



哈尔滨工业大学

海量数据计算研究中心

Massive Data Computing Lab @ HIT

算法设计与分析—入门篇

第三讲 分治法

哈尔滨工业大学

王宏志

wangzh@hit.edu.cn

<http://homepage.hit.edu.cn/pages/wang/>

本讲内容

3.1 分治法

3.2 分治法的简单实例

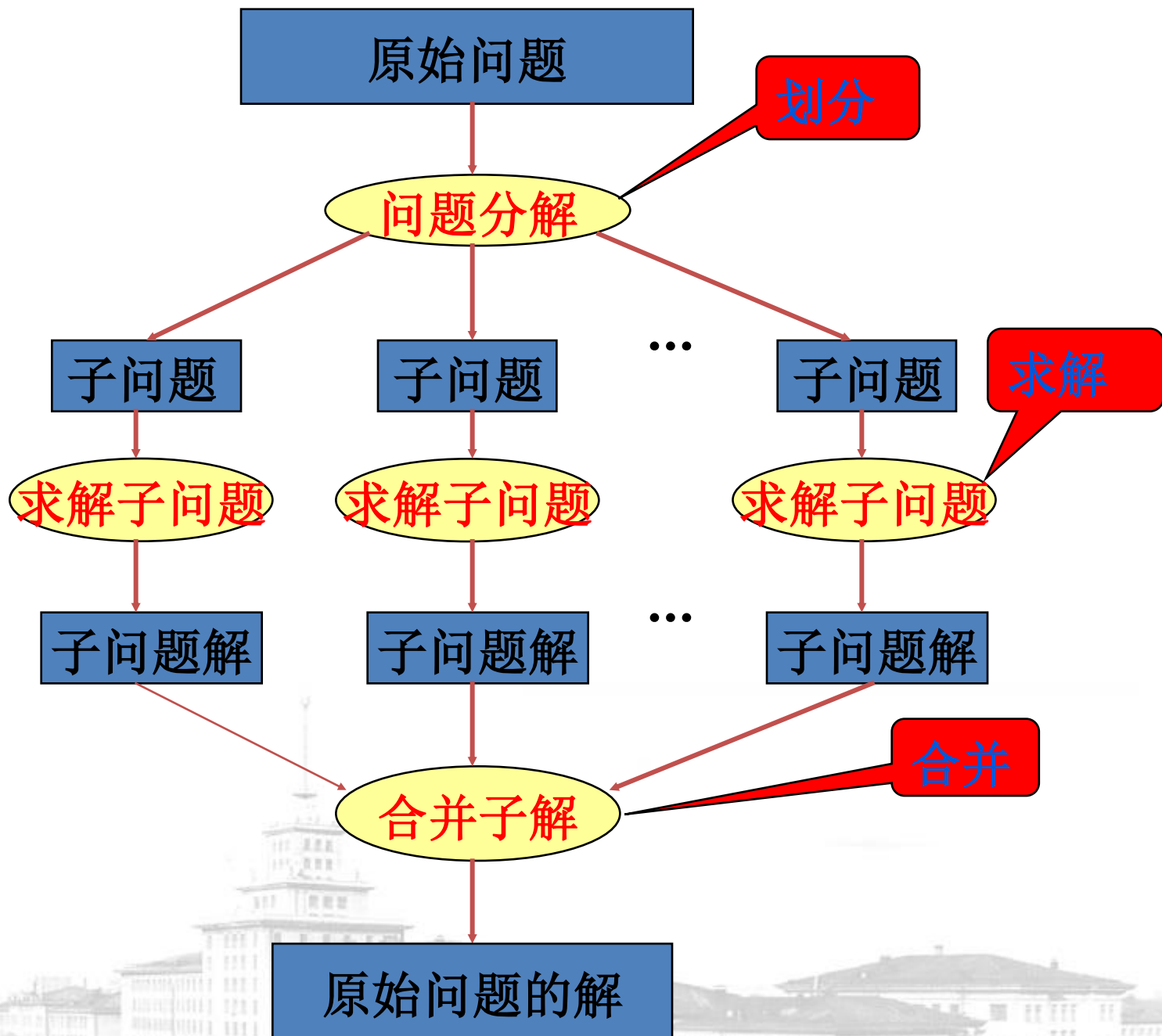
3.3 元素选取问题的线性时间算法

3.4 快速傅里叶变换

分治算法的设计

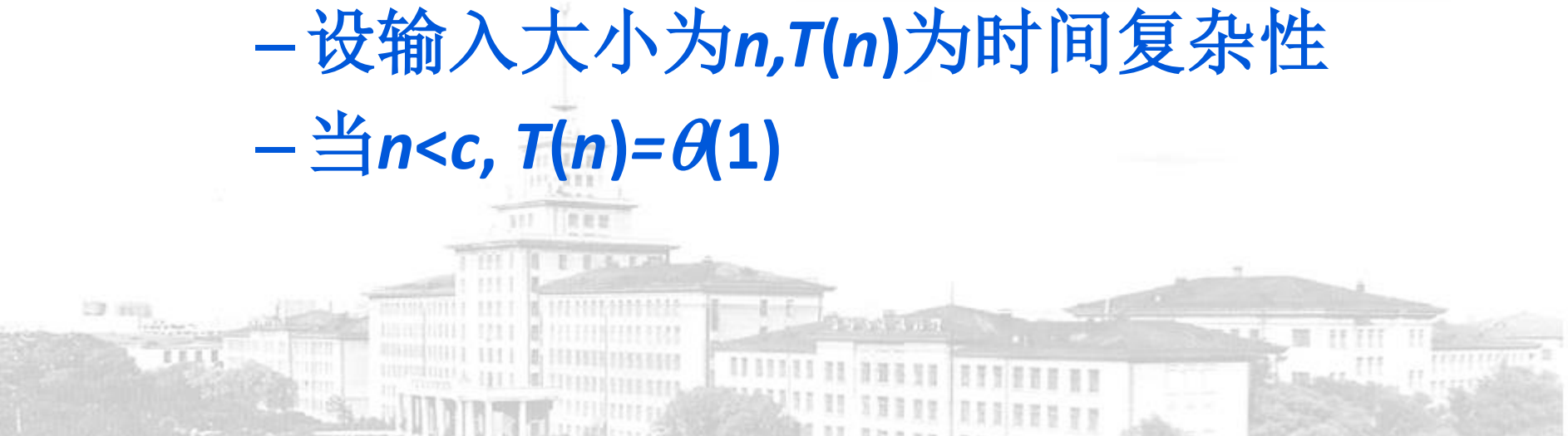
- 设计过程分为三个阶段
 - 划分: 整个问题划分为多个子问题
 - 求解: 求解各子问题
 - 递归调用正设计的算法
 - 合并: 合并子问题的解, 形成原始问题的解





分治算法的分析

- 分析过程
 - 建立递归方程
 - 求解
- 递归方程的建立方法
 - 设输入大小为 n , $T(n)$ 为时间复杂性
 - 当 $n < c$, $T(n) = \theta(1)$



