算法设计与分析

第2章 递归与分治策略
(1) 之
递归表达式求解方法

学习要点

- 掌握设计有效算法的分治策略
- 理解递归的概念
- 递归表达式的求解方法

- 重点和难点：
  - 递归和分治的概念与基本思想
  - 递归方程的求解方法

求解递归表达式的方法

- 如何获取递归表达式的 “Θ” or “O” 界?
  - 代换法(Substitution method) :
    - 猜一个边界，然后使用数学推导来证明这个猜测是正确的
  - 迭代法(Iteration method):
    - 把递归转化为一个求和式子
    - 递归树
  - 主方法(Master method) :
    - 给出了形如该递归式的边界求解方法： $T(n) = aT(n/b)+f(n)$, 其中 $a \geq 1$, $b > 1$, , $f(n)$ 是一个渐近正的函数.

求解递归表达式的方法

- 忽略细节不会影响算法的渐近分析
- 对递归表达式的预处理方法：忽略技术细节
  - 1) 假设函数的参数都是整型
  - 2) 忽略边界条件
  - 3) 忽略上取整和下取整

求解递归表达式的方法

- 预处理方法
  - 1)假设函数的自变量是整数 (integer arguments)
    - 算法运行时间 T(n)在定义时都假设n为 整数
  - 例如， MERGE-SORT最坏运行时间为：
  - $T(n) = \begin{cases} \Theta(1) & if\ n = 1 \\ 2T\left(\left\lceil \frac{n}{2} \right\rceil\right) + 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + \Theta(n) & if\ n > 1 \end{cases}$

求解递归表达式的方法

- 预处理方法
  - 2) 忽略边界条件(boundary conditions)

  - 假设当n较小时，$T(n) = \Theta(1)$（为常数）
  - 例如：
    - 递归表达式为 $T(n) = 2T(n/2) + \Theta(n)$, (省略了 $T(1)=\Theta(1)$)
  - 改变边界T(1)值，可能改变递归式的解，但不改变解的函数增长率

代换法(Substitution)

- 主要步骤
  - (1)猜测解的形式
  - (2)用数学归纳法找出使解真正有效的常数
- 例1 $T(n) = \begin{cases} 1 & if\ n = 1 \\ 2T\left(\frac{n}{2}\right) + n & if\ n > 1 \end{cases}$

解：
(1)猜解的形式为：T(n)=nlgn+n
(2) 利用数学归纳法进行证明。
当n=1时，有nlogn+n=1=T(n).
假设k<n时有：T(k)=klgk+k （根据归纳假设），则：
T(n)=2T(n/2)+n= 2((n/2)lg(n/2)+n/2) +n=nlg(n/2)+2n = nlgn+n ， 得证。

代换法(Substitution)

- 代换法可用来确定递归式的上界或下界（O或Ω).
- 例2：请确定下列递归式的一个上界

$$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$$

  - (1)猜其解为T(n) = O(n lg n).
  - (2)证明 T(n)≤cn lg n ，其中c > 0是某个常数.

  假设这个界对n/2成立，即$T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) \le c\left\lfloor \frac{n}{2} \right\rfloor \lg\left(\left\lfloor \frac{n}{2} \right\rfloor\right)$.

  对递归式作**替换**，得

$$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \le 2\left(c\left\lfloor \frac{n}{2} \right\rfloor \lg\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\right) + n \le cnlg\left(\frac{n}{2}\right) + n$$

$$= cnlgn - cnlg2 + n = cnlgn - cn + n \le cnlgn$$

    - 最后一步只要 c ≥ 1就成立.

