PKU-04833060
算法设计与分析『实验班』

第3讲 分治策略 (1/2)

罗国杰

gluo@pku.edu.cn

2025年春季学期

幂乘计算

问题：设 a 是给定实数，计算 a^n，n 为自然数

朴素算法：$\Theta(n)$

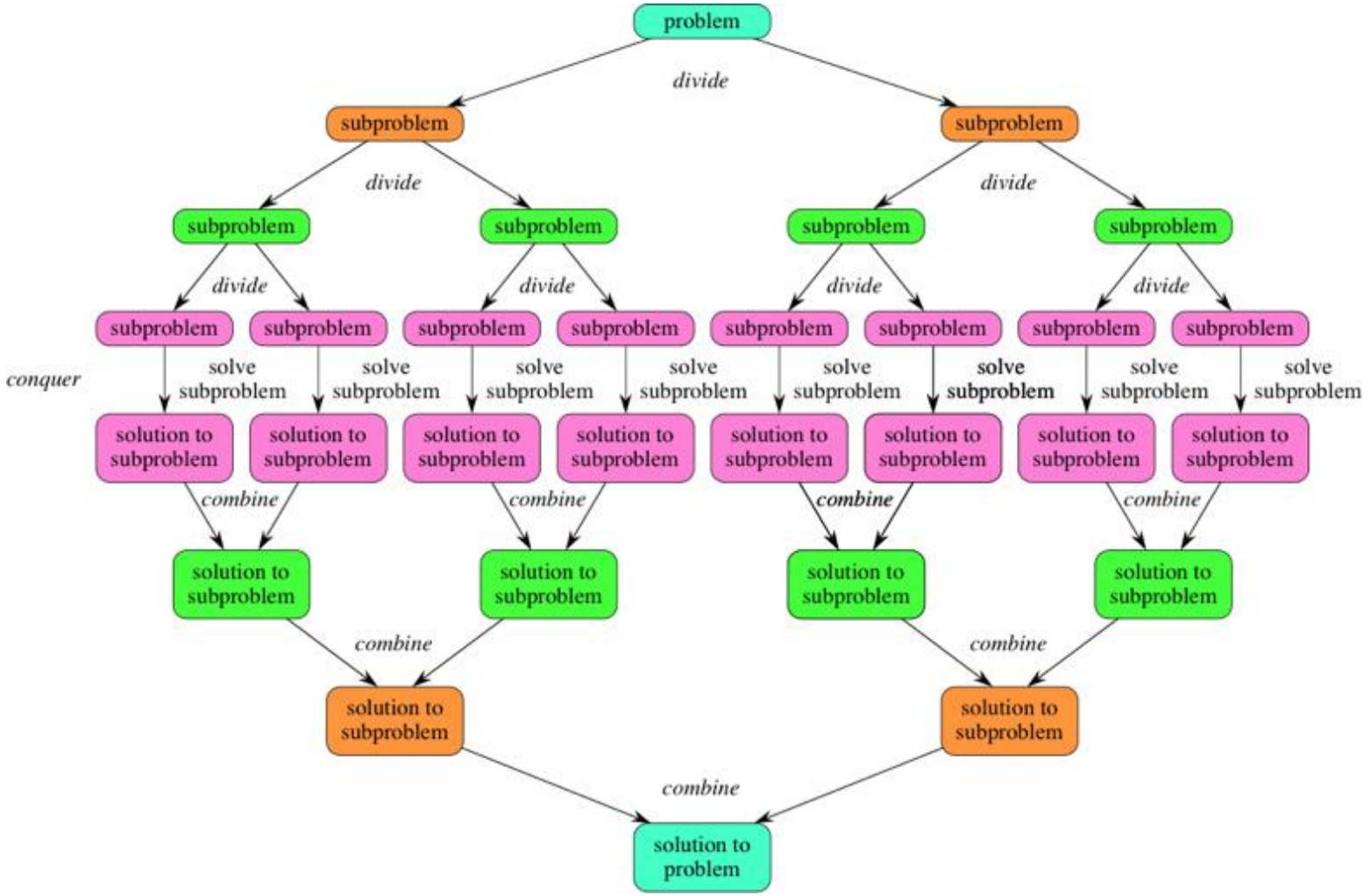
分治法

$$a^n = \begin{cases} a^{n/2} \times a^{n/2} & n \text{ 为偶数} \\ a^{(n-1)/2} \times a^{(n-1)/2} \times a & n \text{ 为奇数} \end{cases}$$