– 划分阶段的时间复杂性

- 划分问题为 a 个子问题。
- 每个子问题大小为 n/b 。
- 划分时间可直接得到= $D(n)$

– 递归求解阶段的时间复杂性

- 递归调用
- 求解时间= $aT(n/b)$

– 合并阶段的时间复杂性

- 时间可以直接得到= $C(n)$



—总之

- $T(n) = \theta(1)$ if $n < c$
- $T(n) = aT(n/b) + D(n) + C(n)$ if $n \geq c$

—求解递归方程 $T(n)$

- 使用第二章的方法



本讲内容

- 3.1 分治法
- 3.2 分治法的简单实例
- 3.3 元素选取问题的线性时间算法
- 3.4 快速傅里叶变换

大整数乘法

输入： n 位二进制整数 X 和 Y

输出： X 和 Y 的乘积

通常，计算 $X * Y$ 时间复杂度为 $O(n^2)$ ，
我们给出一个复杂度为 $O(n^{1.59})$ 的算法。



简单分治算法

$$X = \overset{n/2\text{位}}{A} \overset{n/2\text{位}}{B} \quad Y = \overset{n/2\text{位}}{C} \overset{n/2\text{位}}{D}$$

$$\begin{aligned} XY &= (A2^{n/2} + B)(C2^{n/2} + D) \\ &= AC2^n + (AD+BC)2^{n/2} + BD \end{aligned}$$

算法

1. 划分产生A,B,C,D;
2. 计算 $n/2$ 位乘法AC、AD、BC、BD;
3. 计算AD+BC;
4. AC左移 n 位, (AD+BC)左移 $n/2$ 位;
5. 计算XY。

时间复杂性

$$T(n) = 4T(n/2) + \theta(n)$$

$$T(n) = \theta(n^2)$$

算法的数学基础

$$X = \overset{n/2\text{位}}{\boxed{A}} \overset{n/2\text{位}}{\boxed{B}} \quad Y = \overset{n/2\text{位}}{\boxed{C}} \overset{n/2\text{位}}{\boxed{D}}$$

$$\begin{aligned} XY &= (A2^{n/2} + B)(C2^{n/2} + D) \\ &= AC2^n + (\textcolor{red}{AD} + \textcolor{red}{BC})2^{n/2} + BD \end{aligned}$$

$$\textcolor{red}{AD} + \textcolor{red}{BC} = (A+B)(C+D) - AC - BD$$

算法

1. 划分产生A,B,C,D;
2. 计算A-B和C-D;
3. 计算 $n/2$ 位乘法AC、BD、 $(A+B)(C+D)$;
4. 计算 $(A+B)(C+D) - AC - BD$;
5. AC左移 n 位, $(A+B)(C+D) - AC - BD$ 左移 $n/2$ 位;
6. 计算XY

