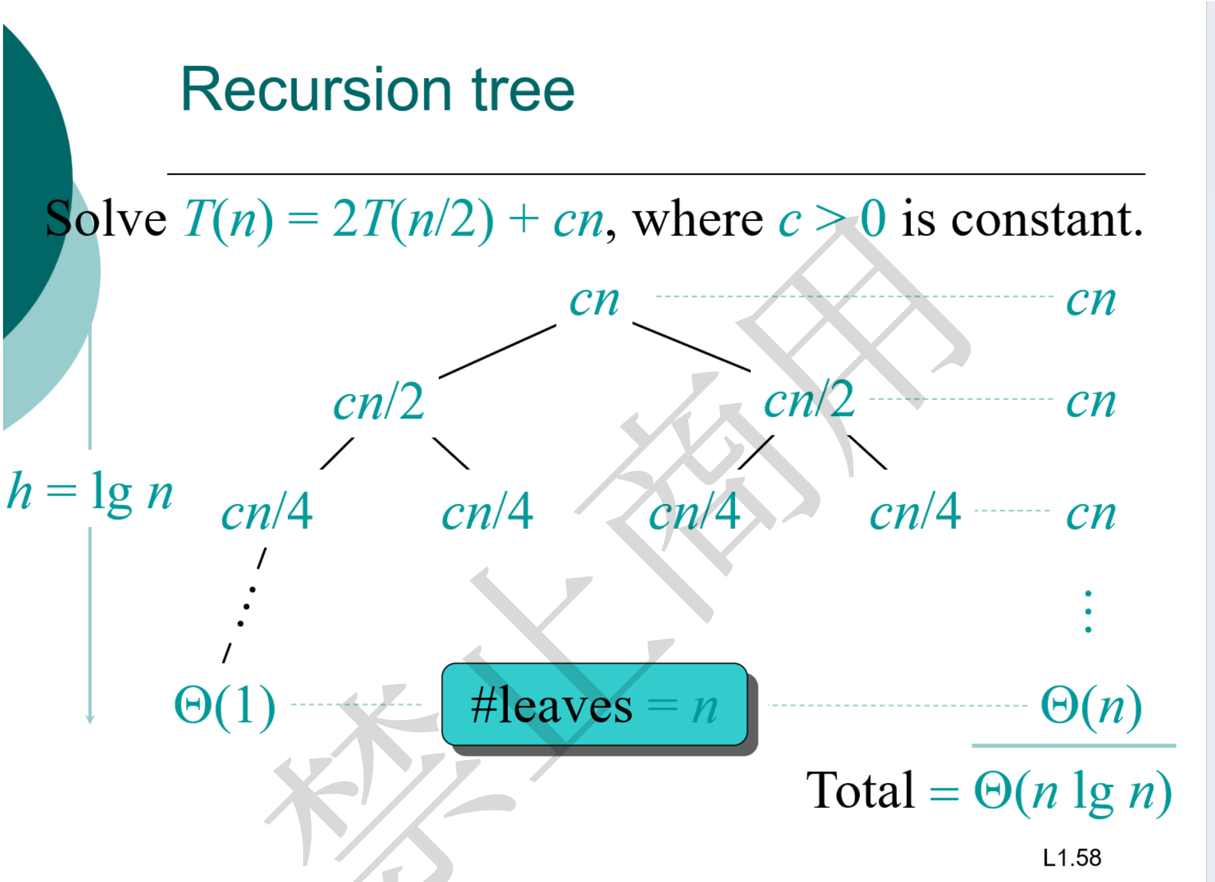
分治策略

$$T(n)\begin{cases}O(1)\\aT(n/b)+D(n)+C(n)\end{cases}$$

一般来说$a=b=2$但是也可以是其他值

分析树

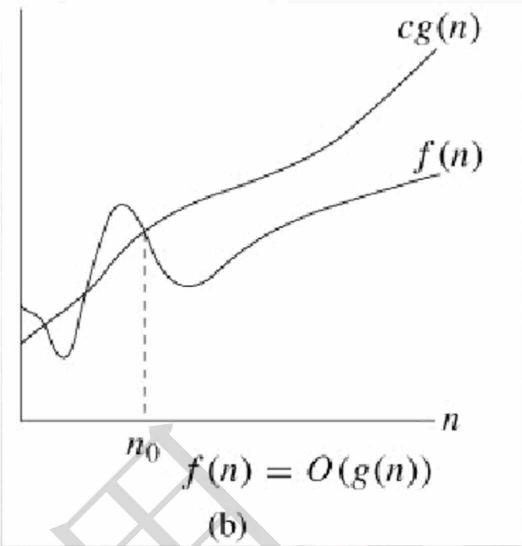


复杂度符号

3.1 渐近记号

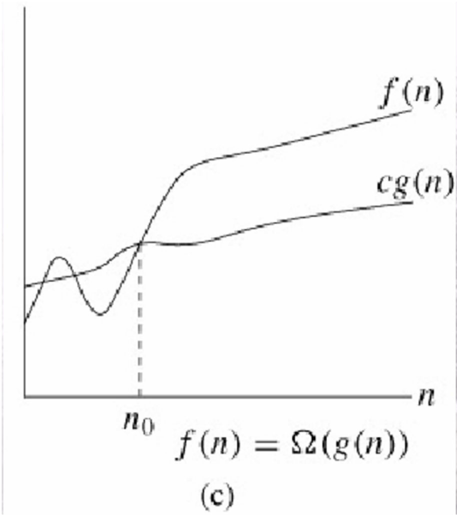
- O记号
  - $O(g(n)) = \{f(n)$:存在正常数c和n_0使得对所有$n \geq n_0$，有$0 \leq f(n) \leq cg(n)\}$.
  - 渐近上界
- 由定义可知$\Theta(g(n)) \subseteq O(g(n))$
- [例]:证明$an + b = O(n^2)$

  取$c = a + |b|$，$n_0 = 1$即可



3.1 渐近记号

- Ω记号
  - $\Omega(g(n)) = \{f(n)$: 存在正常数c和n_0，使得对所有$n \geq n_0$，有$0 \leq cg(n) \leq f(n)\}$
  - 渐进下界
- 由定义可知$\Theta(g(n)) \subseteq \Omega(g(n))$
- [例]:证明$6n^3 = \Omega(n^2)$
  - 取$c = 1, n_0 = 1$



函数之间的关系

- 自反性：

$f(n)=\Theta(f(n))$

$f(n)=O(f(n))$

$f(n)=\Omega(f(n))$

- 对称性：

$f(n)=\Theta(g(n))$当且仅当$g(n)=\Theta(f(n))$.

- 转置对称性：

$f(n)=O(g(n))$当且仅当$g(n)=\Omega(f(n))$

将f与g的渐近比较和两个实数a和b的比较做类比：

$$f(n)=O(g(n)) \quad \approx a\le b$$
$$f(n)=\Omega(g(n)) \quad \approx a\ge b$$
$$f(n)=\Theta(g(n)) \quad \approx a=b$$

