



上节课内容

- 排序问题复杂度下界
- 冒泡排序
- 堆排序
- 决策树确定下界





选择算法的时间 复杂度分析

下界证明方法：构造最坏输入

- 任意给定一个算法 A ， A 对于任意输入 x 都存在一个确定的操作序列 τ
- τ 中的操作分成两类：
 - 决定性的：能够对确定输出结果提供有效信息
 - 非决定性的：对确定结果没有帮助的冗余操作
- 根据算法 A 构造某个输入实例 x ，使得 A 对 x 的操作序列 τ 包含尽量多的非决定性操作。
- 给出冗余操作+必要的操作的计数公式



选择算法的有关结果

	算法	最坏情况	空间
选最大	顺序比较	$n-1$	$O(1)$
选最大和最小	顺序比较	$2n-3$	$O(1)$
	算法 FindMaxMin	$\lceil 3n/2 \rceil - 2$	$O(1)$
选第二大	顺序比较	$2n-3$	$O(1)$
	锦标赛方法	$n + \lceil \log n \rceil - 2$	$O(n)$
选中位数	排序后选择	$O(n \log n)$	$O(1)$
	算法Select	$O(n) \sim 2.95n$	$O(\log n)$

选最大算法 Findmax是最优的算法



北京大學



选最大与最小算法

定理6 任何通过比较找最大和最小的算法至少需要 $\lceil 3n/2 \rceil - 2$ 次比较.

证明思路: 任给算法 A , 根据算法 A 的比较结果构造输入 T , 使得 A 对 T 至少做 $\lceil 3n/2 \rceil - 2$ 次比较.

证: 不妨设 n 个数彼此不等, A 为任意找最大和最小的算法. \max 是最大, A 必须确定有 $n-1$ 个数比 \max 小, 通过与 \max 的比较被淘汰. \min 是最小, A 也必须确定有 $n-1$ 个数比 \min 大, 通过与 \min 的比较而淘汰. 总共需要 $2n-2$ 个信息单位.



基本运算与信息单位

数的状态标记及其含义：

N：没有参加过比较

W：赢

L：输

WL：赢过且至少输1次

如果比较后数的状态改变，则提供信息单位，状态不变不提供信息单位，每增加 1 个W 提供 1个信息单位
每增加 1 个L 提供 1 个信息单位.

两个变量通过一次比较增加的信息单位个数不同：0,1,2

case1 : $N, N \rightarrow W, L$: 增加2个信息单位

case2 : $W, N \rightarrow W, L$: 增加1个信息单位

case3 : $W, L \rightarrow W, L$: 增加0个信息单位





算法输出与信息单位

算法输出的条件:

$n-2$ 个数带有 W 和 L 标记, 最大数只带 W 标记, 最小数只带 L 标记, 总计 $2n-2$ 个信息单位

对于任意给定的算法, 构造输入的原则是:

根据算法的比较次序, 针对每一步参与比较的两个变量的状态, 调整对参与比较的两个变量的赋值, 使得每次比较后得到的信息单位数达到最小. 从而使得为得到输出所需要的 $2n-2$ 个信息单位, 该算法对所构造的输入至少要做 $\lceil 3n/2 \rceil - 2$ 次比较.





对输入变量的赋值原则

x 与 y 的状态	赋值策略	新状态	信息单位
N,N	$x > y$	W,L	2
W,N; WL,N	$x > y$	W,L; 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; WL,L; W,WL	$x > y$	不变	0
WL,WL	保持原值	不变	0



一个赋值的实例

$x_1, x_2 \dashrightarrow x_1 > x_2$; $x_1, x_5 \dashrightarrow x_1 > x_5$; $x_3, x_4 \dashrightarrow x_3 > x_4$; $x_3, x_6 \dashrightarrow x_3 > x_6$
 $x_3, x_1 \dashrightarrow x_3 > x_1$; $x_2, x_4 \dashrightarrow x_2 > x_4$; $x_5, x_6 \dashrightarrow x_5 > x_6$; $x_6, x_4 \dashrightarrow x_6 > x_4$...

