

第二章

算法分析的数学基础

《Introduction to Algorithms》

- 第三章
- 第四章
- 附录

《Concrete Mathematics: A Foundation for Computer Science》

Ronald L. Graham, Donald E. Knuth, and Oren Patashnik

2.1 计算复杂性函数的阶

2.2 和式的计算与估计

2.3 递归方程

2.1 计算复杂性函数的阶

2.2 和式的计算与估计

2.3 递归方程

2.1 计算复杂性函数的阶

- 计算复杂性函数的阶
 - 算法执行时间增长的阶（增长率）
 - 执行时间函数的主导项

例如：

$$T(n)=an^2+bn+c$$

主导项： an^2

当输入大小 n 较大时，其它低阶项相对来说意义不大
系数 a 也相对来说意义不大

即：函数 $T(n)$ 的阶为 n^2

2.1 计算复杂性函数的阶

2.1.1 同阶函数

2.1.2 低阶函数

2.1.3 高阶函数

2.1.4 严格低阶函数

2.1.5 严格高阶函数

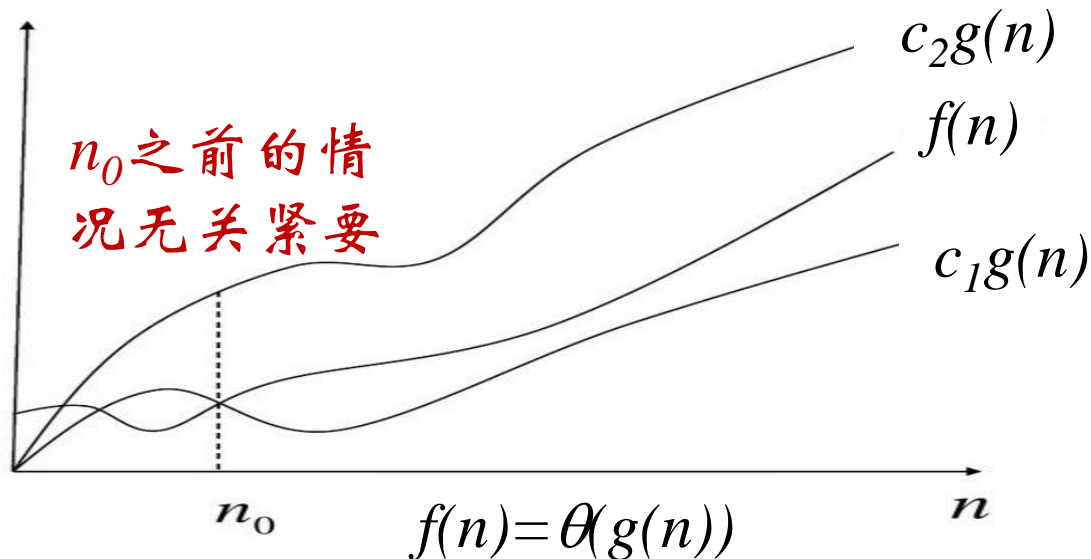
2.1.6 函数阶的性质

2.1.1 同阶函数集合

定义2.1.1. 设 $f(n)$ 和 $g(n)$ 是**正值**函数。如果 $\exists c_1, c_2 > 0, n_0$, $\forall n > n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)$, 则称 $f(n)$ 与 $g(n)$ 同阶, 记作 $f(n) = \Theta(g(n))$ 。

$\Theta(g(n))$ 可以视为所有与 $g(n)$ 同阶的函数集合:

$$\{f(n) \mid \exists c_1, c_2 > 0, n_0, \forall n > n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$



2.1.1 同阶函数集合

• 例1, 证明: $(1/3)n^2 - 3n = \theta(n^2)$

$\exists c_1, c_2 > 0, n_0, \forall n > n_0,$

$$c_1 n^2 \leq (1/3)n^2 - 3n \leq c_2 n^2$$

对于左侧不等式, $\forall n > 1$, 有:

$$c_1 \leq (1/3) - 3/n = (1/6) + (1/6) - 3/n$$

即当 $n > 18$, $c_1 = 1/6$ 时, 不等式成立

对于右侧不等式, $\forall n > 1$, 有: $(1/3) - 3/n \leq c_2$,

即当 $n > 18$, $c_2 = 1/3$ 时, 不等式成立

2.1.1 同阶函数集合

例2 证明 $6n^3 \neq \theta(n^2)$

证. 如果存在 $c_1, c_2 > 0$, n_0 使得当 $n \geq n_0$ 时,
 $c_1 n^2 \leq 6n^3 \leq c_2 n^2$,

即 $c_1 \leq 6n \leq c_2$, $n \leq c_2/6$ 。

于是, 当 $n > c_2/6$ 时 与 $n \leq c_2/6$ 矛盾。

2.1.1 同阶函数集合

例3 证明 $f(n) = an^2 + bn + c = \theta(n^2)$,
其中 $a > 0$ 。

证. 往构造 $c_1, c_2 > 0, n_0$, 使得 $n > n_0$ 时,
 $c_1 n^2 \leq an^2 + bn + c \leq c_2 n^2$ 成立.

令 $c_1 = a/4, c_2 = 7a/4$, 构造 n_0 使得
 $an^2/4 \leq an^2 + bn + c \leq 7an^2/4$ 成立.