$$W(n) = W(n/2) + \Theta(1) \quad \Rightarrow \quad W(n) = \Theta(\log n) .$$

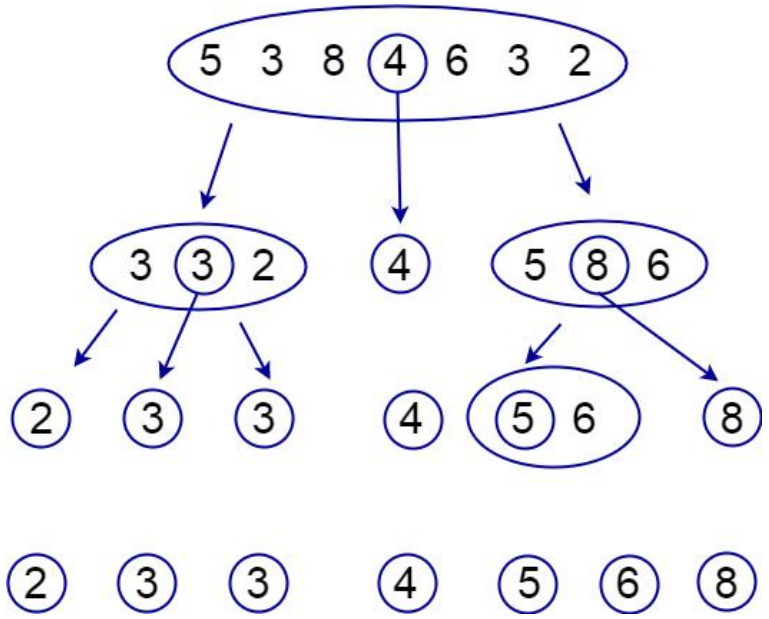
分治策略（Divide-and-Conquer）

- Divide
- Conquer
- Combine

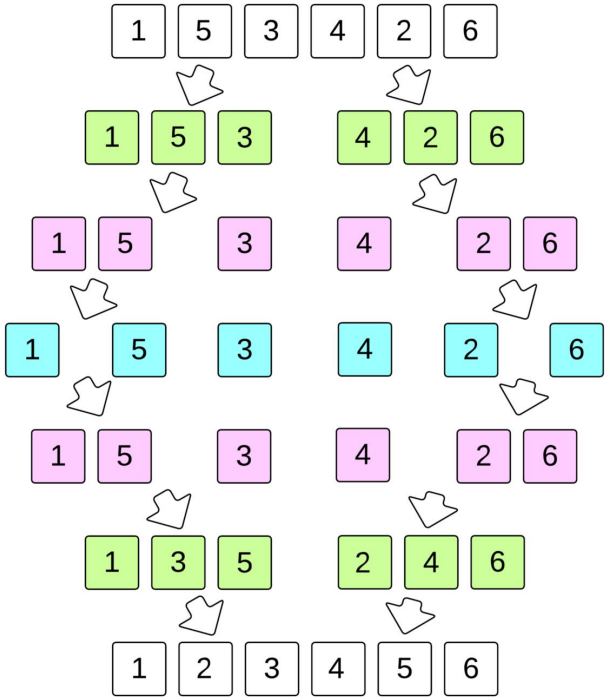


图片来源：khanacademy.org

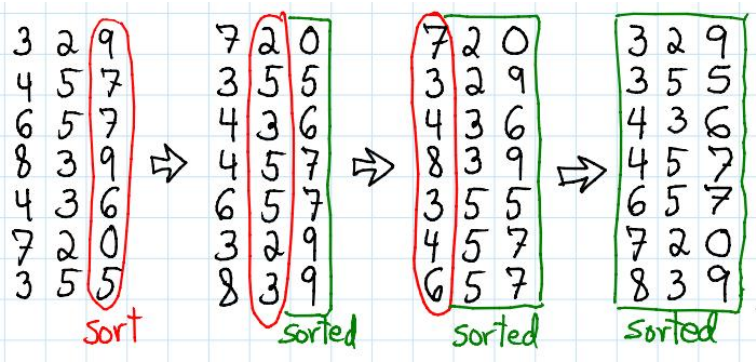
排序问题的分治策略



快速排序



归并排序



基数排序

图片来源：medium.com 和 mit.edu

主要内容

- 分治策略的基本思想-分治算法的一般性描述
- 分治算法的分析技术
- 改进分治算法的途径
  - 通过代数变换减少子问题个数（例子：大数乘法和大矩阵乘法）
  - 利用预处理减少递归内部的计算量（例子：平面最邻近点对）
- 典型实例
  - 选择问题
  - n－1 次多项式在全体 2n 次方根上的求值

分治算法的一般性描述

分治算法 Divide-and-Conquer(P)

```
1.  if  |P| ≤ c  then  return S(P).
2.  (P1, P2, …, Pk) ← Divide(P)
3.  for i = 1 to k
4.      yi = Divide-and-Conquer(Pi)
5.  return Merge(y1, y2, …, yk)
```

算法时间复杂度的递推方程

$$\begin{cases} W(n) = W(|P_1|) + W(|P_2|) + \ldots + W(|P_k|) + f(n) \\ W(c) = C \end{cases}$$

一般原则：子问题均匀划分、递归处理

分治算法的分析技术

分治策略的算法分析工具：递推方程

两类递推方程

$$f(n)=\sum_{i=1}^{k}a_i f(n-i)+g(n)$$

$$f(n)=af(\frac{n}{b})+d(n)$$

求解方法

第一类方程：迭代法、换元法、递归树、尝试法、……

第二类方程：迭代法、递归树、主定理、……

递推方程的解

方程 $T(n) = aT(n/b) + d(n)$

$d(n)$为常数

$$T(n) = \begin{cases} O(n^{\log_b a}) & a \neq 1 \\ O(\log n) & a = 1 \end{cases}$$

$d(n) = cn$

$$T(n) = \begin{cases} O(n) & a < b \\ O(n\log n) & a = b \\ O(n^{\log_b a}) & a > b \end{cases}$$

数逆序对

- “XX云音乐”在开发好友推荐功能，推送有类似音乐偏好的用户
  - 对于 n 首音乐，你的偏好（收听次数、打分等等）从高到低是 1, 2, ..., n
  - 用户 Z 的对同样 n 首音乐的偏好是 $a_1, a_2, ..., a_n$ （1 .. n 的一个排列）
  - 将此排列的“逆序对”数目定义为你和 Z 偏好的距离
    - 直观上，逆序对越多、偏好差异越大
    - (a_i, a_j) 是逆序对，当且仅当 $i < j$ 和 $a_i > a_j$
    - 例如 4, 6, 1, 7, 3, 2, 5 有 11 组逆序对
- 问题：给定用户 Z 的偏好，计算 Z 与你的音乐偏好的距离
- 轻松实现 $O(n^2)$ 的朴素算法。存在时间更短的算法吗？

Kandell's Tau (Kendall Rank Correlation Coefficient)

- Kendall, M.G.: A new measure of rank correlation. Biometrika 30, 81–93 (1938)
- a nonparametric measure of correlation between random variables X and Y
  - observations (X1, Y1), (X2, Y2), ..., (Xn, Yn)
- $\tau = (C-D)/(C+D) = 1 - 2D/(C+D)$
  - C: num. of concordant paris
  - D: num. of discordant pairs (~ the inversion number to permute Y to obtain X)
  - C+D = n(n-1)/2
- $-1 \leq \tau \leq +1$
  - +1: perfect agreement
  - -1: perfect disagreement
  - exp[τ] = 0 for independent X and Y

(2008). Kendall Rank Correlation Coefficient. In: The Concise Encyclopedia of Statistics. Springer, New York, NY. doi: 10.1007/978-0-387-32833-1_211

