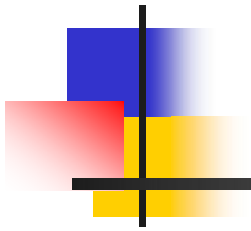
Ch5 Selection and Adversary Arguments



选择和对策论

5.1 Introduction

- 研究一类“selection”问题：极值选择，次元选择，中间元选择等
- 研究有效的算法
- 研究问题的下界----对策论
- 利用对策论计算selection问题的时间复杂度的下界

排序？

排序通常O(nlogn),而有些选择问题可在线性时间内解决

数组E包含n个数据元素，这n个数据元素的关键字的值来自某个线性有序集。所谓“选择”问题就是求E中关键字的值第$k(1\le k\le n)$小的数据元素。

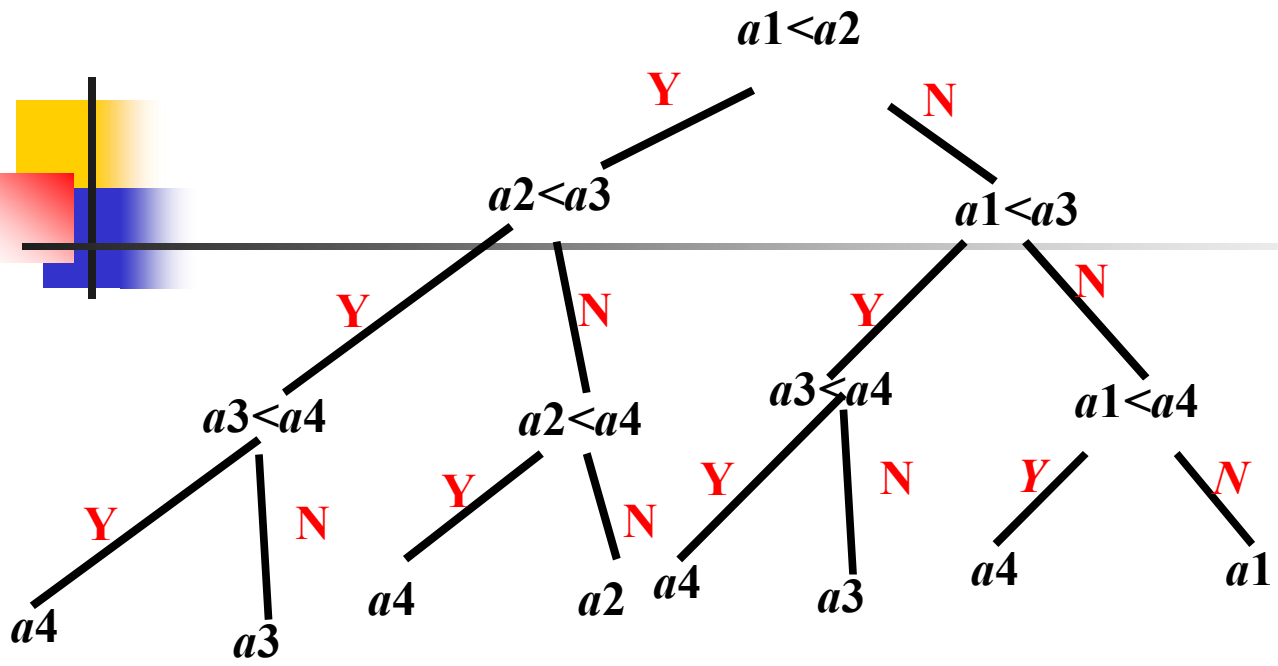
5.1 Introduction

- 主要操作：比较（本章不分析“移动”操作的消耗）
- 约定：数组中不存在相同的数组元素

- 问什么不采用决策（判定）树确定selection问题的时间复杂度的下界?

5.1 Introduction

- n个元素“比较”排序→决策树→下界$\log n!$
- 选择问题，至少有n个叶子，判定树的高度至少为$\lceil \log n \rceil + 1$
- **n=4时找最大值的判定树**



n=4时找最大值的判定树

对n个数据元素找最大值，其判定树的叶子多于n个

无法通过logn确定判定树的高度

不知道有多少叶子结点的值是相同的
所以放弃决策树，采用对策论确定selection
问题的时间复杂度的下界

什么是对策论

- p表示要解决的问题
- I_j表示问题p的一种可能的输入
- A_i表示问题p的基于比较运算的一种解法
- $T(A_i, I_j)$针对输入I_j解法A_i的计算时间复杂性
- $U(n)=\min_i \{\max_j (T(A_i, I_j))\}$