当 $n_0 = 2\max\{\frac{|b|}{a}, \sqrt{|c|/a}\}$ 时, 上述不等式
成立。得证。

命题2.1.1: 对于任意正整数 d 和任意常数 $a_d > 0$, 有:

$$P(n) = \sum_{i=0}^d a_i n^i = \theta(n^d).$$

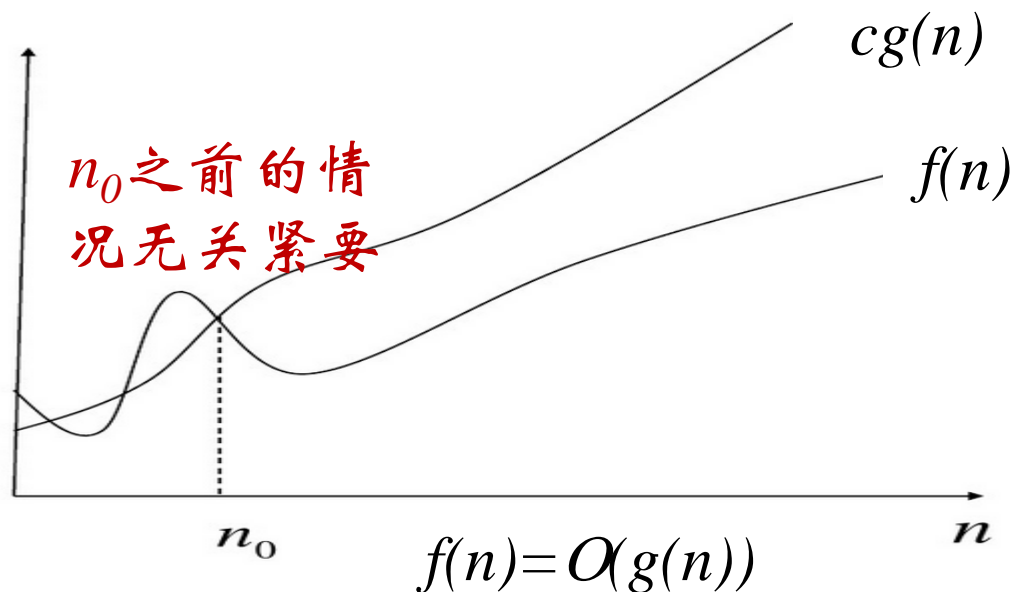
请完成上述命题的证明.

2.1.2 低阶函数集合

定义2.1.2. 设 $f(n)$ 和 $g(n)$ 是正值函数。如果 $\exists c > 0, n_0$, $\forall n > n_0, f(n) \leq cg(n)$, 则称 $f(n)$ 比 $g(n)$ 低阶或 $g(n)$ 是 $f(n)$ 的上界, 记作 $f(n) = O(g(n))$ 。

$O(g(n))$ 可以视为所有比 $g(n)$ 低阶的函数的集合:

$$\{f(n) \mid \exists c, n_0, \forall n > n_0, f(n) \leq cg(n)\}$$



2.1.2 低阶函数集合

例1. $f(n) = \theta(g(n)) \Rightarrow f(n) = O(g(n))$

例2. 证明 $n = O(n^2)$.

证. 取 $c = 1, n_0 = 1$, 则当 $n \geq n_0$ 时, 有

$$0 \leq n \leq cn^2$$

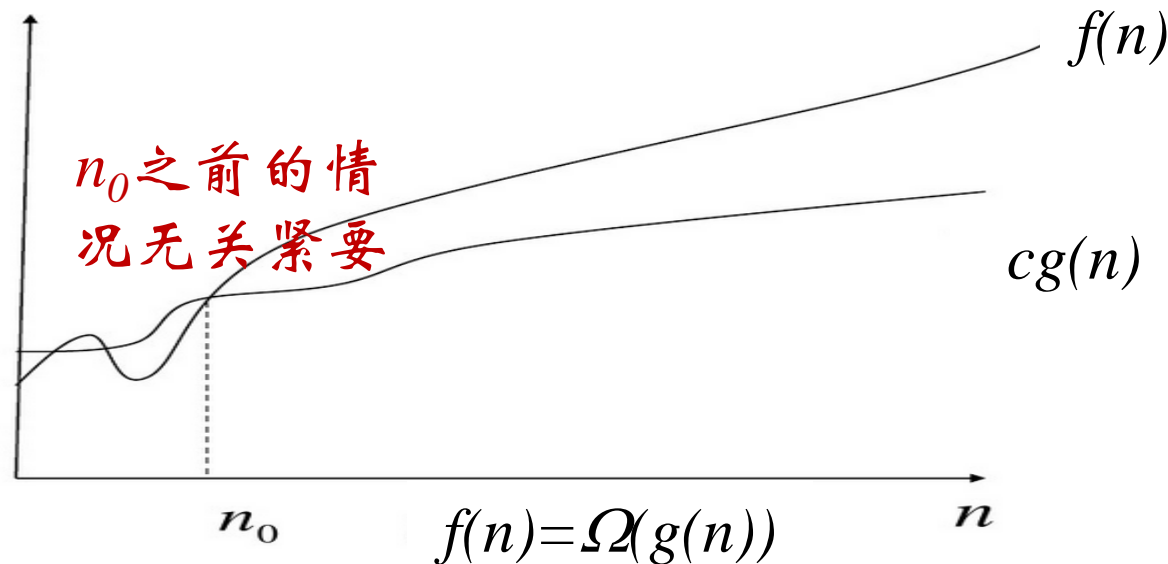
得证.