数逆序对

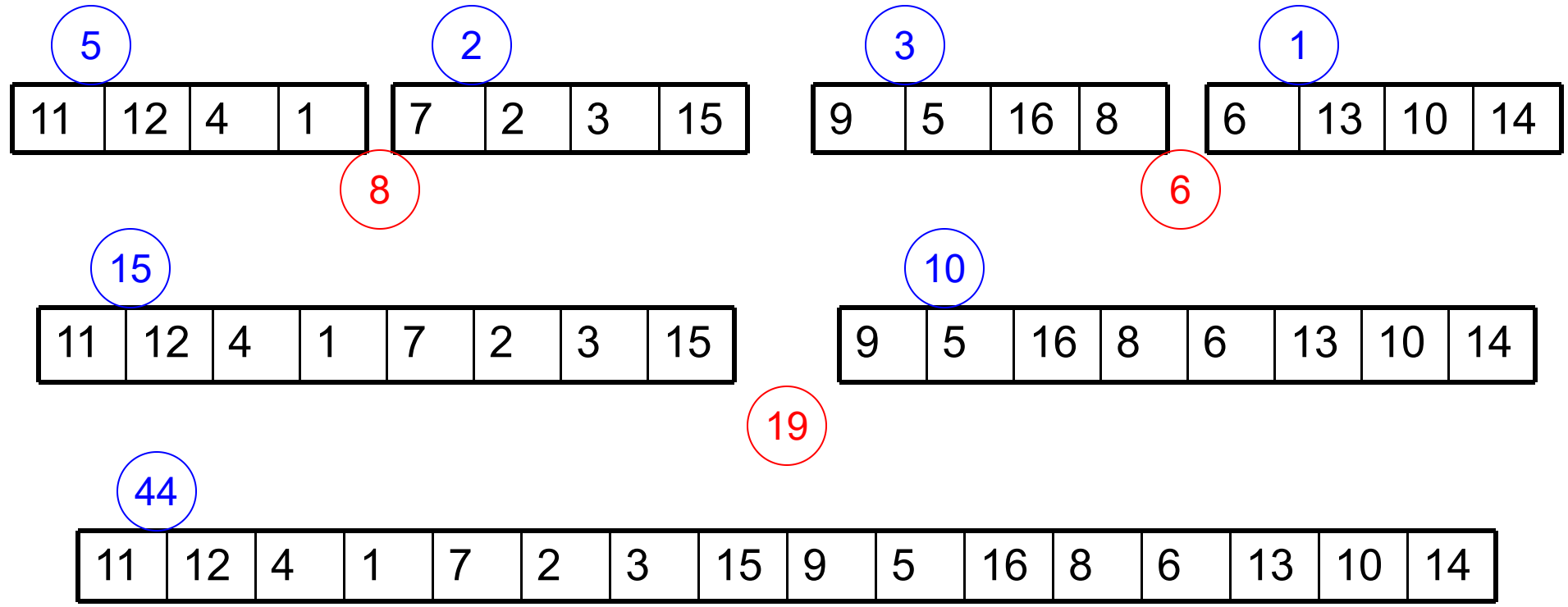
11	12	4	1	7	2	3	15	9	5	16	8	6	13	10	14

数前半段内的逆序对

数后半段内的逆序对

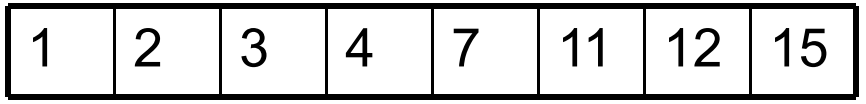
数跨前后两段的逆序对

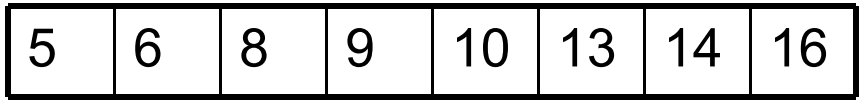
数逆序对

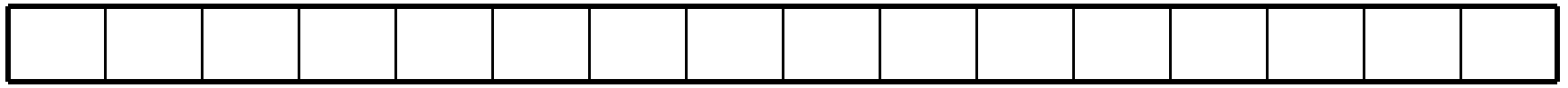

5
11 12 4 1
8
2
7 2 3 15
3
9 5 16 8
6
1
6 13 10 14
15
11 12 4 1 7 2 3 15
10
9 5 16 8 6 13 10 14
19
44
11 12 4 1 7 2 3 15 9 5 16 8 6 13 10 14