	x_1	x_2	x_3	x_4	x_5	x_6
	状态 值	状态 值	状态 值	状态 值	状态 值	状态 值
	N *	N *	N *	N *	N *	N *
$x_1 > x_2$	W 20	L 10				
$x_1 > x_5$	W 20				L 5	
$x_3 > x_4$			W 15	L 8		
$x_3 > x_6$			W 15			L 12
$x_3 > x_1$	WL 20		W 25			
$x_2 > x_4$		WL 10		L 8		
$x_5 > x_6$					WL 5	L 3
$x_6 > x_4$				L 2		WL 3

构造的输入为 (20, 10, 25, 2, 5, 3)



北京大学



问题复杂度的下界

为得到 $2n-2$ 个信息单位, 对上述输入 A 至少做 $\lceil 3n/2 \rceil - 2$ 次比较.

一次比较得到2个信息单位只有case1. A 至多有 $\lfloor n/2 \rfloor$ 个case1, 至多得到 $2\lfloor n/2 \rfloor \leq n$ 个信息单位. 其它case, 1次比较至多获得1个信息单位, 至少还需要 $n-2$ 次比较.

当 n 为偶数, A 做的比较次数至少为

$$\lfloor n/2 \rfloor + n - 2 = 3n/2 - 2 = \lceil 3n/2 \rceil - 2$$

当 n 为奇数, A 做的比较次数至少为

$$\lfloor n/2 \rfloor + n - 2 + 1 = (n-1)/2 + 1 + n - 2 = \lceil 3n/2 \rceil - 2$$

结论: FindMaxMin是最优算法





找第二大问题

元素 x 的权： $w(x)$, 表示以 x 为根的子树中的结点数

初始, $w(x_i)=1, i=1, 2, \dots, n$;

赋值原则： 在比较的时候进行赋值或者调整赋值. 只对没有失败过的元素（权大于0的元素）进行赋值. 权大者胜，原来胜的次数多的仍旧胜，输入值也大.

1. $w(x), w(y) > 0$:

若 $w(x) > w(y)$, 那么 x 的值大于 y 的值; //权大者胜

若 $w(x) = w(y)$, 那么 x 的值大于 y 的值; //权等，任意分配

2. $w(x) = w(y) = 0$, 那么 x, y 值不变; // x 与 y 比较对于确定第二大无意义





实例

	$w(x_1)$	$w(x_2)$	$w(x_3)$	$w(x_4)$	$w(x_5)$	值
初始	1	1	1	1	1	*, *, *, *, *
第1步 $x_1 > x_2$	2	0	1	1	1	20, 10, *, *, *
第2步 $x_1 > x_3$	3	0	0	1	1	20, 10, 15, *, *
第3步 $x_5 > x_4$	3	0	0	0	2	20, 10, 15, 30, 40
第4步 $x_1 > x_5$	5	0	0	0	0	41, 10, 15, 30, 40





构造树

根据算法A 的比较次序, 在比最大的过程中如下构造树:

1. 初始是森林, 含有 n 个结点;
2. 如果 x, y 是子树的树根, 则算法比较 x, y ;
3. 若 x, y 以前没有参加过比较, 任意赋值给 x, y , 比如 $x > y$; 那么将 y 作为 x 的儿子;
4. 若 x, y 已经在前面的比较中赋过值, 且 $w(x) > w(y)$, 那么把 y 作为 x 的儿子, 以 y 为根的子树作为 x 的子树;