2.1.3 高阶函数集合

定义2.1.3. 设 $f(n)$ 和 $g(n)$ 是正值函数。如果 $\exists c > 0, n_0$, $\forall n > n_0, f(n) \geq cg(n)$, 则称 $f(n)$ 比 $g(n)$ 高阶或 $g(n)$ 是 $f(n)$ 的下界, 记作 $f(n) = \Omega(g(n))$ 。

$\Omega(g(n))$ 可以视为所有比 $g(n)$ 高阶的函数集合:

$$\{f(n) \mid \exists c, n_0, \forall n > n_0, f(n) \geq cg(n)\}$$



θ 、 O 、 Ω 之间的关系

- θ 表示渐进紧界
- O 表示渐进上界
- Ω 表示渐进下界
- $\theta(g(n)) \subseteq O(g(n))$
- $f(n) = \theta(g(n)) \Rightarrow f(n) = O(g(n))$
- $f(n) = an^2 + bn + c = \theta(n^2), f(n) = O(n^2)$
- $an + b = O(n^2)$. 为什么?
- $n = O(n^2) !!!$

如果 $f(n) = O(n^k)$, 则称 $f(n)$ 是多项式界限的

θ , O , Ω 之间的关系

- 一些讨论:

- 当我们谈到插入排序的最坏运行时间是 $O(n^2)$, 这个结论适用于所有的输入, 即使对于已经排序的输入也成立, 因为 $O(n) \subseteq O(n^2)$.
- 然而插入排序的最坏运行时间 $\theta(n^2)$ 不能应用到每个输入, 因为对于已经排序的输入, $\theta(n) \neq \theta(n^2)$.

θ , O , Ω 之间的关系

- Ω 用来描述运行时间的最好情况
- 对于插入排序，我们可以说
 - 最好运行时间是 $\Omega(n)$
 - 或者说，运行时间是 $\Omega(n)$
 - 插入排序算法的运行时间在 $\Omega(n)$ 和 $O(n^2)$ 之间
 - 插入排序算法的最坏运行时间是 $\Omega(n^2)$
 - 但说插入排序算法的运行时间是 $\Omega(n^2)$ ，是错误的！

极少用 Ω 来描述算法的运行时间和复杂性

θ, O, Ω 之间的关系

定理2.1. 对于任意 $f(n)$ 和 $g(n)$, $f(n) = \theta(g(n))$
当且仅当 $f(n) = O(g(n))$ 而且 $f(n) = \Omega(g(n))$.

证. \Rightarrow : 如果 $f(n) = \theta(g(n))$, 则 $\exists c_1, c_2 > 0, n_0 > 0$, 使得
 $n \geq n_0$ 时, $c_1 g(n) \leq f(n) \leq c_2 g(n)$ 成立. 这也必然使得
 $f(n) = O(g(n))$ 和 $f(n) = \Omega(g(n))$ 均成立.

\Leftarrow : 如果 $f(n) = O(g(n))$ 而且 $f(n) = \Omega(g(n))$, 则由
 $f(n) = O(g(n))$, 可知 $\exists c_1 > 0, n_1 > 0$, 使得 $n \geq n_1$ 时,
 $f(n) \leq c_1 g(n)$ 成立; 又由 $f(n) = \Omega(g(n))$, 可知 $\exists c_2 > 0, n_2 > 0$, 使得 $n \geq n_2$ 时, $f(n) \geq c_2 g(n)$ 成立.

因此, $\exists c_1, c_2 > 0, n_0 = \max\{n_1, n_2\} > 0$, 使得 $n \geq n_0$ 时,
 $c_1 g(n) \leq f(n) \leq c_2 g(n)$ 成立.

2.1.4 严格低阶函数集合

定义2.1.4. 设 $f(n)$ 和 $g(n)$ 是正值函数。如果
 $\forall c > 0, \exists n_0, \forall n > n_0, f(n) < cg(n)$, 则称 $f(n)$ 严格比
 $g(n)$ 低阶或 $g(n)$ 是 $f(n)$ 的严格上界, 记作
 $f(n) = o(g(n))$ 。

$o(g(n))$ 可以视为所有比 $g(n)$ 严格低阶的函数集合:

$$\{f(n) \mid \forall c, \exists n_0, \forall n > n_0, f(n) < cg(n)\}$$

2.1.4 严格低阶函数集合

关于低阶 O 与严格低阶 o 的进一步说明

- O 标记可能是或不是紧的
 - $2n^2 = O(n^2)$ 是紧的, 但 $2n = O(n^2)$ 不是紧的.
- o 标记用于标记上界但不是紧的情况
 - $2n = o(n^2)$, 但是 $2n^2 \neq o(n^2)$.
- 区别: 某个正常数 c 在 O 标记中, 但所有正常数 c 在 o 标记中.

例1. 证明 $2n = o(n^2)$

证. 对 $\forall c > 0$, 要得到 $2n < cn^2$, 其中 $n > 0$, 只需满足 $2 < cn$, 即 $n > \frac{2}{c}$ 。

因此, 对 $\forall c > 0$, 存在 $n_0 = \frac{2}{c}$, 当 $n > n_0$ 时, $2n < cn^2$ 成立。

例2. 证明 $2n^2 \neq o(n^2)$

证. 当 $c = 1$ 时, 不存在任何 n_0 , 使得 $n > n_0$ 时, $2n^2 < cn^2 = n^2$ 成立。

命题2.1.2. $f(n) = o(g(n)) \Rightarrow \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0.$

证. 由于 $f(n) = o(g(n))$, 对于任意 $\varepsilon > 0$, 存在 $n_0 > 0$, 当 $n \geq n_0$ 时, $0 < f(n) < \varepsilon g(n)$ 成立, 即 $0 < \frac{f(n)}{g(n)} < \varepsilon.$

于是, 有 $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0.$

2.1.5 严格高阶函数集合

定义2.1.4. 设 $f(n)$ 和 $g(n)$ 是正值函数。如果
 $\forall c > 0, \exists n_0, \forall n > n_0, f(n) > cg(n)$, 则称 $f(n)$ 严格比 $g(n)$
高阶或 $g(n)$ 是 $f(n)$ 的严格下界, 记作 $f(n) = \omega(g(n))$ 。

$\omega(g(n))$ 可以视为所有比 $g(n)$ 严格高阶的函数集合:
 $\{f(n) \mid \forall c, \exists n_0, \forall n > n_0, f(n) > cg(n)\}$

命题2.1.3. $f(n) = \omega(g(n))$ 当且仅当 $g(n) = o(f(n))$

证. \Rightarrow : 对 $\forall c > 0$, 有 $\frac{1}{c} > 0$. 由 $f(n) = \omega(g(n))$ 可知, 对 $\forall \frac{1}{c} > 0$, 存在 $n_0 > 0$, 当 $n \geq n_0$ 时, $f(n) > \frac{1}{c} g(n)$, 即 $g(n) < cf(n)$ 成立. 于是有 $g(n) = o(f(n))$.