数逆序对：如何用 O(n) 时间数子问题间的逆序对

- 解决方案：利用归并排序算法数逆序对

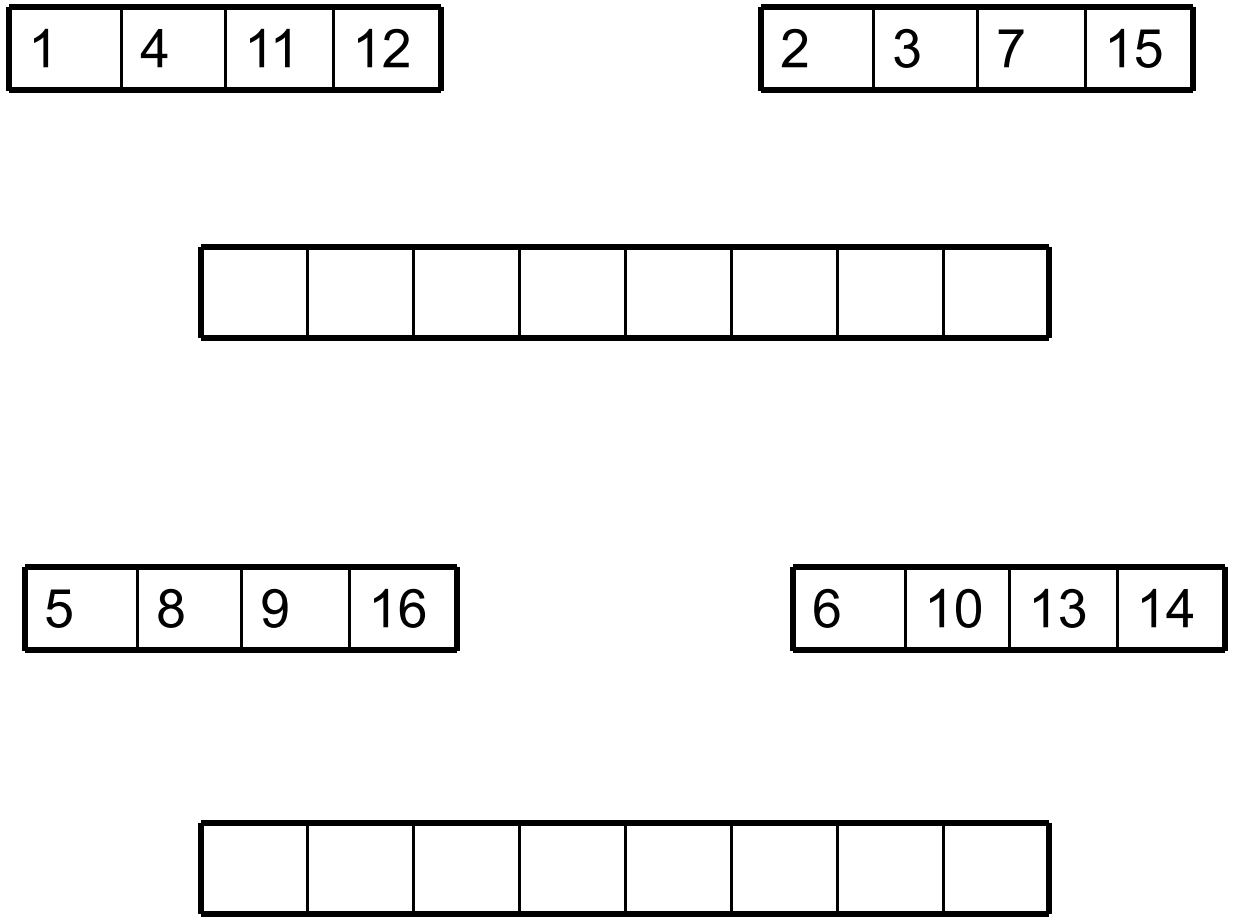






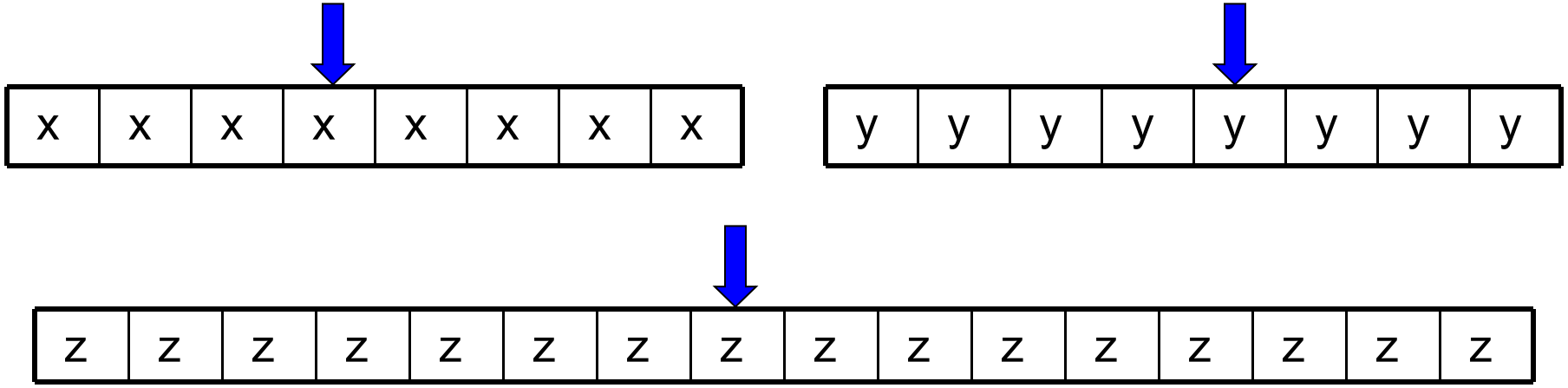
在标准归并排序算法的归并步骤增加数逆序对的操作

数逆序对：利用归并排序算法



数逆序对

- 前后两个有序列表间的逆序对数目
  - 每个元素增加 O(1) 时间的计数操作



- 算法总结
  - 满足标准的分治递归结构
  - T(n) = 2 T(n/2) + cn

数逆序对的应用

- Voting theory.
- Collaborative filtering.
- Measuring the “sortedness” of an array.
- Sensitivity analysis of Google’s ranking function.
- Rank aggregation for meta-searching on the Web.
- Nonparametric statistics (e.g., Kendall’s tau distance).

Rank Aggregation Methods for the Web

Cynthia Dwork* Ravi Kumar† Moni Naor‡ D. Sivakumar§

ABSTRACT

We consider the problem of combining ranking results from various sources. In the context of the Web, the main applications include building meta-search engines, combining ranking functions, selecting documents based on multiple criteria, and improving search precision through word associations. We develop a set of techniques for the rank aggregation problem and compare their performance to that of well-known methods. A primary goal of our work is to design rank aggregation techniques that can effectively combat “spam,” a serious problem in Web searches. Experiments show that our methods are simple, efficient, and effective.