算法的分析

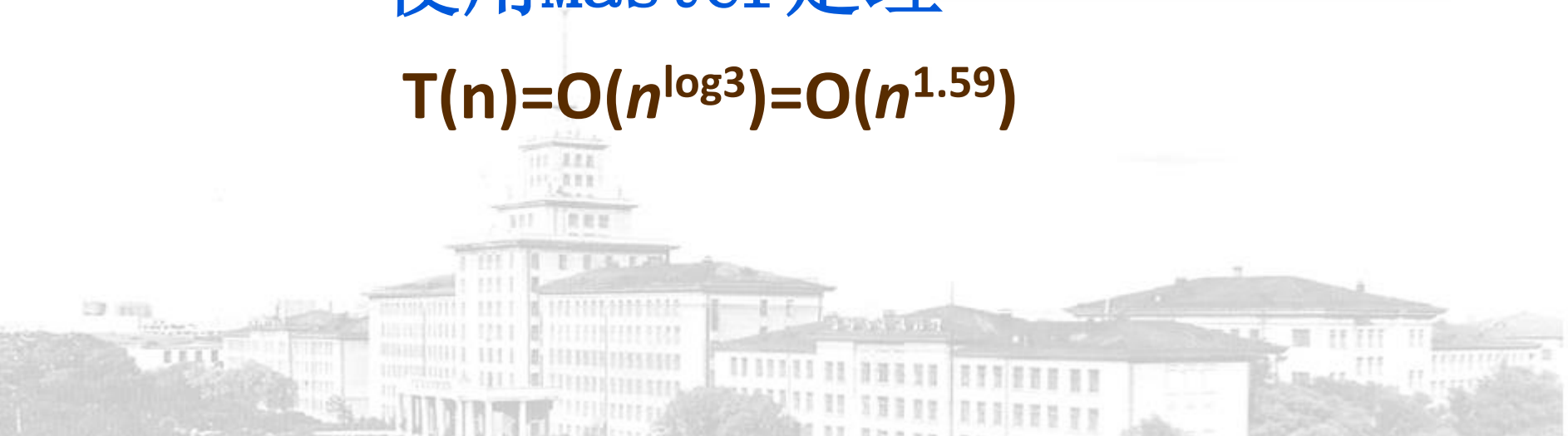
- 建立递归方程

$$T(n)=\theta(1) \quad \text{if } n=1$$

$$T(n)=3T(n/2)+O(n) \quad \text{if } n>1$$

- 使用Master定理

$$T(n)=O(n^{\log 3})=O(n^{1.59})$$



最大值和最小值

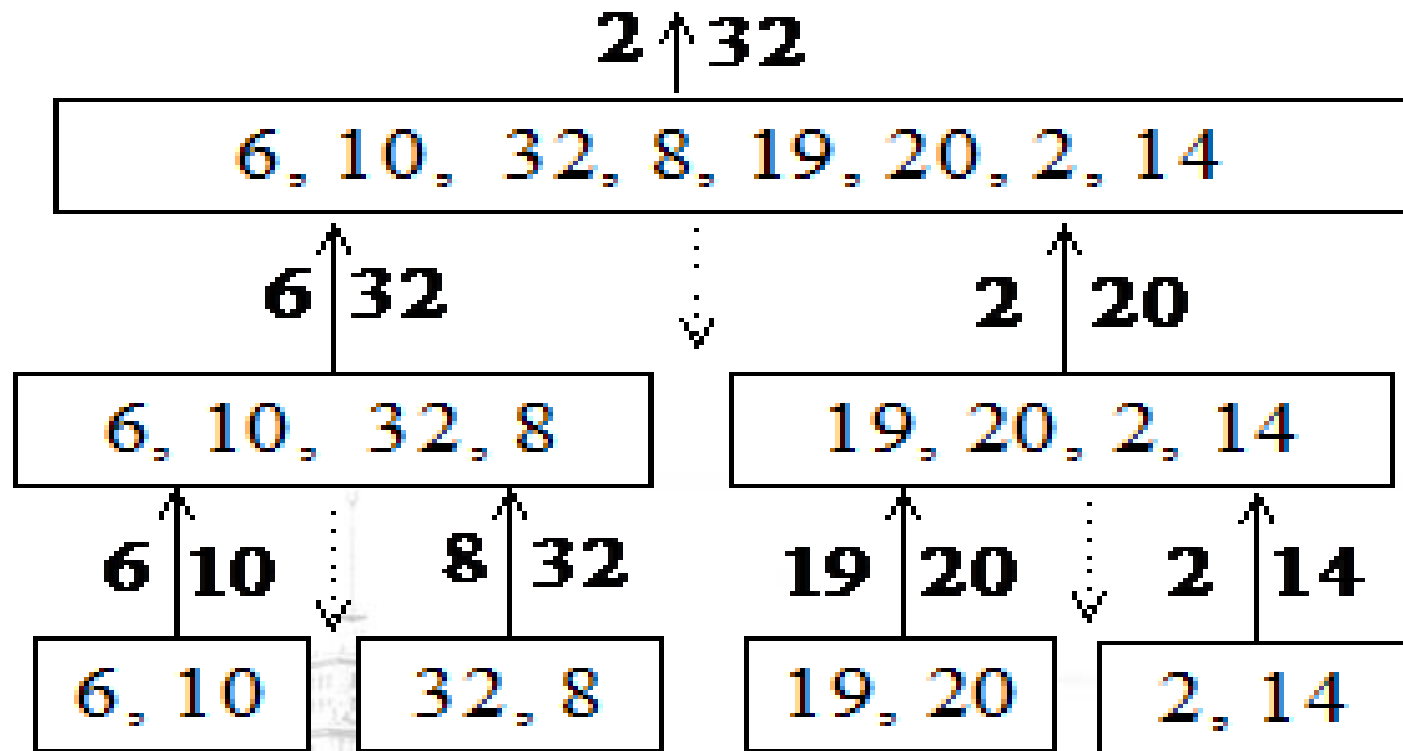
输入：数组 $A[1, \dots, n]$

输出： A 中的 \max 和 \min

通常，直接扫描需要 $2n-2$ 次比较操作
我们给出一个仅需 $\lceil 3n/2 - 2 \rceil$ 次比较操作的算法。



基本思想



算法

算法MaxMin(A)

输入: 数组 $A[i, \dots, j]$

输出: 数组 $A[i, \dots, j]$ 中的max和min

1. If $j-i+1 = 1$ Then 输出 $A[i], A[i]$, 算法结束
2. If $j-i+1 = 2$ Then
3. If $A[i] < A[j]$ Then 输出 $A[i], A[j]$; 算法结束
4. $k \leftarrow (j-i+1)/2$
5. $m_1, M_1 \leftarrow \text{MaxMin}(A[i:k]);$
6. $m_2, M_2 \leftarrow \text{MaxMin}(A[k+1:j]);$
7. $m \leftarrow \min(m_1, m_2);$
8. $M \leftarrow \min(M_1, M_2);$
9. 输出 m, M

算法复杂性

$$T(1)=0$$

$$T(2)=1$$

$$T(n)=2T(n/2)+2$$

$$=2^2T(n/2^2)+2^2+2$$

$$= \dots$$

$$=2^{k-1}T(2)+2^{k-1}+2^{k-2}+\dots+2^2+2$$

$$n=2^k$$

$$=2^{k-1}+2^{k-1}$$

$$=n/2+n-1$$

$$=3n/2-1$$



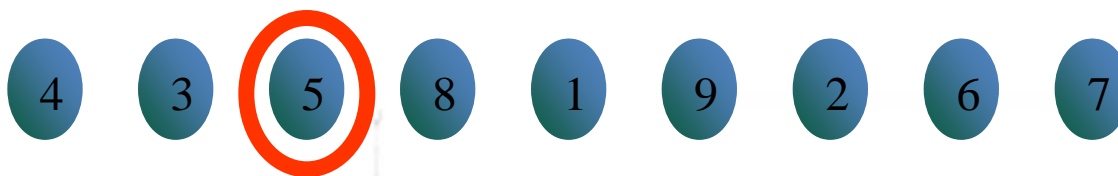
本讲内容

- 3.1 分治法
- 3.2 分治法的简单实例
- 3.3 元素选取问题的线性时间算法
- 3.4 快速傅里叶变换

中位数问题定义

Input: 由 n 个数构成的多重集合 X

Output: $x \in X$ 使得 $-1 \leq |\{y \in X / y < x\}| - |\{y \in X / y > x\}| \leq 1$



中位数选取问题的复杂度

[Blum et al. *STOC*'72 & *JCSS*'73]

– A “shining” paper by five authors:

- Manuel Blum (Turing Award 1995)
- Robert W. Floyd (Turing Award 1978)
- Vaughan R. Pratt
- Ronald L. Rivest (Turing Award 2002)
- Robert E. Tarjan (Turing Award 1986)

– 从 n 个数中选取中位数需要的比较操作的次数介于 $1.5n$ 到 $5.43n$ 之间



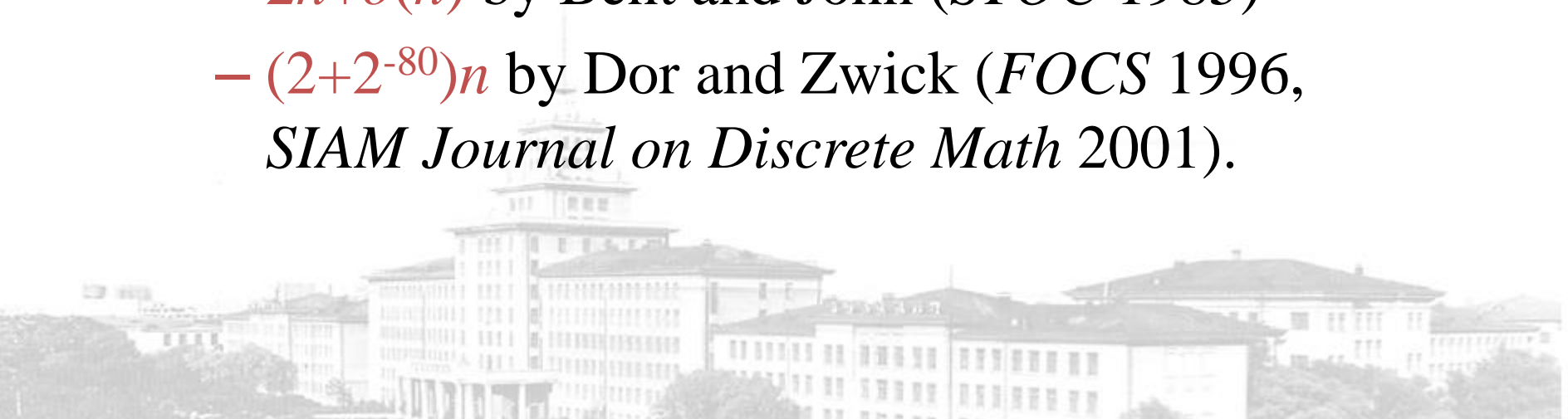
比较操作次数的上下界

- 上界

- $3n + o(n)$ by Schonhage, Paterson, and Pippenger (*JCSS* 1975).
- $2.95n$ by Dor and Zwick (*SODA* 1995, *SIAM Journal on Computing* 1999).