\Leftarrow : 对 $\forall c > 0$, 有 $\frac{1}{c} > 0$. 由 $g(n) = o(f(n))$ 可知, 对 $\forall \frac{1}{c} > 0$, 存在 $n_0 > 0$, 当 $n \geq n_0$ 时, $g(n) < \frac{1}{c} f(n)$, 即 $f(n) > cg(n)$ 成立. 于是有 $f(n) = \omega(g(n))$.

命题2.1.4. $f(n) = \omega(g(n)) \Rightarrow \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$.

证. 对 $\forall c > 0$, 由于 $f(n) = \omega(g(n))$, 必然存在 $n_0 > 0$, 使得当 $n > n_0$ 时, $f(n) > cg(n)$.

即对 $\forall c > 0$, 必然存在 $n_0 > 0$, 当 $n > n_0$ 时, $\frac{f(n)}{g(n)} > c$, 于

是可得 $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$.

2.1.6 函数阶的性质

A. 传递性:

$$\text{➤ } f(n) = \theta(g(n)) \wedge g(n) = \theta(h(n)) \Rightarrow f(n) = \theta(h(n))$$

$$\text{➤ } f(n) = O(g(n)) \wedge g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$$

$$\text{➤ } f(n) = \Omega(g(n)) \wedge g(n) = \Omega(h(n)) \Rightarrow f(n) = \Omega(h(n))$$

$$\text{➤ } f(n) = o(g(n)) \wedge g(n) = o(h(n)) \Rightarrow f(n) = o(h(n))$$

$$\text{➤ } f(n) = \omega(g(n)) \wedge g(n) = \omega(h(n)) \Rightarrow f(n) = \omega(h(n))$$

2.1.6 函数阶的性质(续)

B. 自反性:

➤ $f(n) = \theta(f(n))$

➤ $f(n) = O(f(n))$

➤ $f(n) = \Omega(f(n))$

C. 对称性:

➤ $f(n) = \theta(g(n))$ 当且仅当 $g(n) = \theta(f(n))$

D. 反对称性:

➤ $f(n) = O(g(n))$ 当且仅当 $g(n) = \Omega(f(n))$

➤ $f(n) = o(g(n))$ 当且仅当 $g(n) = \omega(f(n))$

所有函数都是可比的吗？？？

$f(n) = n$ 与 $g(n) = n^{1+\sin(n)}$ 可比吗？

数学公式中函数阶的含义

$n = O(n^2)$ 表示 $f(n) = n, f(n) \in O(n^2)$

数学公式中的函数阶, 独立地表示一个匿名函数, 例如:
 $2n^2 + 3n + 1 = 2n^2 + \theta(n)$ 中, $\theta(n)$ 代表一个函数, 它是 $\theta(n)$ 中的一个元素, 并且使得公式中的等号相等, 此处它代表的是 $f(n) = 3n + 1$.

等号左侧和右侧都出现函数的阶, 表示无论如何选择等号左侧的匿名函数, 都存在一个等号右侧匿名函数的选择方法, 使得公式中的等号成立.

例如: $2n^2 + \theta(n) = \theta(n^2)$, 表示无论如何选择一个 $f(n) \in \theta(n)$, 都存在 $g(n) \in \theta(n^2)$, 使得 $2n^2 + f(n) = g(n)$.

$2n^2 + 3n + 1 = 2n^2 + \theta(n) = \theta(n^2)$ 的含义? 分别处理
两个等号

2.1 计算复杂性函数的阶

2.2 和式的计算与估计

2.3 递归方程

2.2 和式的估计与界限

1. 线性和

命题 2.3.1. $\sum_{k=1}^n (ca_k + b_k) = c \sum_{k=1}^n a_k + \sum_{k=1}^n b_k$

命题 2.3.2. $\sum_{k=1}^n \theta(f(k)) = \theta(\sum_{k=1}^n f(k))$

请完成命题2.3.2的证明.

2. 级数

命题 2.3.3
$$\sum_{i=1}^n i = \frac{n(n+1)}{2} = \theta(n^2)$$

命题 2.3.4
$$\sum_{k=0}^n x^k = 1 + x + x^2 + \dots + x^n = \frac{x^{n+1} - 1}{x - 1} \quad (x \neq 1)$$

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

命题 2.3.5
$$H_n = \sum_{k=1}^n \frac{1}{k} = \ln n + O(1)$$

命题 2.3.6 $\sum_{k=1}^n (a_k - a_{k-1}) = a_n - a_0 .$

命题 2.3.7 $\sum_{k=0}^{n-1} (a_k - a_{k+1}) = a_0 - a_n$

命题 2.3.8 $\sum_{k=1}^{n-1} \frac{1}{k(k+1)} = \sum_{k=1}^{n-1} \left(\frac{1}{k} - \frac{1}{k+1} \right) = 1 - \frac{1}{n}$

命题 2.3.9 $\lg(\prod_{k=1}^n a_k) = \sum_{k=1}^n \lg a_k$

3. 直接求和的界限

例1. $\sum_{k=1}^n k \leq \sum_{k=1}^n n = n^2$

例2. $\sum_{k=1}^n a_i \leq n \times \max\{a_k\}.$

例3. 设对于所有 $k \geq 0$, $a_{k+1}/a_k \leq r < 1$, 求 $\sum_{k=0}^n a_k$ 的上界.