Keywords: rank aggregation, ranking functions, meta-search, multi-word queries, spam

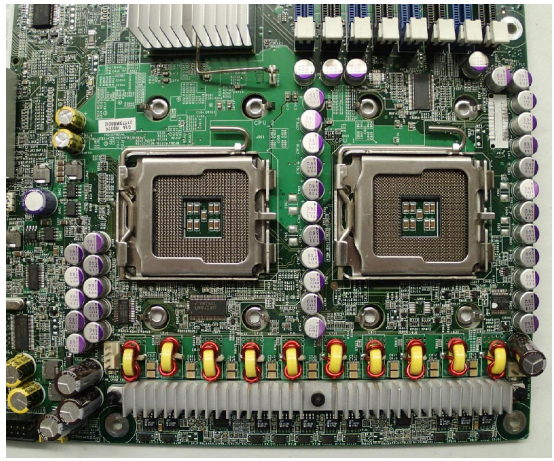
参考资料：https://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/05DivideAndConquerI.pdf

芯片测试 (CLRS 3rd ed. Ex. 4-5; or 4th ed. Ex. 4-6)

条件：有n片芯片，(好芯片至少比坏芯片多1片).
问题：使用0(n)测试次数，从中挑出1片好芯片.

对芯片A与B测试，结果分析如下：

A 报告	B 报告	结论
B是好的	A是好的	A,B 都好或 A,B 都坏
B是好的	A是坏的	至少一片是坏的
B是坏的	A是好的	至少一片是坏的
B是坏的	A是坏的	至少一片是坏的



假想的芯片互测平台

图片来源：ebay.com

算法策略：两两一组测试，淘汰后芯片进入下一轮.
如果测试结果是情况1，那么A、B中留1片，丢1片；
如果是后三种情况，则把A和B全部丢掉.

芯片测试：分治算法

命题 当n是偶数时，在上述规则下，经过一轮淘汰，剩下的好芯片比坏芯片至少多1片.

证 设A与B都是好芯片有 i 组，A与B一好一坏有 j 组，A与B都坏有 k 组，淘汰后，好芯片数 i，坏芯片数 k

$$2i + 2j + 2k = n$$

$$2i+j > 2k+j \Rightarrow i > k$$

当n是奇数时，用其他芯片测试轮空芯片，如果轮空芯片是好的（情况1次数达到(n-1)/2次），算法结束；否则淘汰轮空芯片.

每轮淘汰后，芯片数至少减半，时间复杂度是：

$$\begin{cases} W(n) = W(\frac{n}{2}) + O(n) & n > 3 \\ W(n) = 1 & n \le 3 \end{cases} \Rightarrow \quad W(n) = O(n)$$

芯片测试：伪码描述

算法 Test(n)

```
1.   k ← n
2.   while   k >3   do
3.        将芯片分成⌊k/2⌋组     // 如有轮空芯片，特殊处理
4.        for   i =1   to  ⌊k/2⌋  do
5.             if   报告2片好   then   任取1片留下
6.             else   2片同时丢掉
7.        k ← 剩下的芯片数
8.   if   k = 3
9.        then   任取2片芯片测试
10.            if   1好1坏   then   取没测的芯片
11.            else   任取1片被测芯片
12.  if   k = 2 or 1   then   任取1片
```

选择问题

问题：从给定的集合 L 中选择第 i 小的元素

不妨设 L 为 n 个不等的实数

i=1，称为最小元素；

i=n，称为最大元素；

i=n-1，称为第二大元素；

位置处在中间的元素，称为中位元素

当n为奇数时，中位数只有1个，i=(n+1)/2；

当n为偶数时，中位数有2个，i=n/2, n/2+1. 也可以规定其中的一个

一般性选择问题

- 问题：选第 k 小.
  - 输入：数组 S, S的长度 n, 正整数 k, $1 \le k \le n$.
  - 输出：第 k 小的数
- 朴素算法

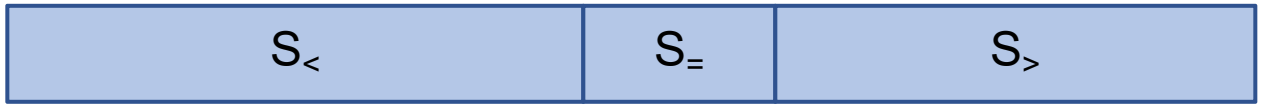
  1．排序

  2．找第 k 小的数

  时间复杂性：O(nlogn)
- 线性时间算法：O(n)
  - 基本思想：减少比较运算的次数
  - 假如 x* < a, a < b, a < c，没必要比较 b 和 c
  - 例：期望O(n)复杂度的快速选择算法
    - 比较 n-1 次，得到 $(n-1) + n_L \times n_R$ 个大小关系
  - 例：确定性O(n)复杂度的分治选择算法

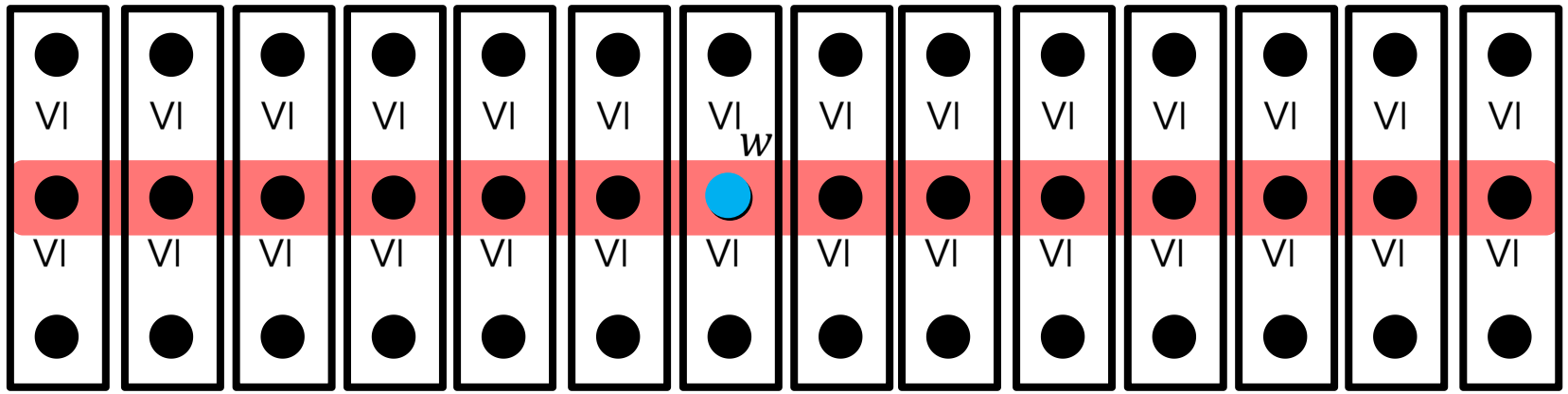
快速选择算法 QSelect(A, k)

```
QSelect(A, k){
    Choose element w from A
    S< = {x in A | x < w}
    S> = {x in A | x > w}
    S= = {x in A | x = w}
    if (|S<| >= k)
        return QSelect(S<, k)
    else if (|S<| + |S=| >= k)
        return w
    else
        return QSelect(S>, k - |S<| - |S=|)
}
```

