算法设计与分析

➢ LECTURE 4

Lecture 4

选择

Outline

- 选择算法
  - 选择最大、最小
  - 选择第二大
  - 选择中位数
- 对手论证分析
  - 选择问题的下界

选择问题

- 问题定义
  - 假设E是一个包含n个元素的数组（两两可比较且各不相同），选择问题是找出第k大的元素。

- Special cases
  - 找最大最小—k=1 or k=n
  - 找中位数—k=n/2

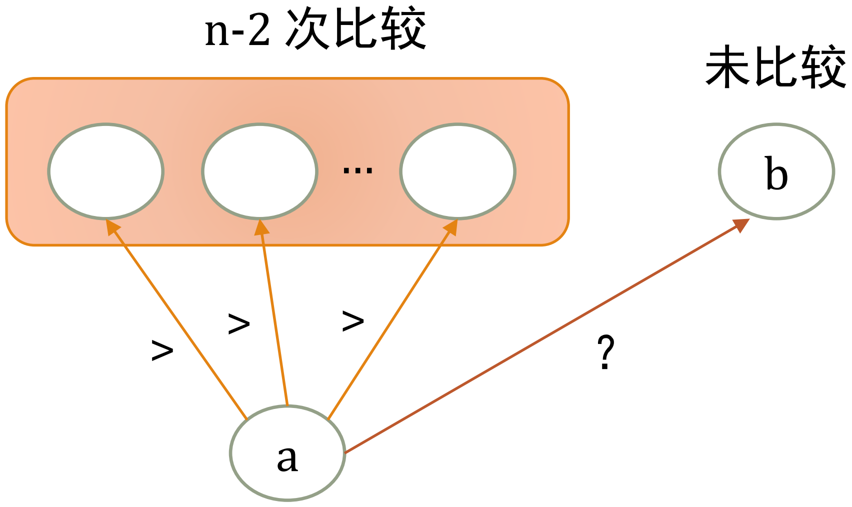
选择最大或最小元素

- 蛮力方法：顺序查找
- 时间代价：n-1

- 定理 对于选择最大最小元素问题，基于比校的选择算法的最坏情况时间复条度是 n-1，即任何基于比较的选择算法，在最坏情况下至少要做 n-1 次比较才能选出最大最小元素。

选择最大或最小元素

- 证明 要证明至少需要 n -1次比较,等价于证明如果比较次数小于等于n-2,则算法一定不正确。我们使用反证法。假设对于任意合法的输入，一个算法总是能正确找到最大元素, 并假没算法至多只需要n-2次比较。



同时选择最大和最小元素

- 蛮力算法
  - 两次遍历查找
  - 代价：$n-1+n-2=2n-3$
- 改进算法
  - 元素两两配对比较
  - 较大元素中选择最大的元素，较小元素中选择最小的元素
  - 代价：$n/2+2(n/2-1)=3n/2-2$
- 下界证明？

对手论证

- 设计者
  - 优势：可以采用任何理论、技术来优化算法的设计，提升算法性能。
  - 劣势：对于所有合法的输入，都必须保证算法的输出是正确的。
- 对手
  - 在所有合法的输入中，挑选对于算法最不利的“坏”输入，使算法付出更多的代价。

- 对手论证基本原理
  - 通过对手论证构造坏的输入，使任何算法总是至少付出一定的代价，而这一“至少要付的代价”，就是我们要求的算法问题的下界。

信息量模型

- 信息量模型：设计一个抽象的数学模型，来刻画所有可能算法的共性特征。
  - 我们将两个元素的比较形象地称为一次比赛，较大的元素称为赢者，较小的元素称为输者。
  - 最大的元素必定赢过 n-1 个人
  - 最小的元素必定输过 n-1 个人

- N(New): 一个元素尚未参加任何比较
- W(Win): 曾经在某次比较中赢过
- L(Lose): 曾经在某次比较中输过
- WL: 曾经在某次比较中赢过，并且曾经在某次比较中输过