递归式

代换法

- 猜测其形式
- 用数学归纳法证明

猜测经验

- 减掉一个低阶项
- 避免陷阱

◆[经验]避免陷阱

[例] $T(n)=2T(\lfloor n/2 \rfloor)+n$

猜解$T(n) = O(n)$，取$T(n) \le cn$

$T(n) \le 2(c\lfloor n/2 \rfloor) + n \le cn + n = O(n)$，

错误！

证明不符

○

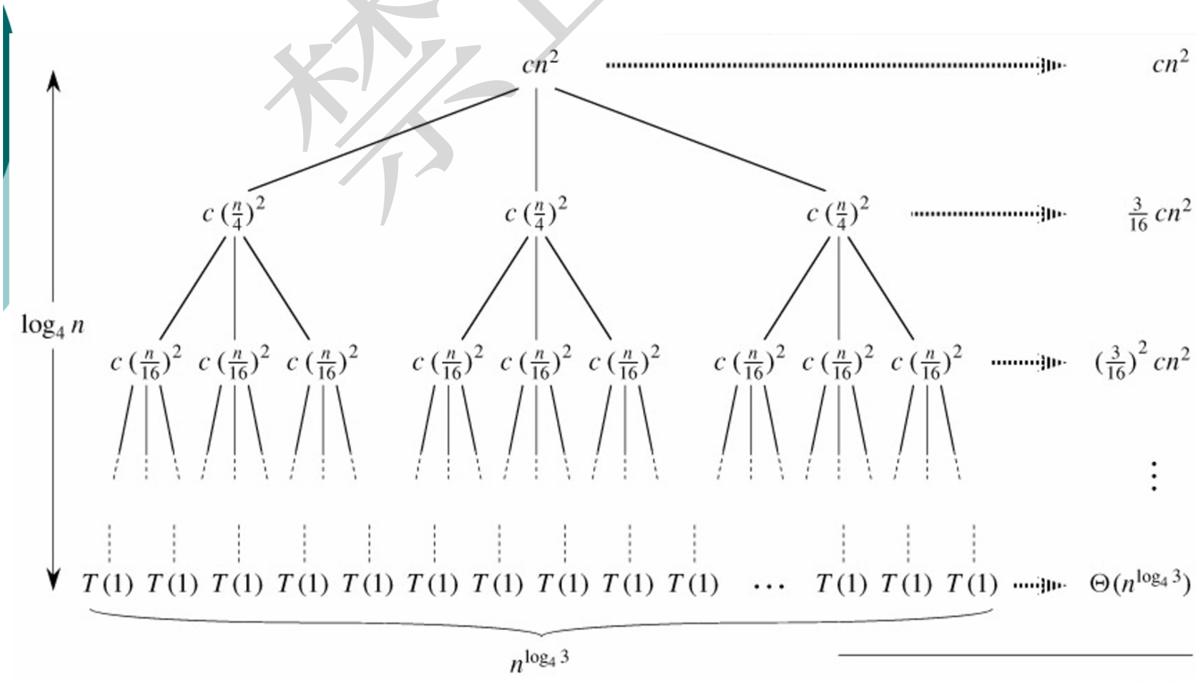
递归树方法

◆通常可以容忍小量的不良量

[例]$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$

忽略顶和底函数，建立递归式$T(n) = 3T(n/4) + cn^2$的递归树，常系数$c > 0$

假设n是4的幂



$$T(n) = cn^2 + \frac{3}{16}cn^2 + (\frac{3}{16})^2 cn^2 + \cdots + (\frac{3}{16})^{\log_4 n-1} cn^2 + \Theta(n^{\log_4 3})$$

主方法

- 给出求解如下形式的递归式的“食谱”方法

$$T(n) = aT(n/b) + f(n)$$

其中$a \geq 1$和 $b > 1$ 是常数

- Master theorem

设$a \geq 1$ 和 $b > 1$ 是常数，设 $f(n)$ 为一函数，$T(n)$ 由递归式 $T(n) = aT(n/b) + f(n)$

对非负整数，其中 n/b 指$\lfloor n/b \rfloor$ 或 $\lceil n/b \rceil$。那么 $T(n)$ 可能有如下的渐近界

1. 若对于某常数$\varepsilon > 0$，有 $f(n) = O(n^{\log_b a-\varepsilon})$，则 $T(n) = \Theta(n^{\log_b a})$
2. 若 $f(n) = \Theta(n^{\log_b a})$，则 $T(n) = \Theta(n^{\log_b a} \lg n)$
3. 若对某常数$\varepsilon>0$，有 $f(n) = \Omega(n^{\log_b a+\varepsilon})$，且对常数$c < 1$与所有足够大的$n$，有$af(n/b) \leq cf(n)$，则$T(n) = \Theta(f(n))$

三种情况需要记住，考试有可能涉及

堆排序

- 最大堆：最大元素放在堆顶
- 最小堆：最小元素放在堆顶

MAX-HEAPIFY时间

对于堆排序中的子树大小最多是2n/3

那么我们可以得到递归表达式

$$T(n) \leq T(2n/3) + \Theta(1)$$

根据主定理可以得到 $T(n) \leq \Theta(\lg n)$

如果要从乱序的堆变为一个最大堆，则需要对每个节点调用一次MAX-HEAPIFY，第一次维护堆的复杂度为$f(n) \leq \Theta(n \lg n)$
从第二次开始维护堆的复杂度为$f'(n) \leq \Theta(\lg n)$
总维护次数为n，所以堆排序的总复杂度为$g(n) \leq \Theta(2n \lg n)$