代换法

- 方法的局限性
  - 只能用于解的形式很容易**猜**的情形.
  - 应用数学归纳法来求解对**边界条件**成立时有时会导致问题
  - 猜对了，证明不了

  - 猜想不是一种方法，需要经验、运气和创新能力
    - 学习本课程的一个原因：训练我们去获得经验和创新能力

- 幸运的是，有很多启发式方法帮助我们进行猜测

代换法

- 提出好猜测的方法1：
  - 如果某个递归式与先前见过的类似, 则可猜测该递归式有类似的解. 例如
  - $T(n) = 2T\left(\left\lfloor\frac{n}{2}\right\rfloor + 17\right) + n$
  - 分析：看起来比较难解，因为右式加了 “17” .
    - 该附加项不会从本质上影响递归解
    - 当n很大时, T(n/2) 和T(n/2 +17) 之间的区别不大.
    - 因此猜 T(n) = O(n lg n), 该结论可用代换法来验证

代换法

- 提出好猜测的方法2
  - 寻找松的上、下渐近界，范围缩小，逐步逼近
- 例如 $T(n) = 2T\left(\left\lfloor\frac{n}{2}\right\rfloor\right) + n$
  - 首先，从一个低的下界开始 T(n) =Ω(n), 因为递归式中有n
  - 然后，证明一个初始上界 T(n) = O(n^2).
  - 降低上界，提高下界，直到得到一个正确的近渐进界 T(n)=Θ(nlgn).

迭代法(iteration)

- 迭代法
  - 不需要猜测
  - 对代数能力的要求较高
- 基本思想：(不断迭代展开为级数并求和)
  - 迭代地展开递归方程右端
  - 总结规律，表示为一个和式
  - 利用边界条件对和式估计得到方程解.

回顾数学基础：几何级数

- 对于实数x≠1，和式 $\sum_{k=0}^{n} x^k = 1 + x + x^2 + \cdots + x^k$ 是一个几何级数

$$\sum_{k=0}^{n} x^k = \frac{x^{n+1} - 1}{x - 1}$$

- 当和是无穷的，且$|x| < 1$时，得到无穷递减几何级数

$$\sum_{k=0}^{\infty} x^k = \frac{1}{1 - x}$$

迭代法

- 例1：汉诺塔算法 $T(n)=2T(n-1)+1 \quad T(1)=1$

$$
\begin{aligned}
T(n) &= 2T(n-1)+1 \\
&= 2(2T(n-2)+1)+1 \\
&= 2^2T(n-2)+3 \\
&= 4(2T(n-3)+1)+3 \\
&= 2^3T(n-3)+7 \\
&= \ldots\ldots \\
&= 2^iT(n-i)+(2^i-1)
\end{aligned}
$$

展开

总结规律

写通式

当n-i=1 **即** i=n-1**时终止迭代，有:**

$T(n) = 2^{n-1}T(1)+(2^{n-1}-1) = 2^n-1$

利用边界条件求解

迭代法

- 例2：合并排序算法 $T(n)=2T(n/2)+n \quad T(1)=1$

$T(n)=2T(n/2)+n$

$=2(2T(n/4)+n/2)+n$

$=4T(n/4)+2n$

$=4(2T(n/8)+n/4)+2n$

$=8T(n/8)+3n$

$=\ldots\ldots$

$=2^iT(n/2^i)+in$

当$n/2^i=1$ **即** $i=log_2n$**时终止迭代，有:**

$T(n)=2^{log_2n}T(1)+nlog_2n$

$=n+nlogn$

展开

总结规律

写通式

利用边界条件求解

迭代法

- 例3: $T(n) = n + 3T\left(\left\lfloor \frac{n}{4} \right\rfloor\right)$ T(1)=1

n一定能被整除吗？

- 分析：
  - 当递归式包含floor和ceiling 函数时, 表达起来会特别复杂
  - 所以通常假设
    - 能被整除
    - n = 4^k for some integer k, floor函数也省略.