一个元素的状态必然是以上四种之一。

信息量模型

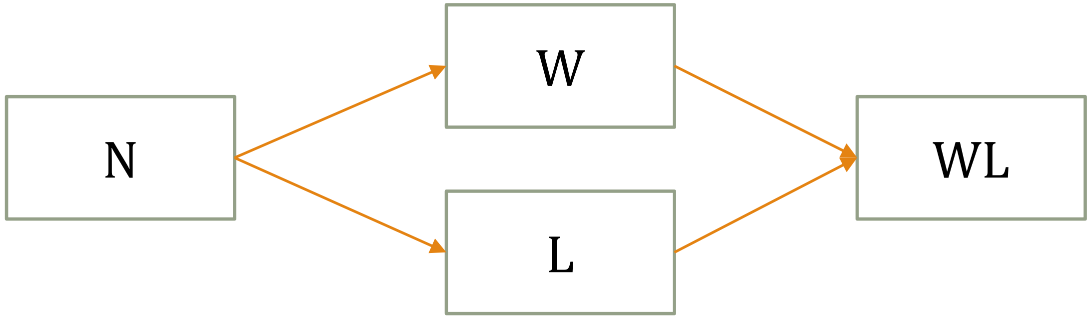


图 1 选择最大和最小元素过程中元素状态的变化

信息量：每经过一条边进行一次状态转换，我们记为增加了 1 个单位的信息量。

确定最大或最小元素，至少要获得 2n-2 的信息量

基于信息量模型的对手策略设计

- 设计者：如何能够使每一次比较尽量获得更多的信息量。
- 对手：挑选合法“坏”输入的准则就是每次比较都尽量少地提供信息量。

比较前的状态	对手的应对	比较后的状态	获得的信息量
N, N	$x>y$	W, L	2
(W, N) 或 (WL, N)	$x>y$	(W, L) 或 (WL, L)	1
L, N	$x<y$	L, W	1
W, W	$x>y$	W, WL	1
L, L	$x>y$	WL, L	1
(W, L) 或 (WL, L) 或 (W, WL)	$x>y$	不变	0
WL, WL	与前面已分配的值一致	不变	0

对手策略分析

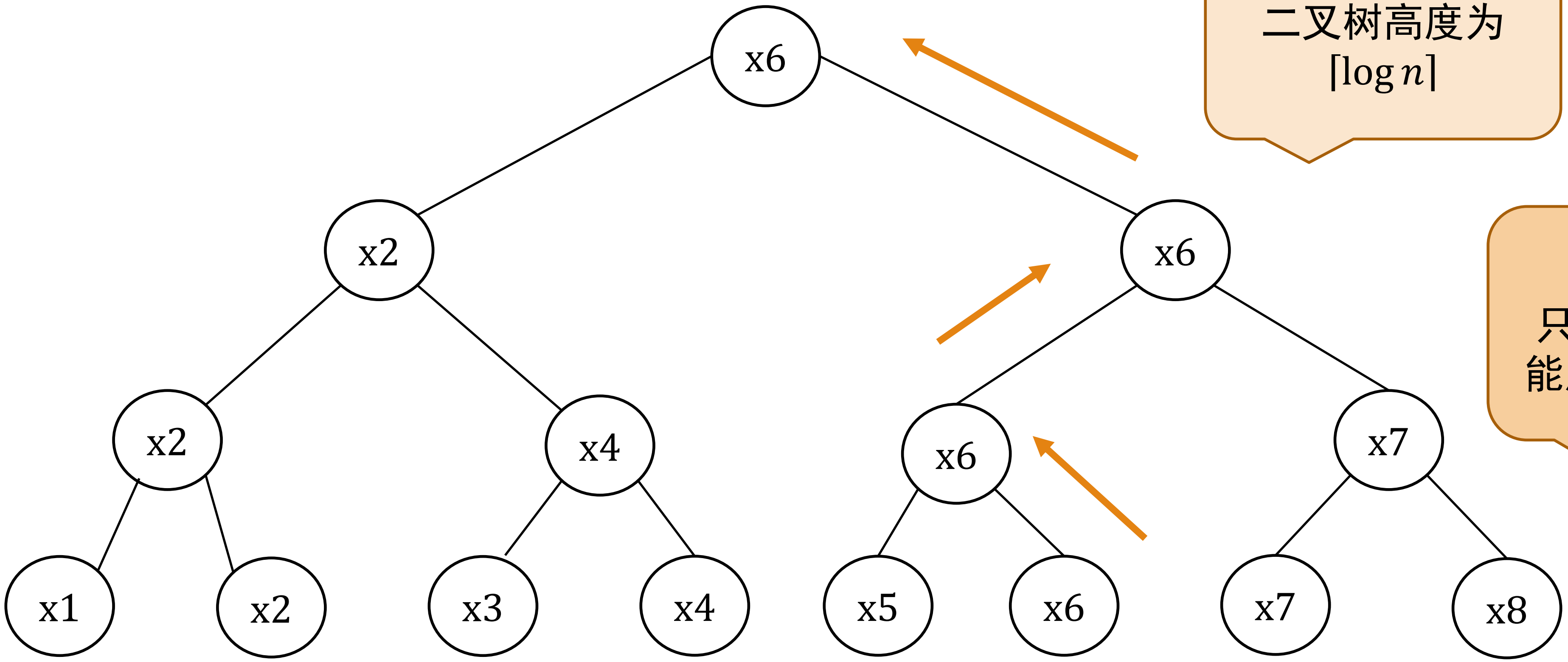
- 首先做获得2个信息量的比较，最多能做 n/2 次
- 再做获得1个信息量的比较
- 总共必须获得至少 2(n-1) 的信息量
- 下界：n/2+n-2=3n/2-2