解: $a_1/a_0 \leq r \Rightarrow a_1 \leq a_0 r,$

$$a_2/a_1 \leq r \Rightarrow a_2 \leq a_1 r \leq a_0 r^2,$$

$$a_3/a_2 \leq r \Rightarrow a_3 \leq a_2 r \leq a_0 r^3 \dots\dots$$

$$a_k/a_{k-1} \leq r \Rightarrow a_k \leq a_{k-1} r \leq a_0 r^k$$

于是, $\sum_{k=0}^n a_k \leq \sum_{k=0}^{\infty} a_0 r^k = a_0 \sum_{k=0}^{\infty} r^k = \frac{a_0}{1-r}.$

例 4. 求 $\sum_{k=1}^{\infty} (k/3^k)$ 的界

解. 使用例3中的方法. $\frac{\frac{k+1}{3^{k+1}}}{\frac{k}{3^k}} = \frac{1}{3} \times \frac{k+1}{k} \leq \frac{2}{3} = r.$

$$\begin{aligned} \text{因此, } \sum_{k=1}^{\infty} k/3^k &\leq \sum_{k=1}^{\infty} a_1 r^{k-1} = \sum_{k=1}^{\infty} \frac{1}{3} \left(\frac{2}{3}\right)^{k-1} \\ &= \frac{1}{3} \times \frac{1}{1-\frac{2}{3}} = 1 \end{aligned}$$

例 5. 用分裂和的方法求 $\sum_{k=1}^n k$ 的下界.

解: $\sum_{k=1}^n k = \sum_{k=1}^{n/2} k + \sum_{k=n/2+1}^n k \geq \sum_{k=1}^{n/2} 0 + \sum_{k=n/2+1}^n n/2 \geq \left(\frac{n}{2}\right)^2 = \Omega(n^2).$

例 6. 求 $\sum_{k=0}^{\infty} \frac{k^2}{2^k}$ 的上界

解：当 $k \geq 3$ 时， $\frac{(k+1)^2 / 2^{k+1}}{k^2 / 2^k} = \frac{(k+1)^2}{2k^2} \leq \frac{8}{9}$

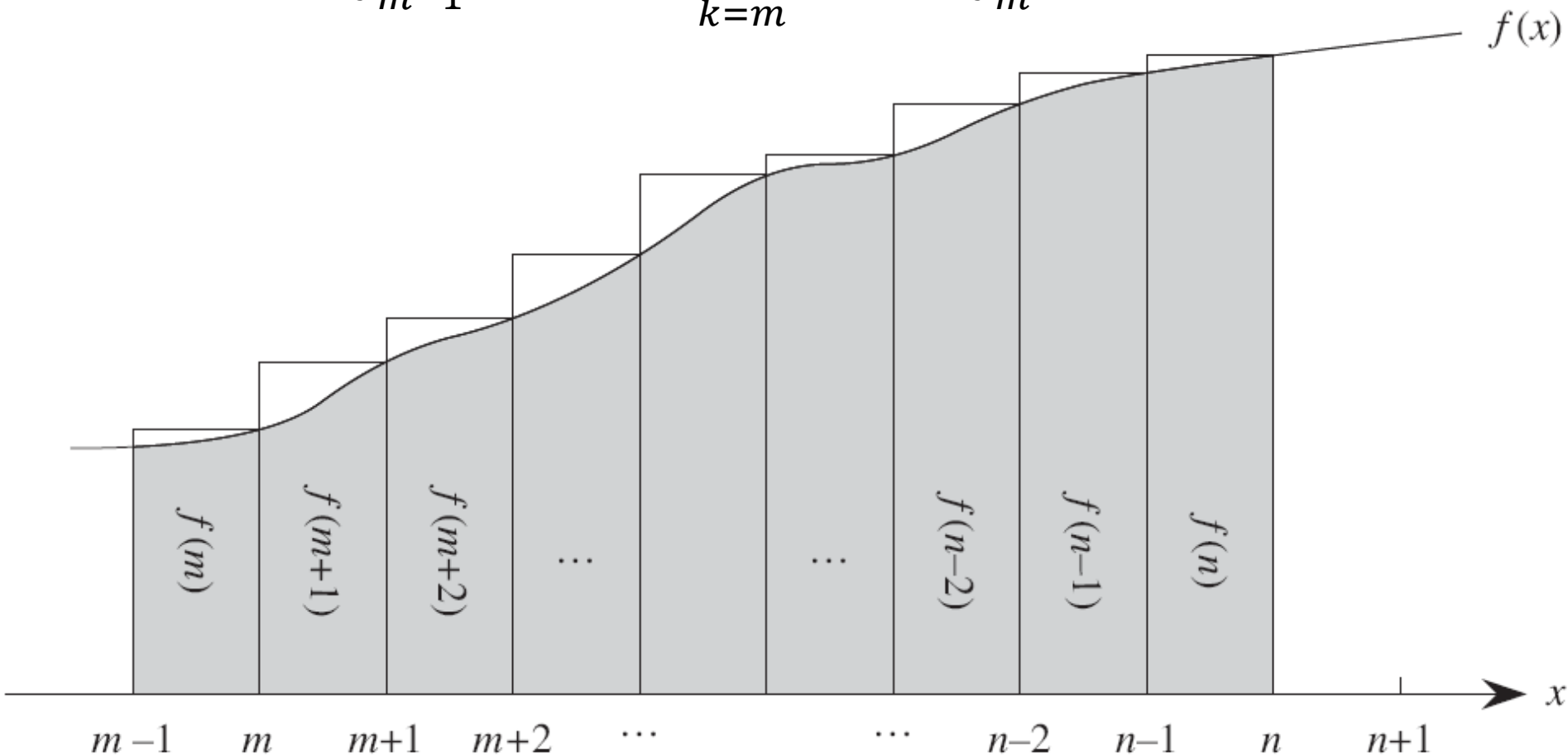
$$\begin{aligned} \text{于是, } \sum_{k=0}^{\infty} \frac{k^2}{2^k} &= \sum_{k=0}^2 \frac{k^2}{2^k} + \sum_{k=3}^{\infty} \frac{k^2}{2^k} \\ &\leq \theta(1) + \sum_{k=1}^{\infty} \frac{9}{8} \left(\frac{8}{9}\right)^k = O(1) \end{aligned}$$