迭代法

- 例3: $T(n) = n + 3T\left(\left\lfloor \frac{n}{4} \right\rfloor\right)$ T(1)=1

$$T(n) = n + 3T\left(\left\lfloor \frac{n}{4} \right\rfloor\right)$$

$$= n + 3\left(\left\lfloor \frac{n}{4} \right\rfloor + 3T\left(\left\lfloor \frac{n}{16} \right\rfloor\right)\right)$$

$$= n + 3\left\lfloor \frac{n}{4} \right\rfloor + 9T\left(\left\lfloor \frac{n}{16} \right\rfloor\right)$$

$$= n + 3\left\lfloor \frac{n}{4} \right\rfloor + 9\left\lfloor \frac{n}{16} \right\rfloor + 27T\left(\left\lfloor \frac{n}{64} \right\rfloor\right)$$

$$= \cdots$$

$$= n + 3\left\lfloor \frac{n}{4} \right\rfloor + 9\left\lfloor \frac{n}{16} \right\rfloor + 3^{i-1}\left\lfloor \frac{n}{4^{i-1}} \right\rfloor + 3^{i}T\left(\left\lfloor \frac{n}{4^{i}} \right\rfloor\right)$$

展开

通项

迭代法

- 例3: $\mathrm{T(n)} = n + 3T\left(\left\lfloor\frac{n}{4}\right\rfloor\right)$ T(1)=1

因为$\left\lfloor\frac{n}{4^{\mathrm{i}}}\right\rfloor \le n/4^i$，有：

$$\mathrm{T}(n) \le n + 3\frac{n}{4} + 9\frac{n}{16} + 3^{i-1}\frac{n}{4^{i-1}} + 3^i T\left(\frac{n}{4^i}\right)$$

当$n/4^i$=1 即i=$\log_4$n时终止迭代，有：

何时结束

$$\mathrm{T}(n) \le n \sum_{\mathrm{i}=0}^{\log_4 n-1} \left(\frac{3}{4}\right)^i + 3^{\log_4 n} T(1)$$

$$\le n \sum_{\mathrm{i}=0}^{\infty} \left(\frac{3}{4}\right)^i + 3^{\log_4 n} T(1)$$

无穷递减几何级数

$$= n \sum_{\mathrm{i}=0}^{\infty} \left(\frac{3}{4}\right)^i + n\log_4{}^3 \quad = 4n + \theta(n^{\log_4 3}) \quad = O(n)$$

迭代法

- 两个关键
  - 递归何时终止
  - 级数求和
- 在展开递归式为迭代求和的过程中，有时只需要部分展开，然后根据其规律来猜想递归式的解，接着用代换法进行证明。