$$\underbrace{2}_{\text{每次比较获得的信息量}} \times \underbrace{\frac{n}{2}}_{\text{比较的次数}} + \underbrace{1}_{\text{每次比较获得的信息量}} \times \underbrace{(n-2)}_{\text{比较的次数}} = 2n-2$$

选择次大元素

- 蛮力算法-找最大元素两次
  - 需要 2n-3 次比较
- 改进算法
  - 从第一次选择中寻找更多有用的信息
- 观察
  - 输给过不是max的元素不可能是第二大元素

单败淘汰



二叉树高度为
$\lceil \log n \rceil$

x6为最大
只有 x2, x7, x5 可
能成为第二大元素

代价分析

- 任何算法找第二大元素必须先找最大元素
  - 代价：n-1
- 第二大元素只有可能是那些直接输给最大元素的
- 最大元素最多打败了 $\lceil \log n \rceil$ 个元素
- 从这些元素中找最大元素
  - 代价： $\lceil \log n \rceil$ -1
- 总代价：

$$W(n) = \underbrace{n-1}_{\text{选出最大元素的代价}} + \underbrace{\lceil \log n \rceil - 1}_{\text{选出次大元素的代价}} = n + \lceil \log n \rceil - 2$$

下界证明

- **定理** 选择第二大元素的算法的下界是 $n + \lceil \log n \rceil - 2$，即任意一个基于比较的算法，在最坏情况下至少要做 $n + \lceil \log n \rceil - 2$ 次比较才能选择出第二大元素。

- **证明**
  - 使用对手论证

金币模型

- 将选择最大元素的过程看成是单败淘汰的比赛。
- 假设初始时，每个选手（元素）有1个金币。
- 比赛后，胜者拿走双方全部的金币。

金币个数的比较	对手的应对	金币个数的更新
w(x)>w(y)	X>y	W(x):=w(x)+w(y), w(y):=0
W(x)=w(y)>0	同上	同上
W(y)>w(x)	Y>x	W(y):=w(x)+w(y), w(x):=0
W(x)=w(y)=0	与以前的结果一致	无变化

对手策略分析

- 总的金币数总是 n
- 假设 x 最大，那么最终 x 是唯一有金币的，w(x)=n
- 根据对手策略：
  - $w_k(x) \le 2w_{\{k-1\}}(x)$
- K 是 x 最后所赢得的次数
  - $w_K(x) = n \le 2^K w_0(x) = 2^K$
  - $K \ge \lceil \log n \rceil$