初始

x_1

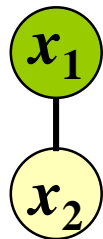
x_2

x_3

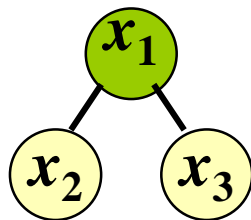
x_4

x_5

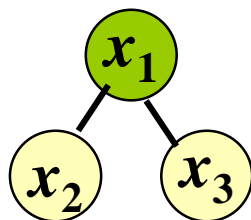
$x_1 > x_2$



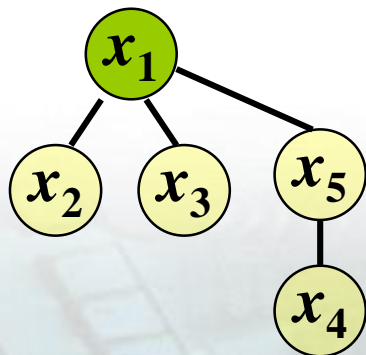
$x_1 > x_3$



$x_5 > x_4$



$x_1 > x_5$



x_3

x_4

x_5

x_4

x_5

x_5

x_4

实例



北京大学

找第二大问题复杂度下界

针对这个输入，估计与max比较而淘汰的元素数
根的权为 n ，其它的结点权为0，根为max

w_k 表示 max 在它第 k 次比较后形成以max为根子树的结点
总数，则 $w_k \leq 2w_{k-1}$ ，设 K 为max最终与权不为0的结点的比
较次数，则

$$n = w_K \leq 2^K w_0 \leq 2^K \Rightarrow K \geq \log n \Rightarrow K \geq \lceil \log n \rceil$$

这 K 个元素彼此不同，因为同一个元素不可能被 计数 2 次。
其中为确定第二大，要淘汰 $K-1$ 个元素，至少用 $\lceil \log n \rceil - 1$ 次
比较。

结论：锦标赛方法是找第二大的最优算法。





找中位数问题

定理8 设 n 为奇数，任何通过比较运算找 n 个数的中位数 (median) 的算法在最坏情况下至少做 $3n/2 - 3/2$ 次比较

证 首先定义决定性的比较与非决定性的比较。

决定性的比较: 建立了 x 与 median 的关系的比较。

$\exists y (x > y \text{ 且 } y \geq \text{median})$, x 满足上述条件的第一次比较

$\exists y (x < y \text{ 且 } y \leq \text{median})$, x 满足上述条件的第一次比较

(比较时 y 与 median 的关系可以不知道)

非决定性的比较: 当 $x > \text{median}$, $y < \text{median}$, 这时 $x > y$ 的比较不是决定性的。

为找到中位数，必须要做 $n-1$ 次决定性的比较。

针对算法构造输入，使得非决定性比较达到 $(n-1)/2$ 次。





输入构造方法

1. 分配一个值给中位数 **median**;
2. 如果A比较 x 与 y , 且 x 与 y 没有被赋值, 那么赋值 x, y 使得 $x > \text{median}, y < \text{median}$;
3. 如果A比较 x 与 y , 且 $x > \text{median}$, y 没被赋值, 则赋值 y 使得 $y < \text{median}$;
4. 如果A比较 x 与 y , 且 $x < \text{median}$, y 没被赋值, 则赋值 y 使得 $y > \text{median}$;
5. 如果存在 $(n-1)/2$ 个元素已得到小于**median**的值, 则对未赋值的全部分配大于**median**的值;
6. 如果存在 $(n-1)/2$ 个元素已得到大于**median**的值, 则对未赋值的全部分配小于**median**的值.
7. 如果剩下1个元素则分配**median**给它.





构造实例

$x_1, x_2 \text{---} x_1 > x_2; \quad x_3, x_4 \text{---} x_3 > x_4; \quad x_5, x_6 \text{---} x_5 > x_6;$
 $x_1, x_3 \text{---} x_1 > x_3; \quad x_3, x_7 \text{---} x_3 > x_7; \quad x_7, x_4 \text{---} x_7 > x_4; \quad \dots$

1. 初始 $\text{median}=4$

2. $x_1 > x_2$ $x_1=7, x_2=1$

3. $x_3 > x_4$ $x_3=5, x_4=2$

4. $x_5 > x_6$ $x_5=6, x_6=3$

5. $x_7=4$

6. $x_1 > x_3$

7. $x_3 > x_7$

8. ...

非决定性比较

决定性比较



北京大学

复杂性分析

元素状态 N: 未分配值; S: 得到小于median值;
L: 得到大于median值

比较前的状态	分配策略
N, N	一个大于median, 一个小于median
L, N 或 N, L	分配给状态N的元素的值小于median
S, N 或 N, S	分配给状态N的元素的值大于median

这样赋值的输入使得A在这个输入下所进行的上述比较都是非决定性的. 这样的比较至少有 $(n-1)/2$ 个. 因此总比较次数至少为

$$(n-1) + (n-1)/2 = 3n/2 - 3/2$$

结论: Select算法在阶上达到最优.