- 下界

- $2n + o(n)$ by Bent and John (*STOC* 1985)
- $(2 + 2^{-80})n$ by Dor and Zwick (*FOCS* 1996, *SIAM Journal on Discrete Math* 2001).

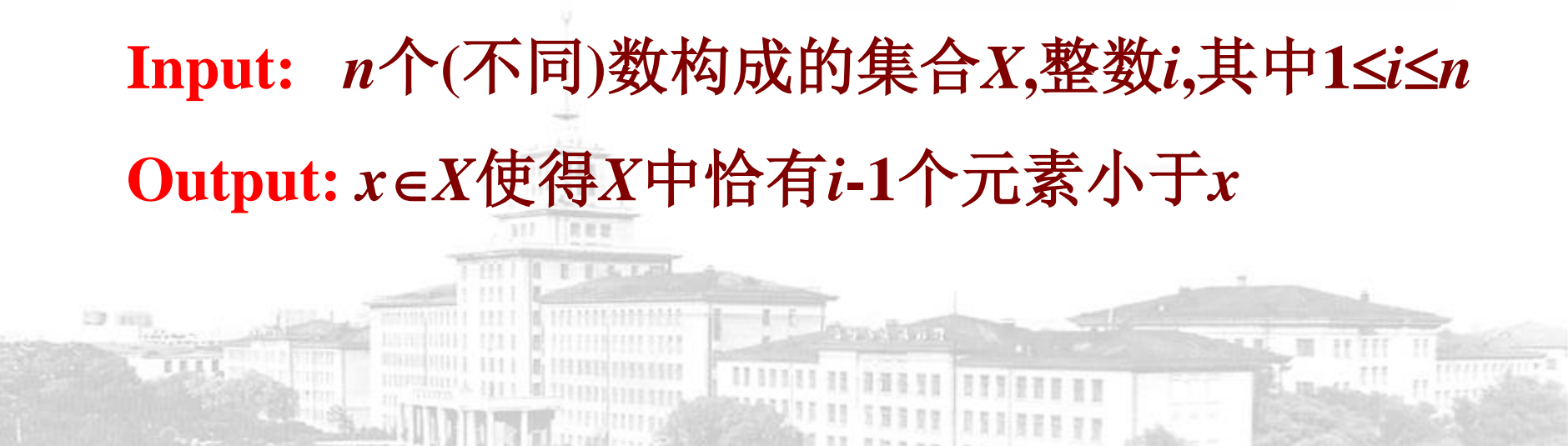


线性时间选择

- 本节讨论如何在 $O(n)$ 时间内从 n 个不同的数中选取第 i 大的元素
- 中位数问题也就解决了，因为选取中位数即选择第 $n/2$ -大的元素

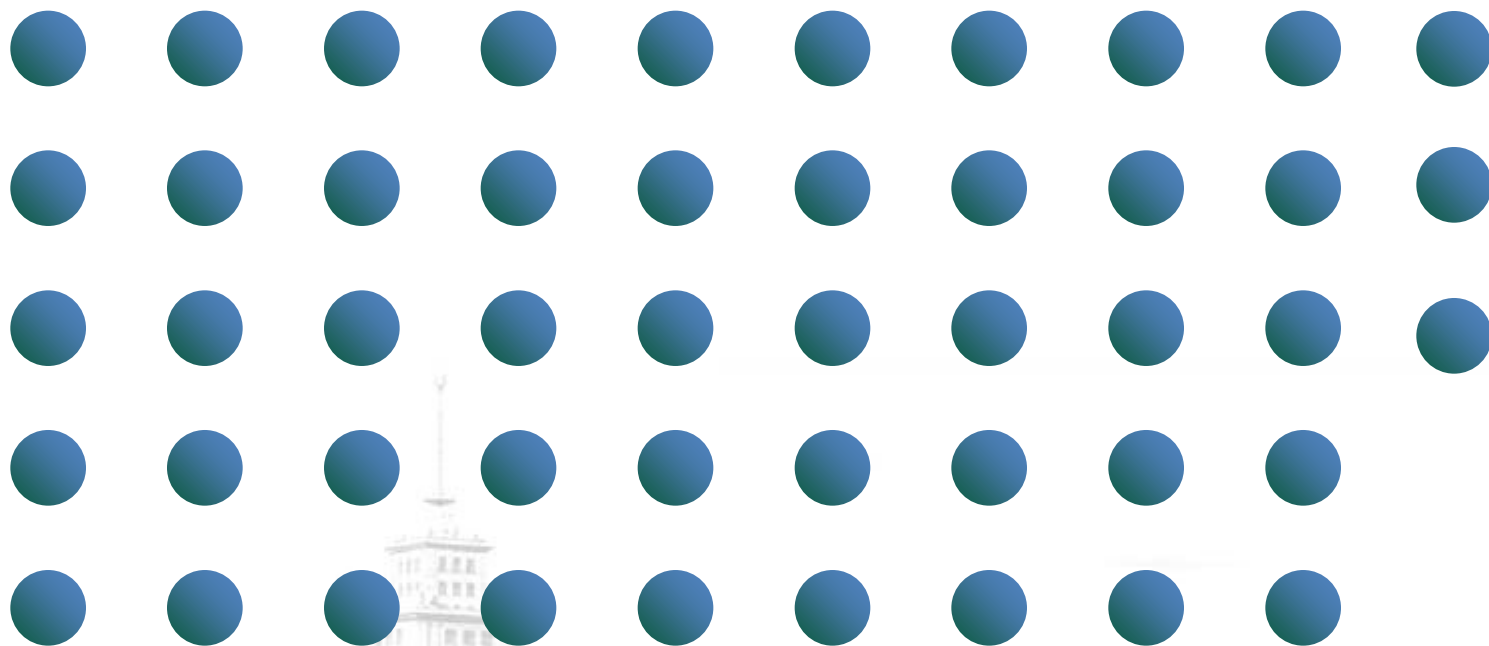
Input: n 个(不同)数构成的集合 X , 整数 i , 其中 $1 \leq i \leq n$