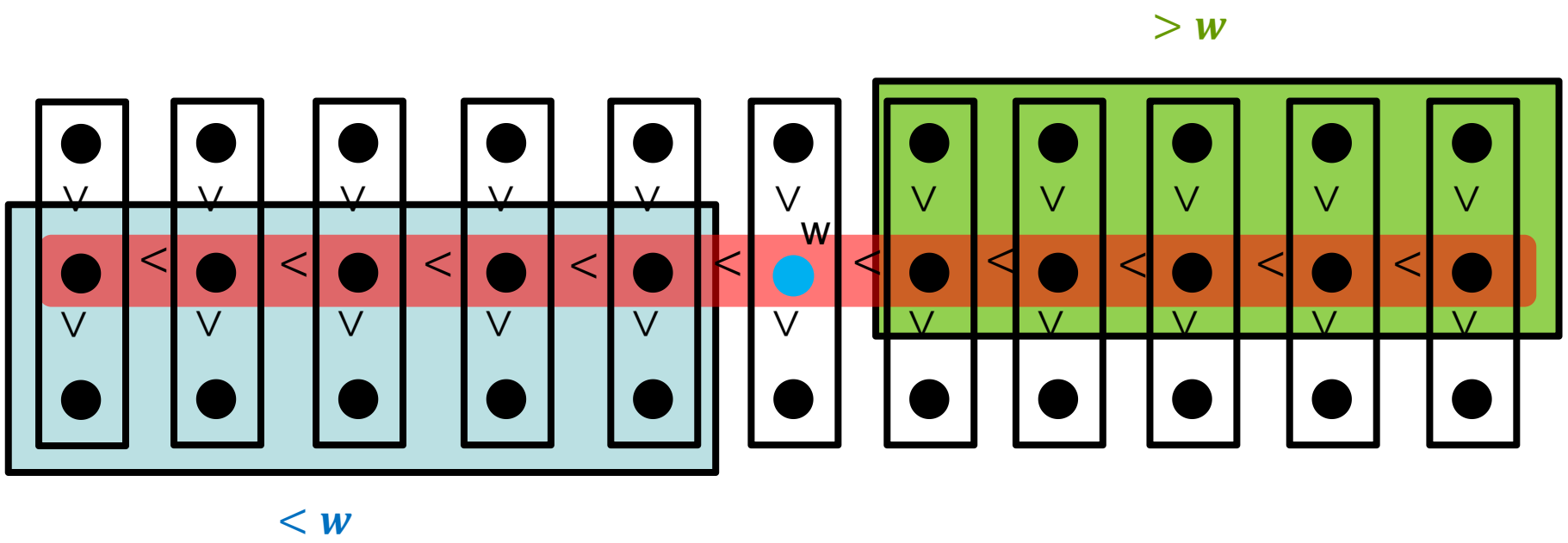


确定性选择算法：如何选择元素w?

- 尝试
  - 将数字分成 n/3 份
  - midpoints = { 每份3个元素的中位数 }，共需O(n)时间
  - w = Select(midpoints, (n/3)/2)



确定性选择算法：如何估算 |S<| 和 |S>| 的下界?



- $|S_<| \geq 2 \times \left(\frac{n}{6}\right) = \frac{n}{3}$
- $|S_>| \geq 2 \times \left(\frac{n}{6}\right) = \frac{n}{3}$.

$$\frac{n}{3} \leq |S_<|, |S_>| \leq \frac{2n}{3}$$

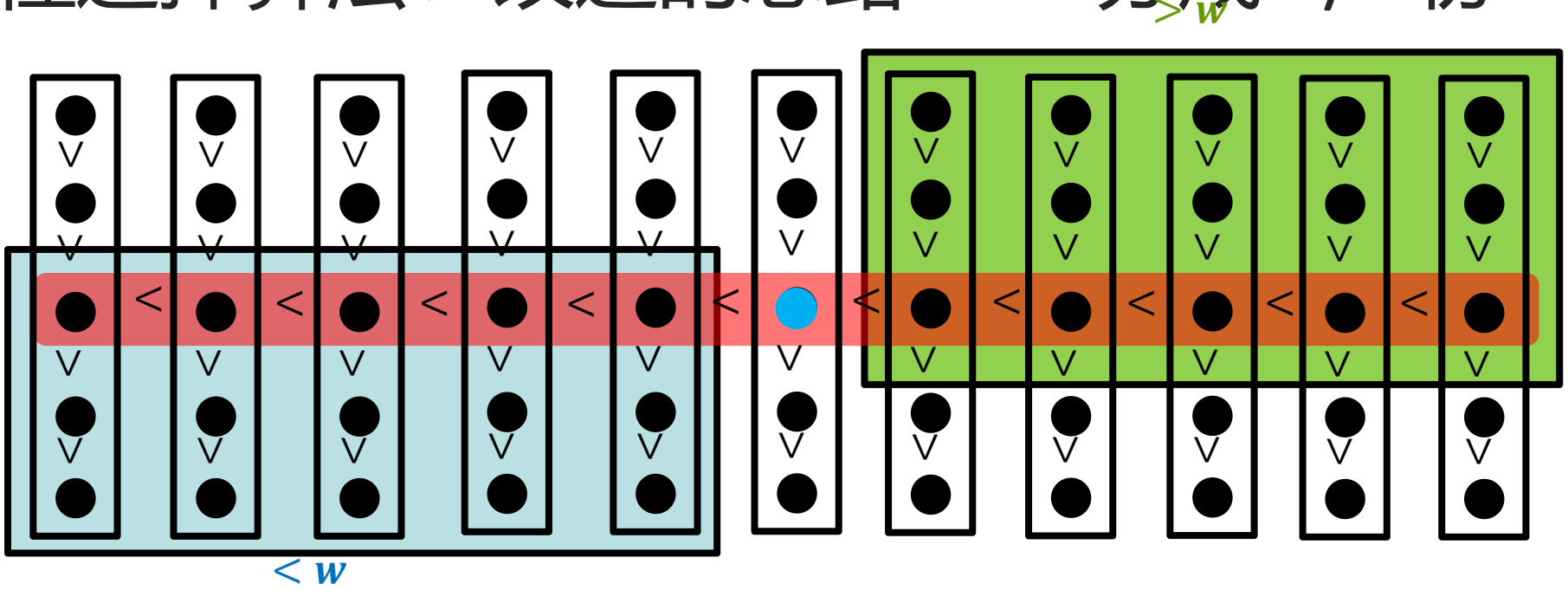
$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + O(n) \Rightarrow T(n) = O(n \log n)$$