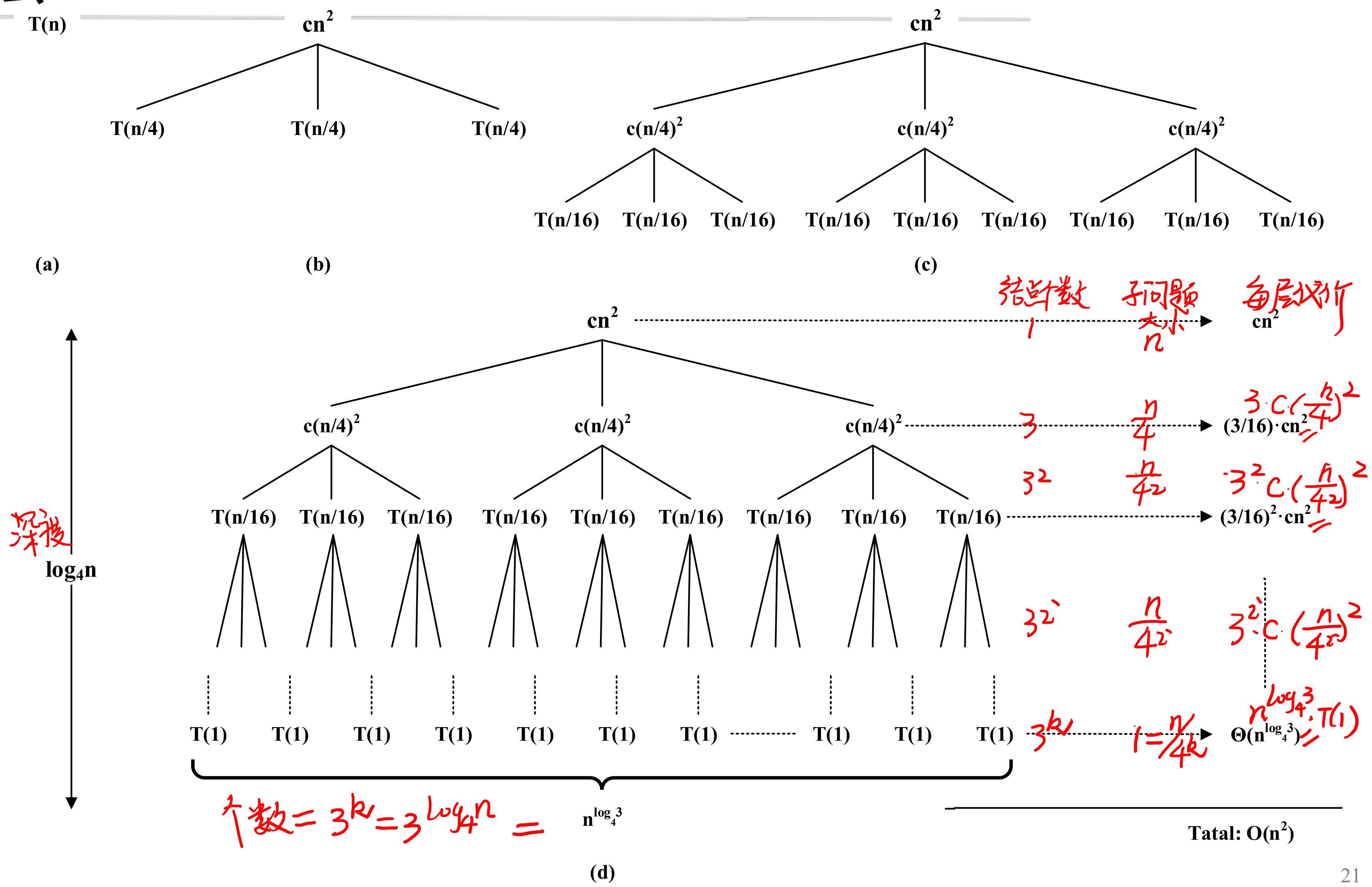
递归树方法

- 画**递归树**可以从直观上表示迭代法，也有助于猜想递归式的解

- 递归树特别适用于分治算法的时间复杂度的递归表达式
  - 每一个结点代表递归函数调用集合中一个子问题的代价。
  - 例：$T(n)=3T(n/4)+cn^2$的递归树

递归树方法

$T(n)=3T(n/4)+cn^2$



递归树方法

- 画递归树的关键
  - 1）根结点
  - 2）每层节点的个数，个数与深度有什么关系
  - 3）确定何时终止，或最后一层是什么？

- $\mathrm{T}(n) = cn^2 + 3c(\frac{n}{4})^2 + 3^2c(\frac{n}{4^2})^2 + \cdots 3^i c\left(\frac{n}{4^i}\right)^2 + 3^{k-1}\, c\left(\frac{n}{4^{k-1}}\right)^2 + 3^k\, T(1)$
- $= cn^2 + \frac{3}{16}cn^2 + (\frac{3}{16})^2 cn^2 + \cdots \left(\frac{3}{16}\right)^i cn^2 + \left(\frac{3}{16}\right)^{k-1} cn^2 + n^{\log_4 3} T(1)$
- $\leq \sum_{i=0}^{\infty} (\frac{3}{16})^i\, cn^2 + n^{\log_4 3}$
- $= \frac{16}{13} cn^2 + n^{\log_4 3} = \mathrm{O}(\mathrm{n}^2)$

递归树方法

- 用代换法证明：T(n)=O(n^2)是递归式$T(n) = 3T(\lfloor n/4 \rfloor) + cn^2$的一个上界。
- 证：要证上界即要证存在常数d>0，使得T(n)$\leq \mathrm{dn}^2$ 成立。

$$
\begin{aligned}
T(n) &= 3T(\lfloor n/4 \rfloor) + cn^2 \\
&\leq 3\, d(\lfloor n/4 \rfloor)^2 + cn^2 \leq 3\, d(n/4)^2 + cn^2 \\
&= \frac{3}{16} dn^2 + cn^2 \leq dn^2 \ (\text{当}\mathrm{d} \geq \frac{16}{13}\mathrm{c}\text{时})
\end{aligned}
$$