Output: $x \in X$ 使得 X 中恰有 $i-1$ 个元素小于 x

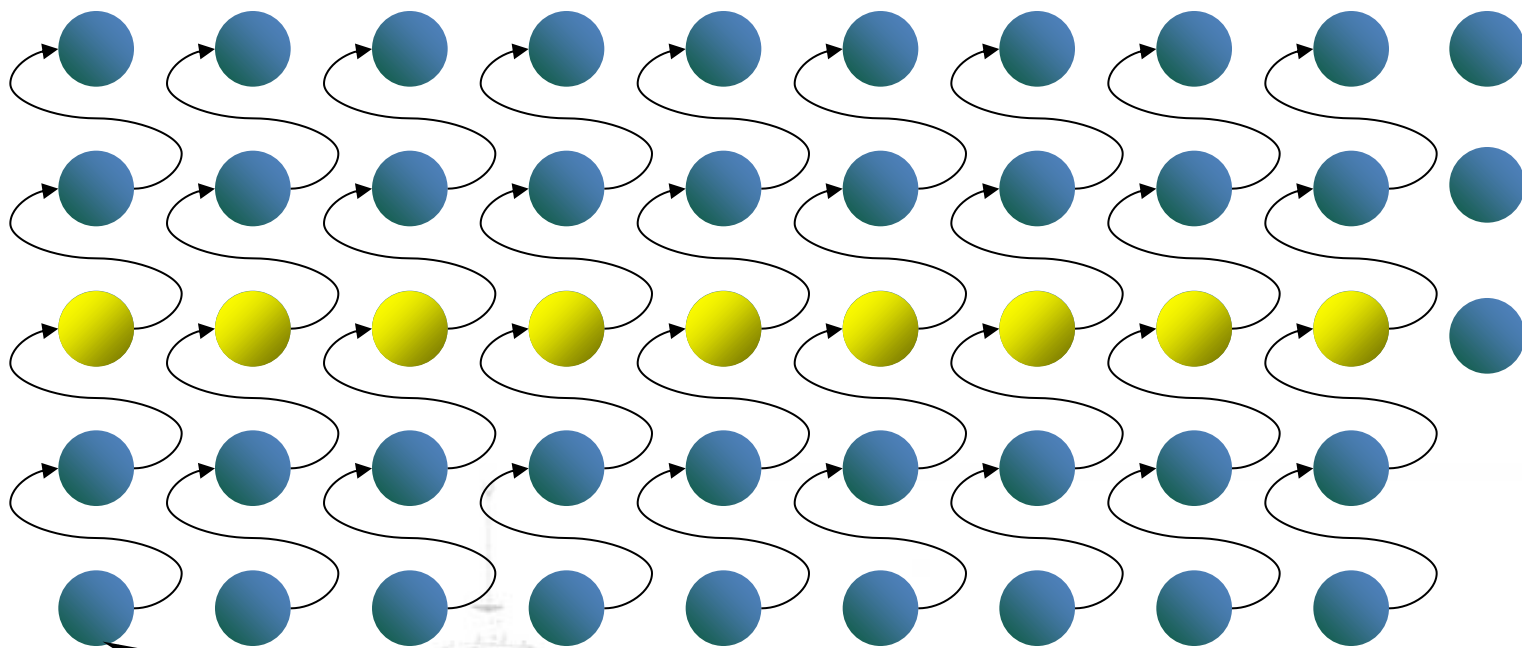


求解步骤

第一步： 分组，每组5个数
最后一组可能少于5个数

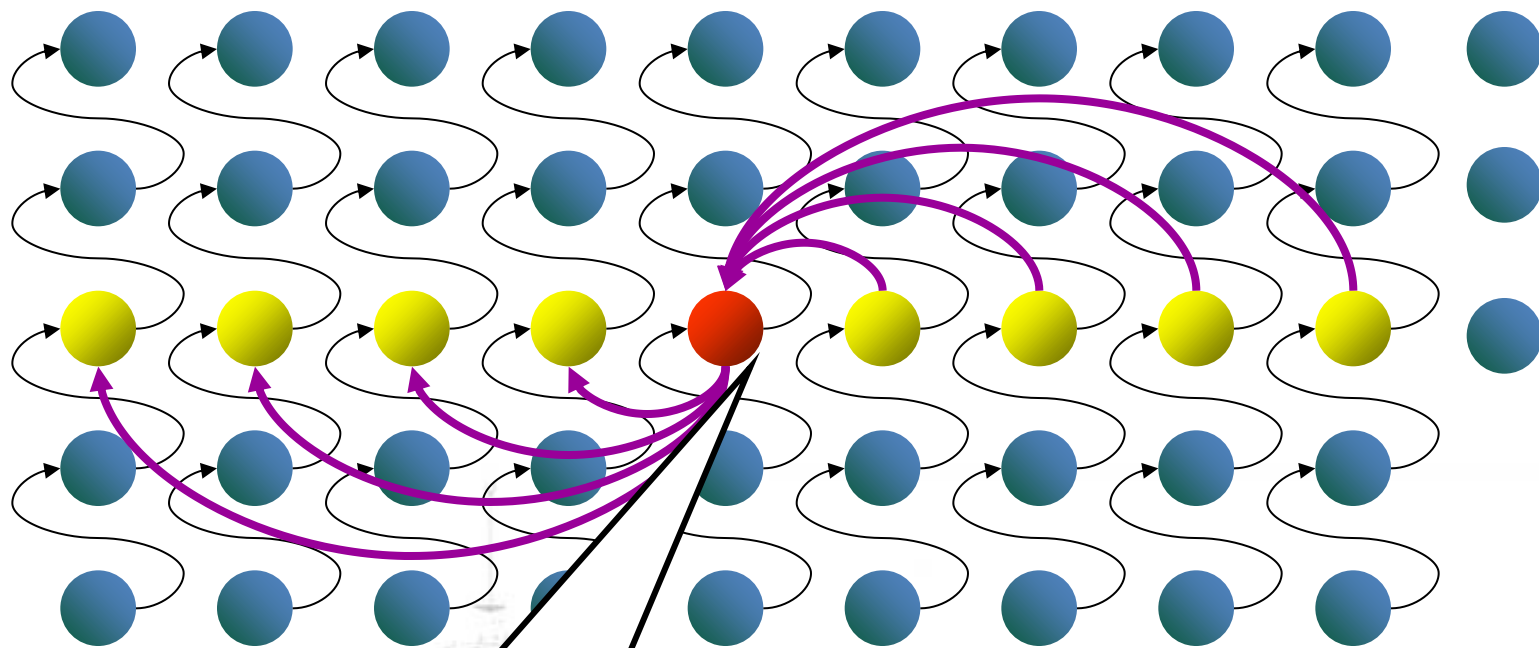


第二步：将每组数分别用InsertionSort排序 选出每组元素的中位数



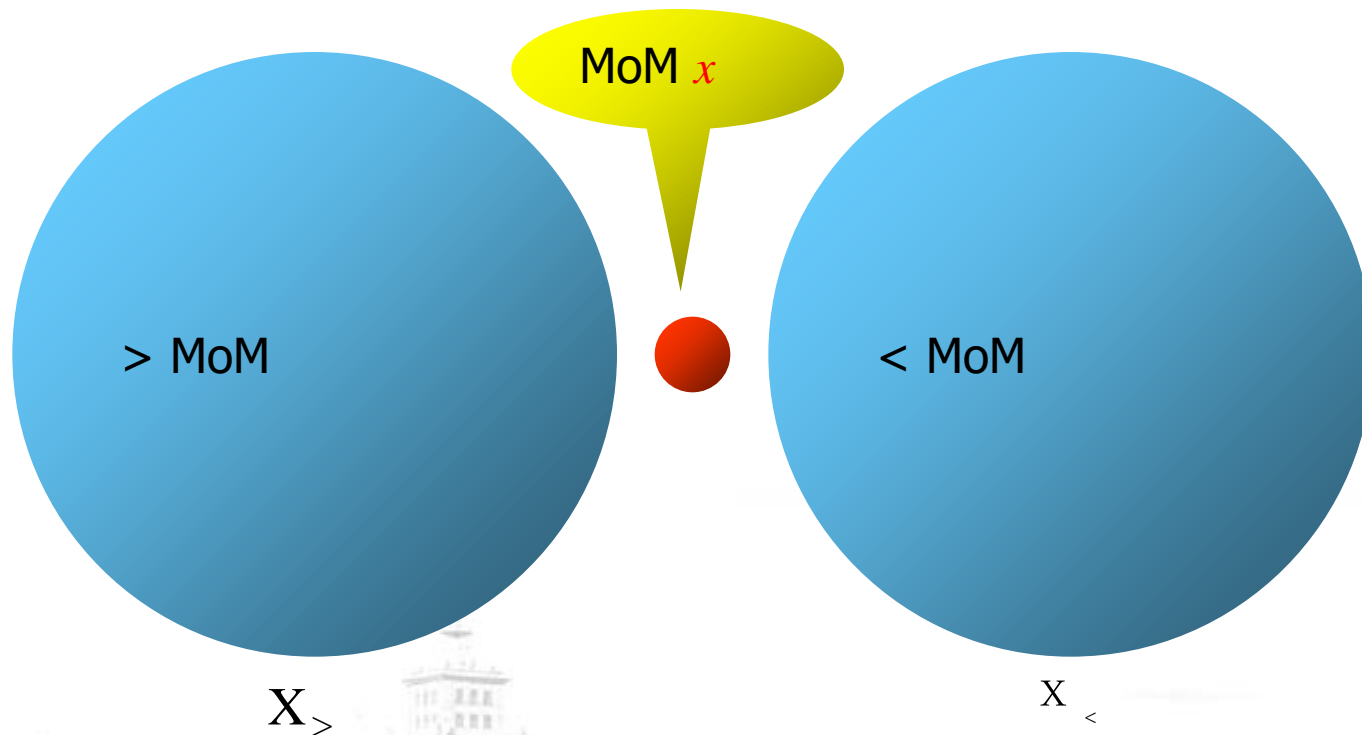
排序每组数时，比较操作的次数为 $5(5-1)/2=10$ 次
总共需要 $10*\lceil n/5 \rceil$ 次比较操作

第三步：递归调用算法求得这些中位数的中位数(MoM)



时间复杂性: $T(\lfloor n/5 \rfloor)$

第四步: 用 MoM 完成划分



时间复杂性 $O(n)$

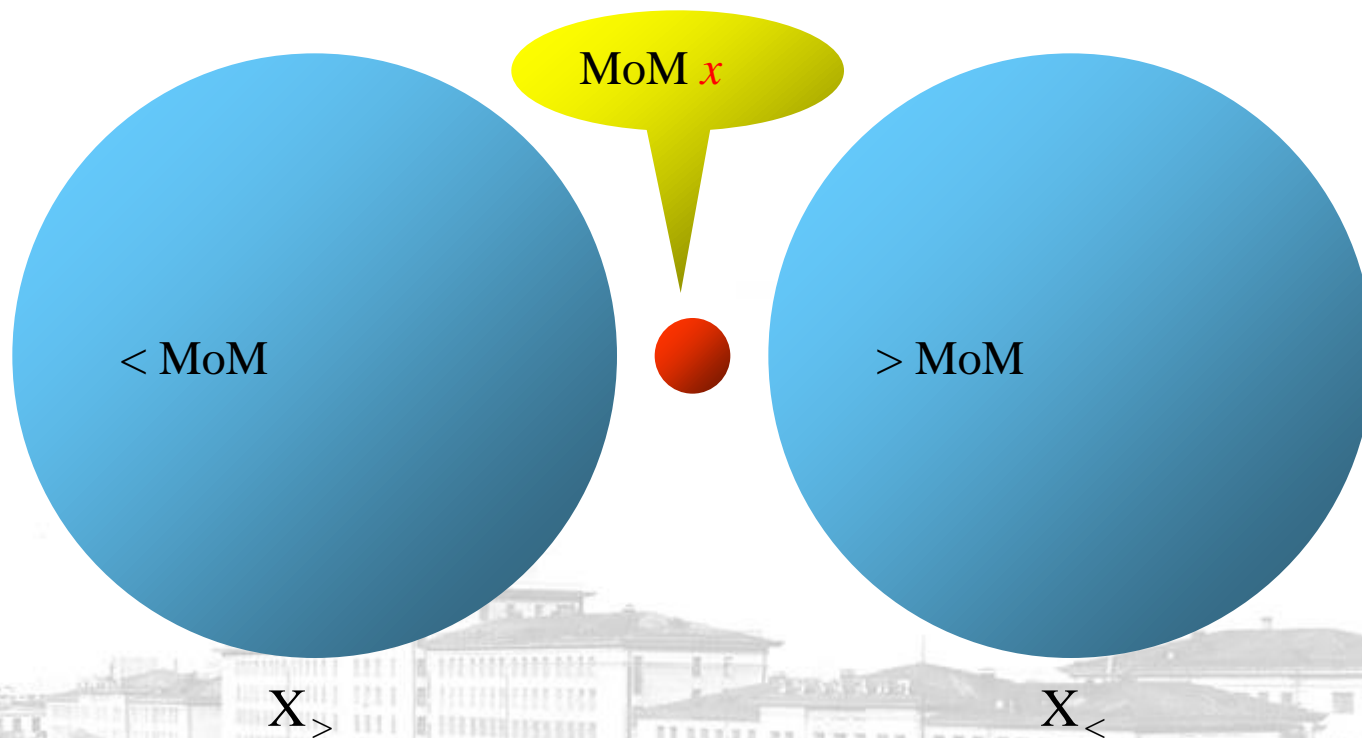
第五步:递归

设 x 是中位数的中位数(MoM),划分完成后其下标为 k

如果 $i=k$,则返回 x

如果 $i < k$,则在第一个部分递归选取第 i -大的数

如果 $i > k$,则在第三个部分递归选取第 $(i-k)$ -大的数



算法Select(A, i)

Input: 数组 $A[1:n]$, $1 \leq i \leq n$

Output: $A[1:n]$ 中的第 i -大的数

1. for $j \leftarrow 1$ to $n/5$ ← 第一步
2. InsertSort($A[(j-1)*5+1 : (j-1)*5+5]$); } 第二步
3. swap($A[j], A[(j-1)*5+3]$); }
4. $x \leftarrow \text{Select}(A[1: n/5], n/10)$; ← 第三步
5. $k \leftarrow \text{partition}(A[1:n], x)$; ← 第四步
6. if $k=i$ then return x ;
7. else if $k>i$ then retrun $\text{Select}(A[1:k-1], i)$; } 第五步
8. else retrun $\text{Select}(A[k+1:n], i-k)$; }