几种选择算法的总结

问题	算法	最坏情况	问题下界	最优性
找最大	Findmax	$n-1$	$n-1$	最优
找最大最小	FindMaxMin	$\lceil 3n/2 \rceil - 2$	$\lceil 3n/2 \rceil - 2$	最优
找第二大	锦标赛	$n + \lceil \log n \rceil - 2$	$n + \lceil \log n \rceil - 2$	最优
找中位数	Select	$O(n)$	$3n/2 - 3/2$	阶最优
找第 k 小	Select	$O(n)$	$n + \min\{k, n-k+1\} - 2$	阶最优





通过归约确认问题 计算复杂度的下界

问题P, 问题Q

问题Q的复杂度已知（至少线性） $\Omega(g(n))$

存在变换 f 将Q的任何实例转换成 P 的实例, f 的时间为线性时间 $f(n)=O(n)$, 解的反变换 $s(n)$ 也是线性时间

解Q的算法: $T_Q(n)=f(n)+T_P(n)+s(n)$

1. 将Q的实例 I 变成 $f(I)$, $f(n)$
2. 用解 P 的算法作为子程序解 $f(I)$, 时间与解P的时间为同样的阶 $T_P(n)$
3. 将解变换成原问题的解 $s(n)$

解P的算法可以解Q. 且时间的阶一样, 因此 P至少与Q一样难. $Q \leq_l P$ (l 表示线性时间)

$$f(n)+T_P(n)+s(n)=T_Q(n)=\Omega(g(n))$$



因子分解与素数测试

- 问题:

因子分解 factor: 输入正整数 n , $\text{factor}(n)$ 是多重集(全部素因子)规定 $\text{factor}(1)=\{1\}$

素数测试 testp: 输入正整数 n , $\text{test}(n)$ 为 “Yes” 或者 “No”

- 归约: $\text{testp} \leq \text{factor}$

假设 testp 问题的难度是 $W(n)$.

素数测试算法 $A(n)$

1. if $n=1$ then return “No”
2. else $p \leftarrow \text{factor}(n)$
3. if $|p| \geq 2$ then return “No”
4. else return “Yes”

- 结论: $\Omega(W(n)) = T_{\text{testp}}(n) \leq T_{\text{factor}}(n)$





元素唯一性问题

- 问题：给定 n 个数的集合 S ，判断 S 中的元素是否存在相同元素。

- 元素唯一性问题的复杂度为 $\Theta(n \log n)$

输入：多重集 $S = \{ n_1 \cdot a_1, n_2 \cdot a_2, \dots, n_k \cdot a_k \}$

构造决策树，树叶为 S 的全排列数

$$\frac{n!}{n_1! n_2! \dots n_k!}$$

最坏情况下树深为

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



最邻近点对与唯一性问题

- P问题与Q问题:
P: 平面直角坐标系中 n 个点的最邻近点对问题Close
Q: 元素的唯一性问题Uniqueness $\Omega(n \log n)$
- 变换 f :
Q的实例: x_1, x_2, \dots, x_n , 变成点 $(x_1, 0), (x_2, 0), \dots, (x_n, 0)$
- 解Q算法:
 1. 利用求最邻近点对算法 P 计算最短距离 d .
 2. if $d=0$ then return “No”
 3. else return “Yes”
- 结论: 计算平面直角坐标系中 n 个点的最邻近点对问题的时间是 $\Omega(n \log n)$, 其中算法以比较为基本运算



最小生成树与唯一性问题

- P问题与Q问题:

P: 平面直角坐标系中 n 个点的最小生成树问题;

Q: 元素的唯一性问题Uniqueness $\Omega(n\log n)$

- 变换 f :

Q的实例: x_1, x_2, \dots, x_n , 变成X轴上的 n 个点,

- 解Q算法:

1. 利用求最小生成树算法P构造树 T , 确定 T 的最短边 e .
2. 检测 e 的长度是否为0
3. if $|e|=0$ then 不唯一, else 是唯一的.

- 结论: 计算平面直角坐标系 n 点最小生成树时间是 $\Omega(n\log n)$, 其中算法以比较为基本运算