确定性选择算法：时间复杂度分析

- $T(n)=T(n/3)+T(2n/3)+n \Rightarrow T(n)=O(n \log n)$
- 第一层时间 n
- 第二层时间 n/3+2n/3，跟第一层同阶
- 每层类似情况。因此，总时间 n log n

- 问题1：n/3 怎么来?
- 问题2：如何减少?
- 问题3：能否实现 O(n) 时间?

确定性选择算法：改进的思路 —— 分成 n/5 份



- $|S_<(w)| \geq 3 \times \left(\frac{n}{10}\right) = \frac{3n}{10}$
- $|S_>(w)| \geq 3 \times \left(\frac{n}{10}\right) = \frac{3n}{10}$

$$\Rightarrow \quad \frac{3n}{10} \leq |S_<(w)|, |S_>(w)| \leq \frac{7n}{10}$$

$$T(n) = T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) + O(n) \Rightarrow T(n) = O(n)$$

确定性O(n)的分治选择算法

算法 Select(S,k)

输入：数组 S，正整数 k

输出：S 中的第 k 小元素

1．将 S 划分成 5 个一组，共$\lceil n/5 \rceil$个组

2．每组找一个中位数，所有个中位数放到集合 M

3．$m^* \leftarrow \text{Select}(M, \lceil |M|/2 \rceil)$　// 将 S 划分成 A,B,C,D 四个集合

4．把A和D的每个元素与m^*比较，小的构成 $S_<$, 大的构成 $S_>$

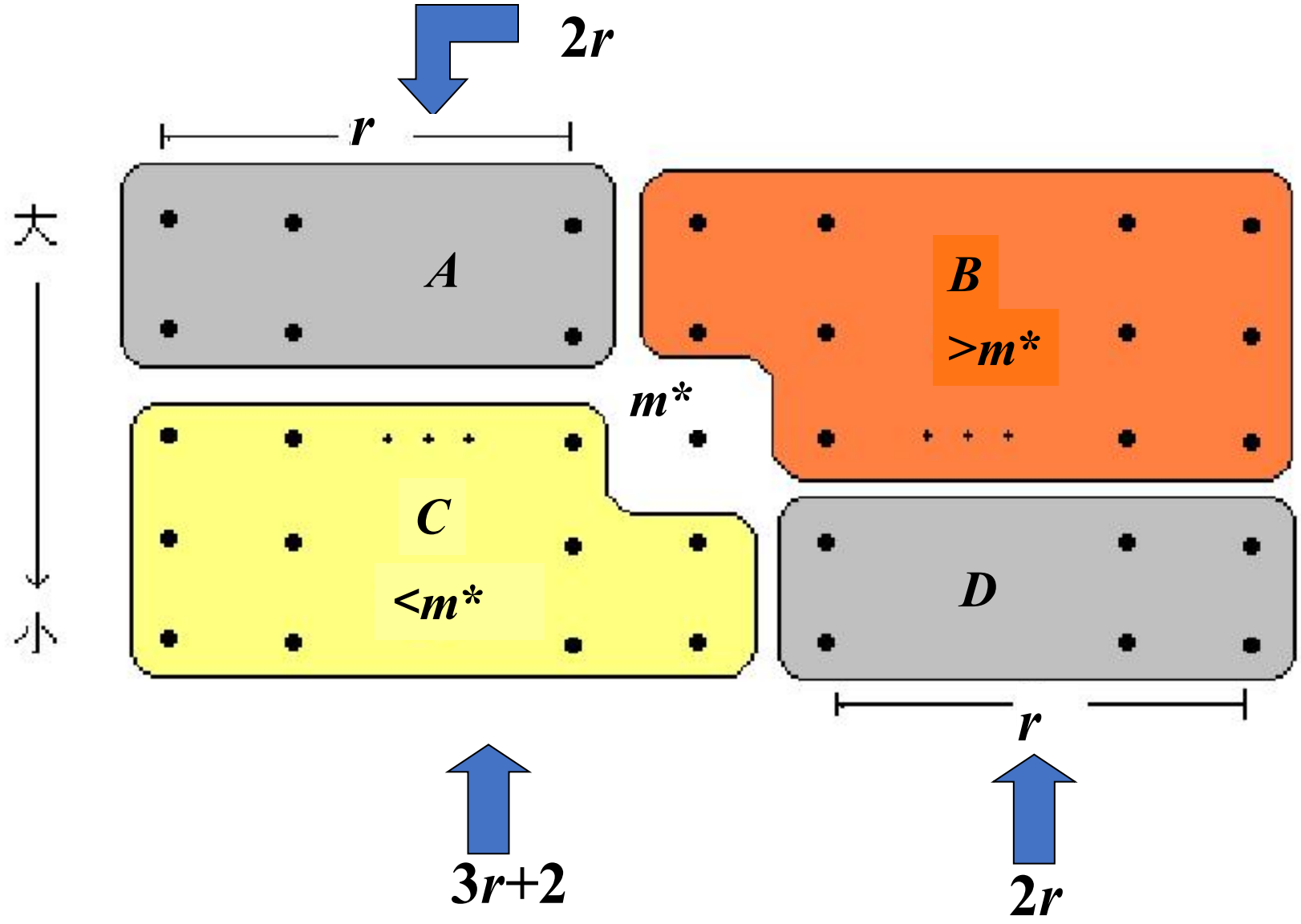
5．$S_< \leftarrow S_< \cup C$; $S_> \leftarrow S_> \cup B$