主方法

- 主定理
  - 设a≥1 和b>1为常数, 设f(n)为一正函数, T(n)由递归式T(n)=aT(n/b) + f(n)对非负整数定义
    - 其中n/b是指$\lceil n/b \rceil$或 $\lfloor n/b \rfloor$.
  - 那么T(n)可能有如下渐进界：
    - 1. 对于某常数$\varepsilon > 0$，如果$f(\mathrm{n}) = O\left(\mathrm{n}^{(\log_b a)-\varepsilon}\right)$,则有：$\mathrm{T}(n) = \Theta(n^{\log_b a})$
    - 2. 若$f(\mathrm{n}) = O\left(\mathrm{n}^{\log_b a}\right)$, 则有： $\mathrm{T}(n) = \Theta(n^{\log_b a} lgn)$
    - 3. 对于某常数$\varepsilon > 0$，如果$f(\mathrm{n}) = \Omega\left(\mathrm{n}^{(\log_b a)+\varepsilon}\right)$，且对常数c<1与所有足够大的n，有$af(n/b) \leq cf(n)$，则有：$\mathrm{T}(n) = \Theta(f(n))$

主方法

- $T(n)=aT(n/b) + f(n)$
- 将 $f(n)$和 $n^{\log_b a}$ 进行比较. 谁大就由谁决定
  - Case 1, $n^{\log_b a}$ 更大， 则$T(n) = \Theta(n^{\log_b a})$
  - Case 3, f(n)更大，则$T(n) = \Theta(f(n))$
  - Case 3,两个函数相同大小，则乘以一个对数因子

$$T(n) = \Theta\left(n^{\log_b a} \lg n\right) = \Theta(f(n) \lg n)$$

$$(af(n/b) \le cf(n), c<1)$$

主方法

- $\mathrm{T}(n) = \mathrm{a}T\left(\frac{n}{b}\right) + f(n)$, a≥1 和b>1，存在 $\varepsilon > 0, c < 1$
  - $\mathrm{T}(n) = \begin{cases} \Theta(n^{\log_b a}) & , f(n) = O(n^{\log_b a-\varepsilon}) \\ \Theta(n^{\log_b a} lgn) & , f(n) = \Theta(n^{\log_b a}) \\ \Theta(f(n)) & , f(n) = \Omega(n^{\log_b a+\varepsilon}) \text{ and } af\left(\frac{n}{b}\right) < cf(n), c < 1 \end{cases}$
- 例1 ：T(n)=9T(n/3)+n

$$a = 9, b = 3, f(n) = n \quad \Rightarrow \quad n^{\log_b a} = n^{\log_3 9} = n^2 = \Theta(n^2)$$

$$\Rightarrow \quad f(n) = O(n^{\log_3 9-\varepsilon}), \text{where } \varepsilon = 1 \quad \Rightarrow \quad T(n) = \Theta(n^2)$$

主方法

- $T(n) = aT\left(\frac{n}{b}\right) + f(n)$, a≥1 和b>1，存在 $\varepsilon > 0, c < 1$
- $$T(n) = \begin{cases} \Theta\left(n^{\log_b a}\right) & , f(n) = O\left(n^{\log_b a - \varepsilon}\right) \\ \Theta\left(n^{\log_b a} lgn\right) & , f(n) = \Theta\left(n^{\log_b a}\right) \\ \Theta\left(f(n)\right) & , f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right) \textit{ and } af\left(\frac{n}{b}\right) < cf(n), c < 1 \end{cases}$$
- 例2 ： T(n)=T(2n/3)+1

$$a = 1, b = 3/2, f(n) = 1 \quad \Rightarrow \quad n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$$

$$\Rightarrow \quad f(n) = \Theta(n^{\log_3 a}) = \Theta(1) \quad \Rightarrow \quad T(n) = \Theta(\lg n)$$