算法分析

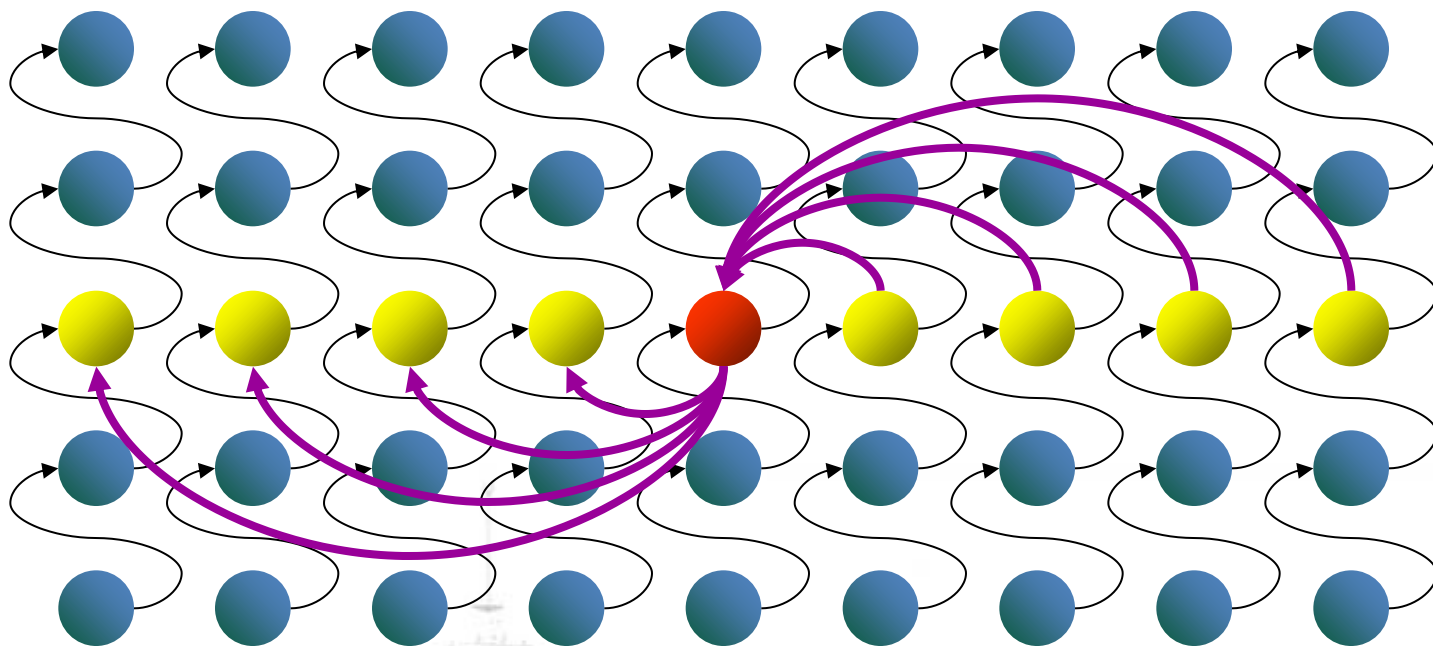
算法Select(A, i)

Input: 数组 $A[1:n]$, $1 \leq i \leq n$

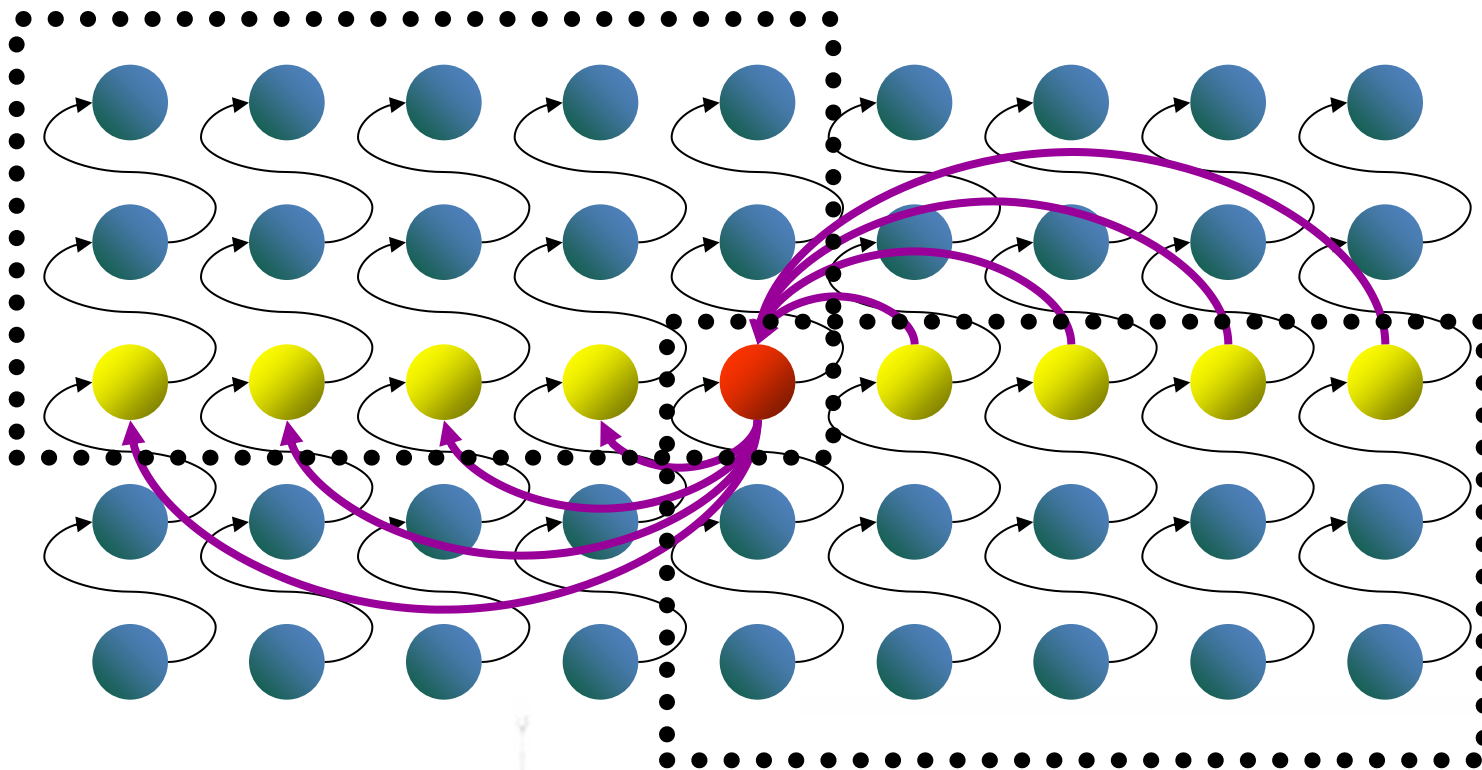
Output: $A[1:n]$ 中的第 i -大的数

1. for $j \leftarrow 1$ to $n/5$
 2. InsertSort($A[(j-1)*5+1 : (j-1)*5+5]$);
 3. swap($A[j], A[(j-1)*5+3]$);
 4. $x \leftarrow \text{Select}(A[1: n/5], n/10);$
 5. $k \leftarrow \text{partition}(A[1:n], x);$
 6. if $k=i$ then return x ;
 7. else if $k>i$ then retrun $\text{Select}(A[1:k-1], i);$
 8. else retrun $\text{Select}(A[k+1:n], i-k);$
- Annotations:
- Lines 2 and 3 are grouped by a brace and labeled $O(n)$.
 - Line 4 has an arrow pointing to it from the label $T(\lfloor n/5 \rfloor)$.
 - Line 5 has an arrow pointing to it from the label $O(n)$.
 - Lines 6, 7, and 8 are grouped by a brace and labeled ???.