例 7. 求 $H_n = \sum_{k=1}^n \frac{1}{k}$ 的上界

解:
$$\begin{aligned} \sum_{k=1}^n \frac{1}{k} &= \frac{1}{1} + \left(\frac{1}{2} + \frac{1}{3} \right) + \left(\frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} \right) \\ &\quad + \left(\frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \frac{1}{11} + \frac{1}{12} + \frac{1}{13} + \frac{1}{14} + \frac{1}{15} \right) + \dots \\ &\leq \sum_{i=0}^{\lceil \lg n \rceil} \sum_{j=0}^{2^i-1} \frac{1}{2^i + j} \leq \sum_{i=0}^{\lceil \lg n \rceil} \sum_{j=0}^{2^i-1} \frac{1}{2^i} \leq \sum_{i=0}^{\lceil \lg n \rceil} 1 \leq \lg n + 1 \end{aligned}$$

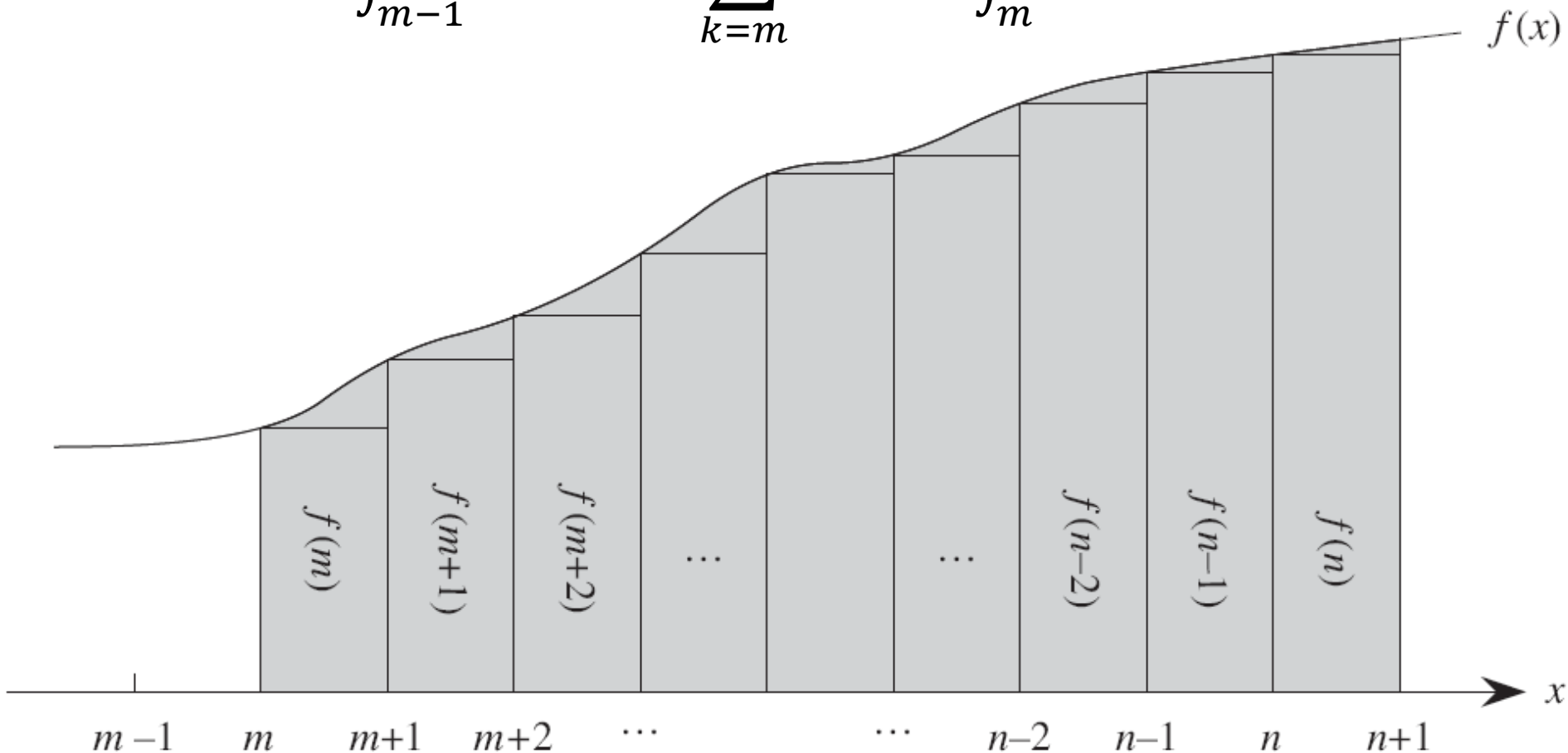
例8. 如果 $f(k)$ 单调递增, 则

$$\int_{m-1}^n f(x)dx \leq \sum_{k=m}^n f(k) \leq \int_m^{n+1} f(x)dx$$



例8. 如果 $f(k)$ 单调递增, 则

$$\int_{m-1}^n f(x)dx \leq \sum_{k=m}^n f(k) \leq \int_m^{n+1} f(x)dx$$



例9. 如果 $f(k)$ 单调递减, 则

$$\int_m^{n+1} f(x)dx \leq \sum_{k=m}^n f(k) \leq \int_{m-1}^n f(x)dx$$

例10.