大顶堆的应用

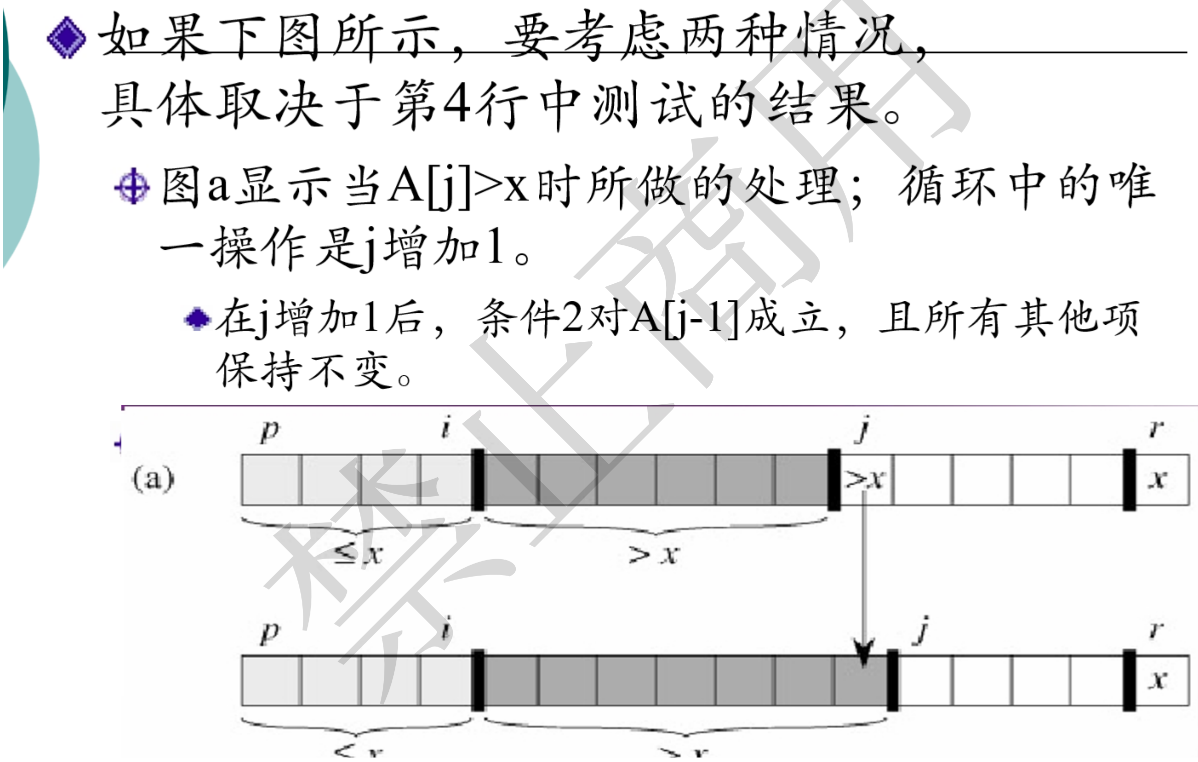
优先队列：根据关键字key进行排序，能够进行去掉并返回S中的具有最大关键字的元素

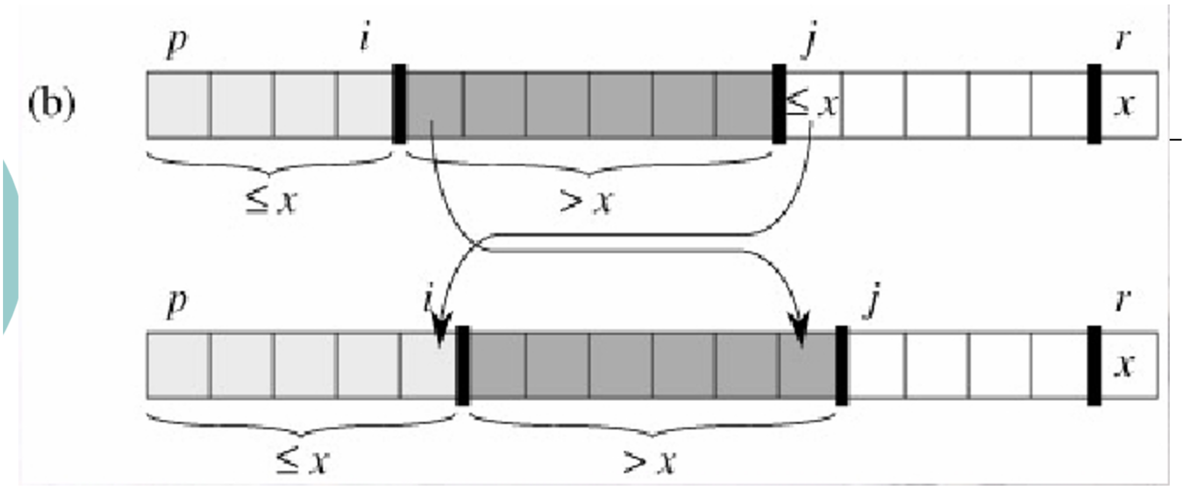
快速排序

特点

对一个包含n个数的数组，最坏情况运行时间是$\Theta(n^2)$，但是平均性能为$\Theta(n\lg n)$

partition





- 图b显示当A[j]<=x时所做的处理：将i增加1，交互A[i]和A[j]，再将j增加1。
  - 因为进行了交换，现在有A[i] <=x，因而条件1满足。
  - 类似地，还有A[j-1]>x，因为根据循环不变式，被交换进A[j-1]的项目是大于x的

上面情况中x为基准，i,j为游标，将数组分为大于基准和小于基准的部分，实现分治。