- -----解决该问题的最好方法

什么是对策论

- 精确求解U(*n*)的困难性：

1. 分析问题*p*的所有可能的输入
2. 分析问题*p*的所有可能的求解算法

- 令：$f(n) \leq U(n)$，$f(n)$尽量接近$U(n)$，为问题*p*的一个好的下界----采用对策论求解$f(n)$
- 对策论：构造一套对于一切解决问题*p*的算法都适用的输入策略，从而问题求解做尽量多的比较和判断，即在相应输入策略下，时间复杂度尽量大。
- 构造策略因问题而异，好的下界需要有好的策略

5.2 Finding max and min

- 方法1： *max*=E[1]; *min*=E[1];

  for (*index*=2;*index*<=*n*;*index*++)

  if (E[*index*]>*max*) *max*=E[*index*];

  if (E[*index*]<*min*) *min*=E[*index*];

  总计2*n*-2次比较

- 方法2： “Find *max*” *n*-1次比较（冒泡），“Find *Min*” *n*-2次比较（冒泡），总计2n-3次比较
  - 分析： Finding max and min相当于各自单干

5.2 Finding max and min

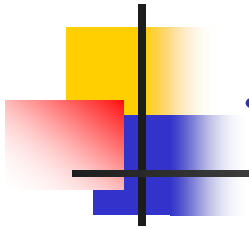
- 方法3：进一步优化，降低比较次数。

  (1) 当n=2k时，E[i]与E[i+1]比较，i=1,3, 5，…,2k-1, **胜者**放入A数组，**败者**放入B数组; A数组比较k-1次选取最大值Max; B数组比较k-1次选取最小值Min。

  总的比较次数为3k-2=3n/2-2

  (2) 当n=2k+1时，E[i]与E[i+1]比较，i=1,3, 5，…,2k-1, **胜者**放入A数组，**败者**放入B数组;将E[2k+1]加入A，A数组比较k次选取最大值Max;将E[2k+1]加入B， B数组比较k次选取最小值Min。

  总的比较次数为3k=3(n-1)/2



5.2 Finding max and min

- Theorem 5.1 Any algorithm to find max and min of n keys by comparison of keys must do at least 3n/2-2 key comparisons in the worst case.
- 分析：若n个数据元素中x为max,y为min，那么除了x外，其他的数据元素在比较中都失败过；除了y外，其他的数据元素在比较中都胜过。

5.2 Finding max and min

- 设“赢”或“失败”为一个信息
- **至少需要2n-2个信息才能得出结论**

- 若n个数据元素中x为max,y为min，那么除了x外，其他的数据元素在比较中都失败过（n-1个信息）；除了y外，其他的数据元素在比较中都胜过（n-1个信息）。

- 每一个数据元素的状态有以下4种：

1. W：从未失败，至少赢过一次
2. L：从未赢过，至少失败一次
3. WL：既成功过，也失败过，至少各一次
4. N：尚未参加过比较

- 初始时，所有元素均未参加比较，状态都是“N”

Finding max and min对策策略

x,y比较前状态	对策论措施	比较后状态	获得的新信息数
N, N	x>y	W, L	2
W, N or WL, N	x>y	W, L or 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 or WL,L or W, WL	x>y	no change	0
WL， WL	一致	no change	0

n 为偶数的情况

- 初始所有的数据元素状态均为N。
- 一次比较获得2个信息只有(N,N)这种情况。而这种情况最多进行n/2次，获得n个信息。即：经过n/2次比较，最多获得n个信息。
- 其他每次比较最多产生1个信息，还需 2n-2-n=n-2个信息，至少需要n-2次比较
- 所以至少要进行n/2+ n-2次比较。

- n 为偶数：Any algorithm to find max and min of n keys by comparison of keys must do at least 3n/2-2 key comparisons in the worst case.

n 为奇数的情况

- 初始所有的数据元素状态均为N。
- 一次比较获得2个信息只有(N,N)这种情况。而这种情况最多进行（n-1）/2次，获得n-1个信息。
- 其他每次比较最多产生1个信息，还需 2n-2-n+1=n-1个信息，至少需要n-1次比较
- 所以至少要进行(n-1)/2+ n-1次比较。

- n 为奇数：Any algorithm to find max and min of n keys by comparison of keys must do at least 3n/2-3/2 key comparisons in the worst case.