6．if $|S_<| < k \le |S|-|S_>|$ then 输出m^*

7．else if $k \le |S_1|$

8．　then Select($S_<, k$)

9．　　else Select($S_>, k - (|S|-|S_<|)$)



最坏情况：子问题大小为 $2r + 2r + 3r + 2 = 7r + 2$

分治选择算法：复杂度估计 W(n)=O(n)

不妨设 $n=5(2r+1)$, $|A|=|D|=2r$, $r=\frac{\frac{n}{5}-1}{2}=\frac{n}{10}-\frac{1}{2}$

算法工作量

行2： $O(n)$

行3： $W(n/5)$

行4： $O(n)$

行8-9： $W(7r+2)$

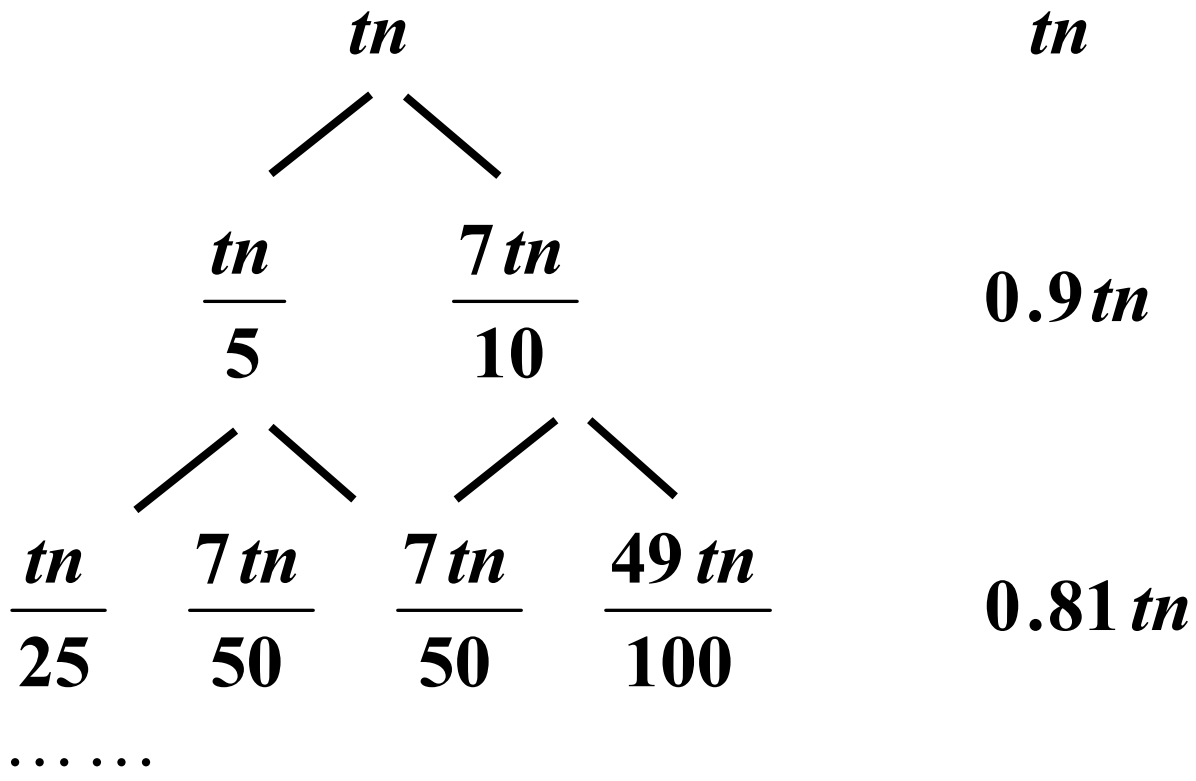
$$W(7r+2)=W(7(\frac{n}{10}-\frac{1}{2})+2)$$

$$=W(\frac{7n}{10}-\frac{3}{2})\le W(\frac{7n}{10})$$

用递归树做复杂度估计

$$w(n)\le W(\frac{n}{5})+W(\frac{7n}{10})+tn\le tn+\frac{9}{10}tn+\frac{81}{100}tn+\ldots=O(n)$$

分治选择算法：递归树



确定性O(n)的分治选择算法

- “BFPRT 算法”
  - Blum, Turing Award 1995
  - Floyd, Turing Award 1978
  - Pratt, KMP 算法的 P
  - Rivest, CLRS 的 R, Turing Award 2002
  - Tarjan, Turing Award 1986 (with Hopcroft)



1936-2001


Time Bounds for Selection*

MANUEL BLUM, ROBERT W. FLOYD, VAUGHAN PRATT,
RONALD L. RIVEST, AND ROBERT E. TARJAN

Department of Computer Science, Stanford University, Stanford, California 94305

Received November 14, 1972

The number of comparisons required to select the i-th smallest of n numbers is shown to be at most a linear function of n by analysis of a new selection algorithm—PICK. Specifically, no more than $5.430\dot{5}\,n$ comparisons are ever required. This bound is improved for extreme values of i, and a new lower bound on the requisite number of comparisons is also proved.

* linear time bounds for median [STOC’72]
* linear time bounds for selection [JCSS’73]

理论：线性时间
* BFPRT优化版 ≤ 5.4305 n 次比较
* 上界 [Dor–Zwick 1995] ≤ 2.95 n 次比较
* 下界 [Dor–Zwick 1999] ≥ $(2 + 2^{-80})$ n 次比较

实践：常数有点大