观察第五步的处理过程



第五步至少删除了 $\lfloor 3n/10 \rfloor$ 个数



$$n - \lfloor 3n/10 \rfloor \leq 7n/10 + 6$$

如果时间复杂度是输入规模的递增函数
则第五步的时间开销不超过 $T(7n/10 + 6)$

$$T(n) \leq \begin{cases} \Theta(1) & \text{if } n \leq C \\ T\lfloor n/5 \rfloor + T(7n/10 + 6) + \Theta(n) & \text{if } n > C \end{cases}$$

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



本讲内容

- 3.1 分治法
- 3.2 分治法的简单实例
- 3.3 元素选取问题的线性时间算法
- 3.4 快速傅里叶变换

问题定义

输入: a_0, a_1, \dots, a_{n-1} , $n=2^k$, a_i 是实数, $(0 \leq i \leq n-1)$

输出: A_0, A_1, \dots, A_{n-1}

$$A_j = \sum_{k=0}^{n-1} a_k e^{\frac{2jk\pi}{n}i}, \text{ 其中 } j=0, 1, \dots, n-1,$$

e 是自然对数的底数, i 是虚数单位

蛮力法利用定义计算每个 A_j , 时间复杂度为 $\Theta(n^2)$

算法的数学基础

$$\text{令 } \beta_n = e^{\frac{2\pi i}{n}}$$

$$\begin{aligned} A_j &= a_0 + a_1 \beta_n^j + a_2 \beta_n^{2j} + \dots + a_{n-1} \beta_n^{(n-1)j} \\ &= [a_0 + a_2 \beta_n^{2j} + a_4 \beta_n^{4j} \dots + a_{n-2} \beta_n^{(n-2)j}] + \\ &\quad [a_1 + a_3 \beta_n^{2j} + a_5 \beta_n^{4j} \dots + a_{n-1} \beta_n^{(n-2)j}] \beta_n^j \\ &= \left[a_0 + a_2 \beta_{n/2}^j + a_4 \beta_{n/2}^{2j} + \dots + a_{n-2} \beta_{n/2}^{\frac{n-2}{2}j} \right] + \\ &\quad \left[a_1 + a_3 \beta_{n/2}^j + a_5 \beta_{n/2}^{2j} + \dots + a_{n-1} \beta_{n/2}^{\frac{n-2}{2}j} \right] \beta_n^j \end{aligned}$$

第一项内形如 $a_0, a_2, a_4, \dots, a_{n-2}$ 的离散傅里叶变换
第二项内形如 $a_1, a_3, a_5, \dots, a_{n-1}$ 的离散傅里叶变换

$$\text{令 } \beta_n = e^{\frac{2\pi i}{n}}$$

$$A_j = a_0 + a_1 \beta_n^j + a_2 \beta_n^{2j} + \dots + a_{n-1} \beta_n^{(n-1)j} \quad 0 \leq j \leq n-1$$

$$= \left[a_0 + a_2 \beta_{n/2}^j + a_4 \beta_{n/2}^{2j} + \dots + a_{n-2} \beta_{n/2}^{\frac{n-2}{2}j} \right] + \left[a_1 + a_3 \beta_{n/2}^j + a_5 \beta_{n/2}^{2j} + \dots + a_{n-1} \beta_{n/2}^{\frac{n-2}{2}j} \right] \beta_n^j$$

$$B_j = a_0 + a_2 \beta_{n/2}^j + a_4 \beta_{n/2}^{2j} + \dots + a_{n-2} \beta_{n/2}^{((n-2)/2)j} \quad 0 \leq j \leq (n-2)/2$$

$$C_j = a_1 + a_3 \beta_{n/2}^j + a_5 \beta_{n/2}^{2j} + \dots + a_{n-1} \beta_{n/2}^{((n-2)/2)j} \quad 0 \leq j \leq (n-2)/2$$

$$\beta_{n/2}^{kj} = e^{\frac{2\pi i}{n/2}kj} = e^{\frac{2\pi i}{n/2}kj - 2k\pi i} = e^{\frac{2\pi i}{n/2}k(j-n/2)} = \beta_{n/2}^{k(j-n/2)}$$

分治算法过程

划分：将输入拆分成 a_0, a_2, \dots, a_{n-2} 和 a_1, a_3, \dots, a_{n-1} 。

递归求解：递归计算 a_0, a_2, \dots, a_{n-2} 的变换 $B_0, B_1, \dots, B_{n/2-1}$
递归计算 a_1, a_3, \dots, a_{n-1} 的变换 $C_0, C_1, \dots, C_{n/2-1}$

合并： $A_j = B_j + C_j \cdot \beta_n^j \quad (j < n/2)$
 $A_j = B_{j-n/2} + C_{j-n/2} \cdot \beta_n^j \quad (n/2 \leq j < n-1)$



算法

算法FFT

输入: a_0, a_1, \dots, a_{n-1} , $n=2^k$

输出: a_0, a_1, \dots, a_{n-1} 的傅里叶变换 A_0, \dots, A_{n-1}

1. $\beta \leftarrow \exp(2\pi i/n)$;
2. If ($n=2$) Then
3. $A_0 \leftarrow a_0 + a_1$;
4. $A_1 \leftarrow a_0 - a_1$;
5. 输出 A_0, A_1 , 算法结束;
6. $B_0, B_1, \dots, B_{n/2-1} \leftarrow \text{FFT}(a_0, a_2, \dots, a_{n-2}, n/2)$;
7. $C_0, C_1, \dots, C_{n/2-1} \leftarrow \text{FFT}(a_1, a_3, \dots, a_{n-1}, n/2)$;
8. For $j=0$ To $n/2-1$
9. $A_j \leftarrow B_j + C_j \cdot \beta^j$;
10. $A_{j+n/2} \leftarrow B_j - C_j \cdot \beta^{j+n/2}$;
11. 输出 A_0, A_1, \dots, A_{n-1} , 算法结束;

算法分析

$$T(n)=\Theta(1) \quad \text{If } n=2$$

$$T(n)=2T(n/2)+\Theta(n) \quad \text{If } n>2$$

$$T(n)=\Theta(n \log n)$$