主方法

- $T(n) = aT\left(\frac{n}{b}\right) + f(n)$, a≥1 和b>1，存在 $\varepsilon > 0, c < 1$
  - $T(n) = \begin{cases} \Theta\left(n^{\log_b a}\right) & , f(n) = O\left(n^{\log_b a-\varepsilon}\right) \\ \Theta\left(n^{\log_b a} lgn\right) & , f(n) = \Theta\left(n^{\log_b a}\right) \\ \Theta\left(f(n)\right) & , f(n) = \Omega\left(n^{\log_b a+\varepsilon}\right) \text{ and } af\left(\frac{n}{b}\right) < cf(n), c < 1 \end{cases}$
- 例3：T(n)=3T(n/4)+nlgn

$$a = 3, b = 4, f(n) = n\lg n \quad \Rightarrow \quad n^{\log_b a} = n^{\log_4 3} = O(n^{0.793})$$

$$\Rightarrow \quad f(n) = \Omega(n^{(\log_4 3)+\varepsilon}), \text{where } \varepsilon \approx 0.2, \text{ and for sufficiently large } n,$$

$$af(n/b) = 3(n/4)\lg(n/4) \le (3/4)n\lg n = cf(n) \text{ for } c = 3/4$$

$$\Rightarrow \quad T(n) = \Theta(n\lg n)$$

主定理之特殊情况

- $T(n) = aT\left(\frac{n}{b}\right) + f(n)$, a≥1 和b>1，存在 $\varepsilon > 0, c < 1$

  - $$T(n) = \begin{cases} \Theta\left(n^{\log_b a}\right) & , f(n) = O(n^{\log_b a-\varepsilon}) \\ \Theta\left(n^{\log_b a} lgn\right) & , f(n) = \Theta(n^{\log_b a}) \\ \Theta\left(f(n)\right) & , f(n) = \Omega\left(n^{\log_b a+\varepsilon}\right) \text{ and } af\left(\frac{n}{b}\right) < cf(n), c < 1 \end{cases}$$

- 注意：不能涵盖所有情况
  - Case 1, f(n) 必须多项式级小于 $n^{\log_b a}$
  - Case 3, f(n) 必须多项式级大于$n^{\log_b a}$
  - 第1种和第2种情况之间存在一个gap, 当f(n)小于$n^{\log_b a}$，但不是多项式级小
  - 同样，第2种和第3种情况之间存在一个gap ，当f(n)大于$n^{\log_b a}$,但不是不是多项式级大

主定理之特殊情况

- $T(n) = aT\left(\frac{n}{b}\right) + f(n)$, a≥1 和b>1，存在 $\varepsilon > 0, c < 1$
  - $T(n) = \begin{cases} \Theta\left(n^{\log_b a}\right) & , f(n) = O\left(n^{\log_b a-\varepsilon}\right) \\ \Theta\left(n^{\log_b a} lgn\right) & , f(n) = \Theta\left(n^{\log_b a}\right) \\ \Theta\left(f(n)\right) & , f(n) = \Omega\left(n^{\log_b a+\varepsilon}\right)\ and\ af\left(\frac{n}{b}\right) < cf(n), c < 1 \end{cases}$