$$\sum_{k=1}^n \frac{1}{k} \geq \int_1^{n+1} \frac{1}{x} dx = \ln(n+1)$$

$$\sum_1^n \frac{1}{k} \leq 1 + \sum_2^n \frac{1}{k} \leq 1 + \int_1^n \frac{1}{x} dx \leq \ln n + 1$$

2.1 计算复杂性函数的阶

2.2 和式的计算与估计

2.3 递归方程

2.3 递归方程

- 递归方程: 递归方程是使用具有较小输入值的相同方程来描述一个方程。

用自身来定义自身

- 递归方程例: Merge-sort排序算法的复杂性方程

$$T(n) = 2T(n/2) + \theta(n) \quad \text{if } n > 1.$$

$$T(n) = \theta(1) \quad \text{if } n = 1 (\text{边界条件})$$

$T(n)$ 的解是 $\theta(n \log n)$

边界条件是根据问题的不同而不同的!

求解递归方程的三个主要方法

- 替换方法：
 - 先猜测方程的解，
 - 然后用数学归纳法证明。
- 迭代方法：
 - 把方程转化为一个和式
 - 然后用估计和的方法来求解。
- Master方法：
 - 求解型为 $T(n)=aT(n/b)+f(n)$ 的递归方程

2.3.1 替换(Substitution)方法

Substitution方法I: 联想已知的 $T(n)$

例1. 求解 $T(n)=2T(n/2 + 17) + n$

解: 猜想, $T(n) = 2T\left(\frac{n}{2} + 17\right) + n$ 与 $T(n) = 2T\left(\frac{n}{2}\right) + n$ 只差一个17.

当 n 充分大时, $T\left(\frac{n}{2} + 17\right)$ 与 $T\left(\frac{n}{2}\right)$ 的差别并不大, 因为 $\frac{n}{2} + 17$ 和 $\frac{n}{2}$ 相差很小.

因此, 我们可以猜想 $T(n) = O(n \lg n)$.

证明: 用数学归纳法

2.3.1 替换(Substitution)方法

Substitution方法I: 联想已知的 $T(n)$

例1. 求解 $T(n) = 2T(n/2 + 17) + n$

解. 猜想 $T(n) = O(n \lg n)$, 往证存在常数 c, n_0 , 使 $n \geq n_0$ 时, 有 $T(n) \leq cn \lg n$. 假设 $m \leq n - 1$ 且 $m \geq n_0$ 时, 有 $T(m) \leq cm \lg m$.

因此 $T(n) = 2T(n/2 + 17) + n$

$$\leq 2c \left(\frac{n}{2} + 17 \right) \lg \left(\frac{n}{2} + 17 \right) + n$$

$$\leq 2c \left(\frac{n}{2} + 17 \right) \lg \left(\frac{3n}{4} \right) + n \quad (n \geq 68)$$

$$= cn \lg n - \lg \left(\frac{4}{3} \right) cn + 34c \lg n - 34c \lg \left(\frac{4}{3} \right) + n$$

令 $c = 2/\lg \left(\frac{4}{3} \right)$ 时, n 足够大使得 $\lg \left(\frac{4}{3} \right) n > 68 \lg n$ 时, $T(n) \leq cn \lg n$. ⁴⁵

Substitution 方法II: 猜测上下界, 减少不确定性范围

例 3. 求解 $T(n) = 2T\left(\frac{n}{2}\right) + n$.

解: 首先证明 $T(n) = \Omega(n)$, $T(n) = O(n^2)$

然后逐阶地降低上界、提高下界。

$\Omega(n)$ 的上一个阶是 $\Omega(n \log n)$,

$O(n^2)$ 的下一个阶是 $O(n \log n)$ 。

细微差别的处理

- 问题：猜测正确，数学归纳法的归纳步似乎证不出来
- 解决方法：从guess中减去一个低阶项，可能work.

例 4. 求解 $T(n) = T(\lfloor n/2 \rfloor) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + 1$

解：(1) 我们猜 $T(n) = O(n)$

$$\text{证： } T(n) \leq c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1 = cn + 1 \neq cn$$

证不出 $T(n) = O(cn)$

(2) 减去一个低阶项，猜 $T(n) \leq cn - b$ ， $b \geq 0$ 是常数

证：设当 $\leq n-1$ 时成立

$$\begin{aligned} T(n) &= T(\lfloor n/2 \rfloor) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + 1 \leq c\left\lfloor \frac{n}{2} \right\rfloor - b + c\left\lceil \frac{n}{2} \right\rceil - b + 1 \\ &= cn - 2b + 1 = cn - b - b + 1 \leq cn - b \quad (\text{只要 } b \geq 1)。 \end{aligned}$$

常数个 $O(n)$ 相加还在 $O(n)$ 中, 然而归纳次数可能并非常数!

避免陷阱

例 5. 求解 $T(n) = 2T(\lfloor n/2 \rfloor) + n$ 。

解: 猜 $T(n) = O(n)$

证: 用数学归纳法证明 $T(n) \leq cn$ 。

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

--错!!

错在哪里? 过早使用了 $O(n)$ 而陷入了陷阱!

应该在证明了 $T(n) \leq cn$ 后才可使用。

从 $T(n) \leq cn + n$ 不可能得到 $T(n) \leq cn$

因为对于 $\forall c > 0$, 我们都得不到 $cn + n \leq cn$ 。

Substitution 方法III: 变量替换

经变量替换把递归方程变换为熟悉的方程.

例 6. 求解 $T(n) = 2T(\sqrt{n}) + \lg n$

解: 令 $m = \lg n$, 则 $n = 2^m$, $T(2^m) = 2T\left(2^{\frac{m}{2}}\right) + m$.

令 $S(m) = T(2^m)$ 则 $T\left(2^{\frac{m}{2}}\right) = S\left(\frac{m}{2}\right)$. 于是, $S(m) = 2S\left(\frac{m}{2}\right) + m$.