- 下界：$n - 1 + \lceil \log n \rceil - 1 = n + \lceil \log n \rceil - 2$

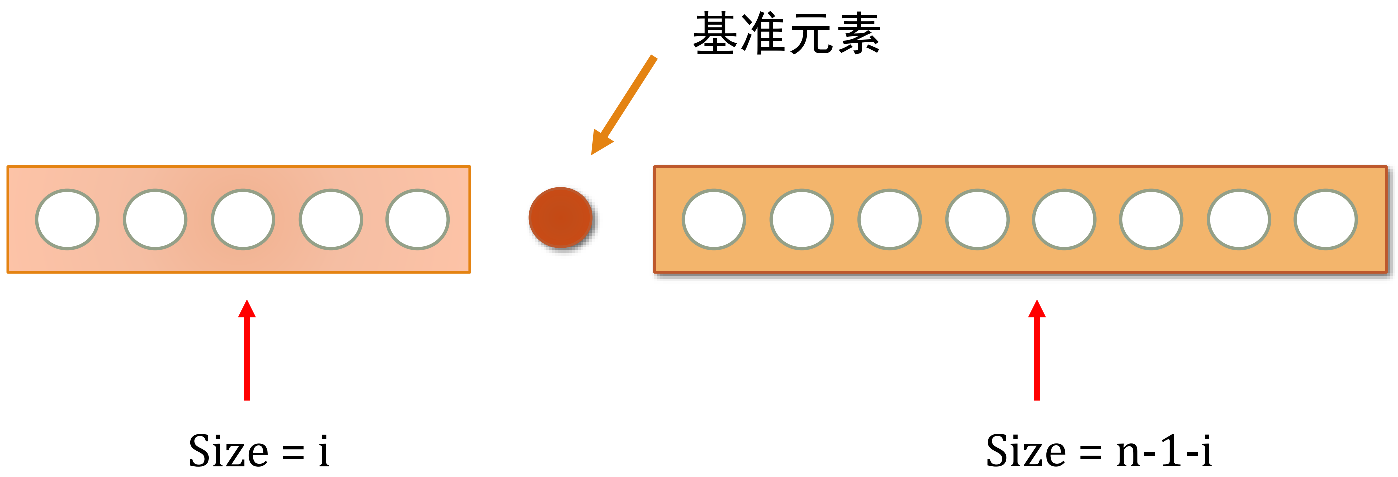
选择中位数

- 如果把元素集合分为两个子集 S1,S2；S1的所有元素小于 S2的所有元素，那么中位数在集合大小更大的子集中；

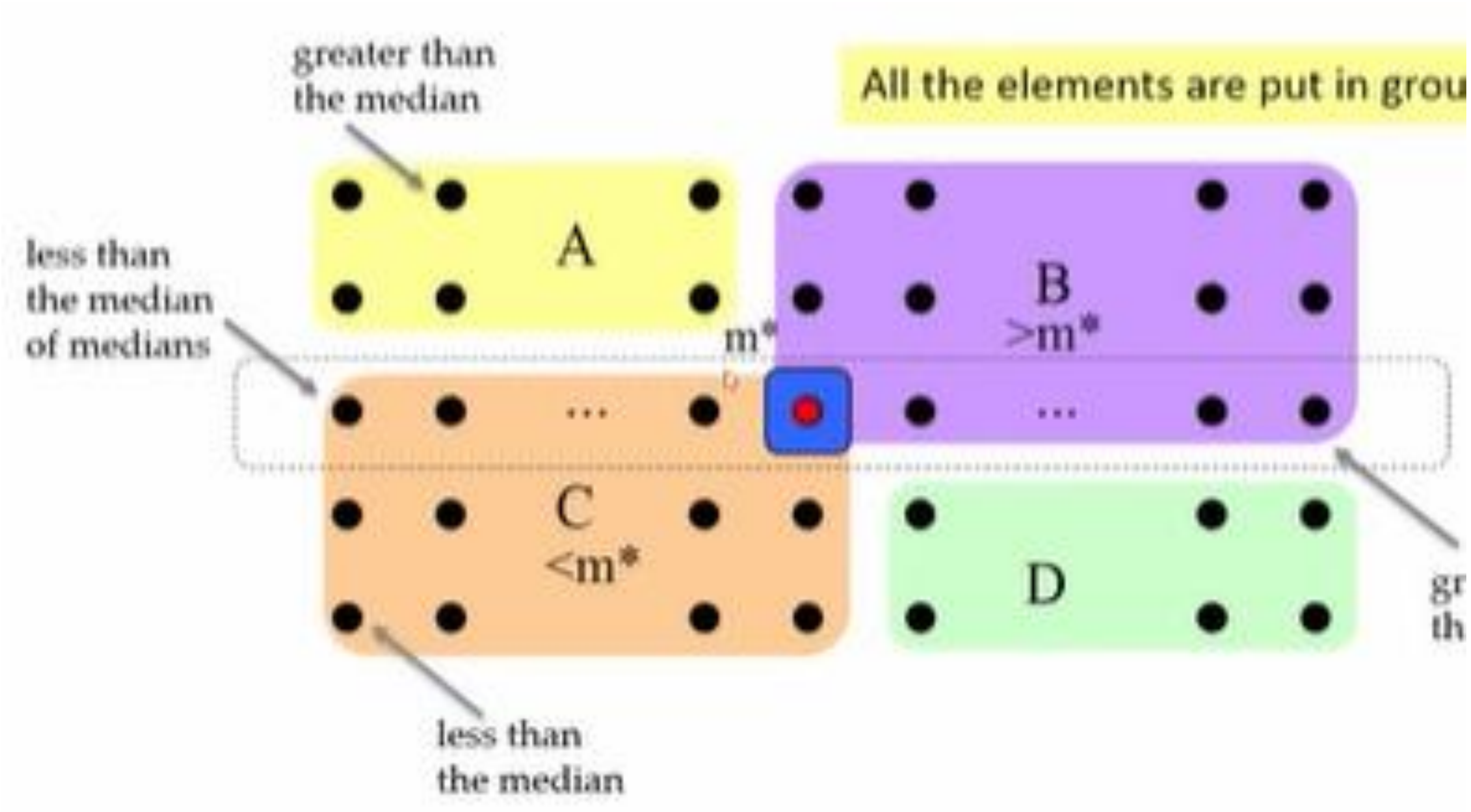
- 分治策略
  - 只需选择一个子集进行递归

期望线性时间选择

- i=k-1, 返回基准元素
- i>k-1, 左边递归，k
- i<k-1, 右边递归，k-n_L-1



最坏情况线性时间选择



greater than
the median
All the elements are put in grou
A
B
>m*
m*
less than
the median
of medians
...
...
C
<m*
D
less than
the median

选择中位数下界—关键比较模型

- 关键比较：如果一次比较帮助算法的设计者确定了中位数与某个元素的关系，则这次比较为一次关键比较。
  - 例：已知 x<medium; if x>y; 则 y<medium；x>y 是一次关键比较

- 非关键比较：如果一次比较无法帮助确定中位数与某个元素的关系，则这次比较为一次非关键比较。
  - 例：已知 x<medium; if x<y; 则 y?medium；x<y 是一次非关键比较

基于关键比较模型的对手策略设计

- N：元素还未参加过比较
- L：元素被赋予一个大于中位数的值
- S：元素被赋予一个小于中位数的值

比较前元素的状态	对手的应对
N, N	使一个元素大于中位数，另一个元素小于中位数
(L, N) 或者 (N, L)	N 变成 S（给状态为N的元素赋一个小于中位数的值）
(S, N) 或者 (N, S)	N 变成 L（给状态为N的元素赋一个大于中位数的值）

- 下界： $\frac{n-1}{2}+n-1=\frac{3}{2}n-\frac{3}{2}$

Thanks