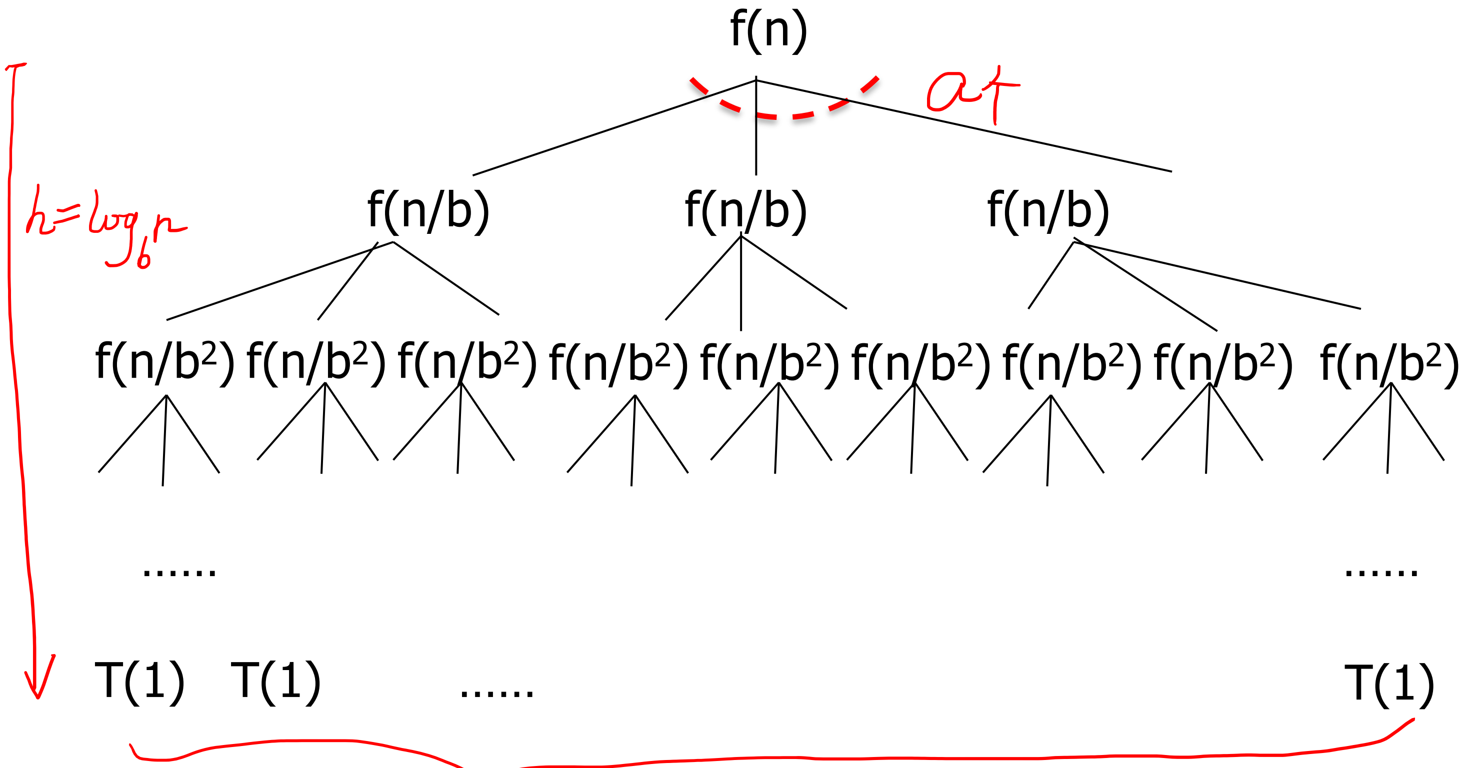
- 例4 ：T(n)=2T(n/2)+nlgn

a=2, b=2, f(n)=nlgn =>$n^{\log_b a}$=n

但是f(n)/n=lgn, 对于任意的正常数c, 这不是多项式级小于n^{ε}，也就是f(n)不是多项式级小于n，落到case 2和case 3之间的gap

主方法

- 主定理法的递归树演示 $T(n) = aT(n/b) + f(n)$



代价

$f(n)$

$af(n/b)$

$a^2f(n/b^2)$

$a^if(n/b^i)$

$n^{\log_b a}T(1)$

$\frac{n}{b^i}=1$

即 $i=\log_b n$

$a^h = a^{\log_b n} = n^{\log_b a}$

小结

- 递归概念
  - 两个基本要素：边界条件，递归函数
  - 递归与分治的关系
- 递归表达式的求解方法
- 案例分析
  - 大整数乘法
  - 阶乘、合并排序
    - 递归算法的分析关键得到递归表达式
  - 整数划分问题
    - 转化为递归的问题进行求解
  - 汉诺塔问题

小结

- 重点和难点：
  - 分治法的基本思想
  - 递归概念、两个要素
    - 尤其递归特征不明显的问题怎么进行转换思维
  - 递归方程的求解方法
    - 代换法
    - 迭代法
    - 主定理法