x来得小。
- 如果Partition过，$i \neq k$，则至多只要在$7n/10+6$个元素的情况下递归调用Select。
- 而先前找出$\lceil n/5 \rceil$小组中位数的中位数时，只在$n/5$个元素的情况下递归调用 Select。
- 故T(n)=T($\lceil n/5 \rceil$)+T($7n/10$+6)+Θ(n), for n >= 140.

最坏情况线性时间的选择

- 利用替换法，令T(n) = O(cn)

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

假设n>140时，c>=20a，上式就可以成立！

最坏情况线性时间的选择

```
Select( A, p, r, i ){
  if ( r-p <= 140 ) {
        用简单的排序算法对数组A[p..r]进行排序；
        return A[p+k-1];
   }
   n = r – p + 1 ;
   for  i ← 0   to   floor(n/5)    //寻找每组的中位数
       将A[p + 5 * i] 至A[p+5*i+4]的第3小元素与A[p+i]交换位置;
   x = Select( A, p, p + floor(n/5), floor(n/10) ); //找中位数的中位数
   i  = Partition(A, p, r, x ) , j = i – p + 1;
   if( k <= j ) return Select ( A, p, i, k );
   else  return Select( A, i + 1, r, k – j );
}
```

谢谢！

Q & A