显然, $S(m) = O(m \lg m)$, 即 $T(2^m) = O(m \lg m)$

由于 $2^m = n$, $m = \lg n$, $T(n) = O(\lg n \times \lg(\lg n))$.

2.3.2 迭代(Iteration)方法

方法：

循环地展开递归方程，
把递归方程转化为和式，
然后可使用求和技术解之。

例 1. $T(n) = n + 3T\left(\left\lfloor \frac{n}{4} \right\rfloor\right), \quad T(1)=1$

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

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

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

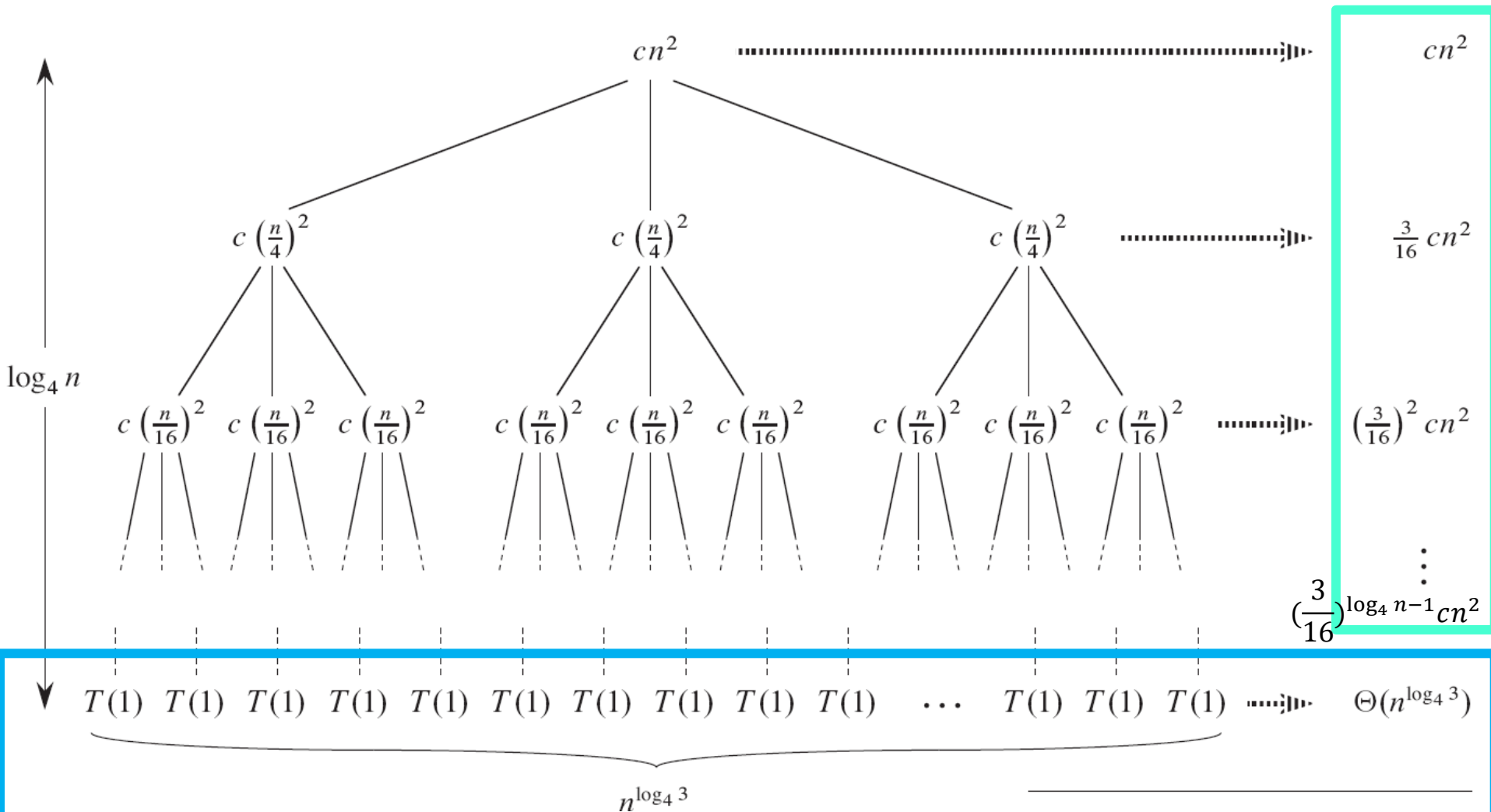
$$= n + 3\left\lfloor \frac{n}{4} \right\rfloor + 3^2\left\lfloor \frac{n}{4^2} \right\rfloor + 3^3\left(\left\lfloor \frac{n}{4^3} \right\rfloor\right) + \dots + 3^i T\left(\left\lfloor \frac{n}{4^i} \right\rfloor\right)$$

$$\boxed{\text{令 } \frac{n}{4^i} = 1 \Rightarrow 4^i = n \Rightarrow i = \log_4 n}$$

$$= n + 3\left\lfloor \frac{n}{4} \right\rfloor + 3^2\left\lfloor \frac{n}{4^2} \right\rfloor + 3^3\left(\left\lfloor \frac{n}{4^3} \right\rfloor\right) + \dots + 3^{\log_4 n} T(\lfloor 1 \rfloor)$$

$$\leq \sum_{i=0}^{\log_4 n} 3^i \frac{n}{4^i} \leq n \sum_{i=0}^{\infty} \left(\frac{3}{4}\right)^i = n \times \frac{1}{1 - \frac{3}{4}} = 4n = O(n)$$

求解 $T(n) = 3T\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + cn^2$



求解 $T(n) = 3T\left(\left\lfloor \frac{n}{4} \right\rfloor\right) + cn^2$

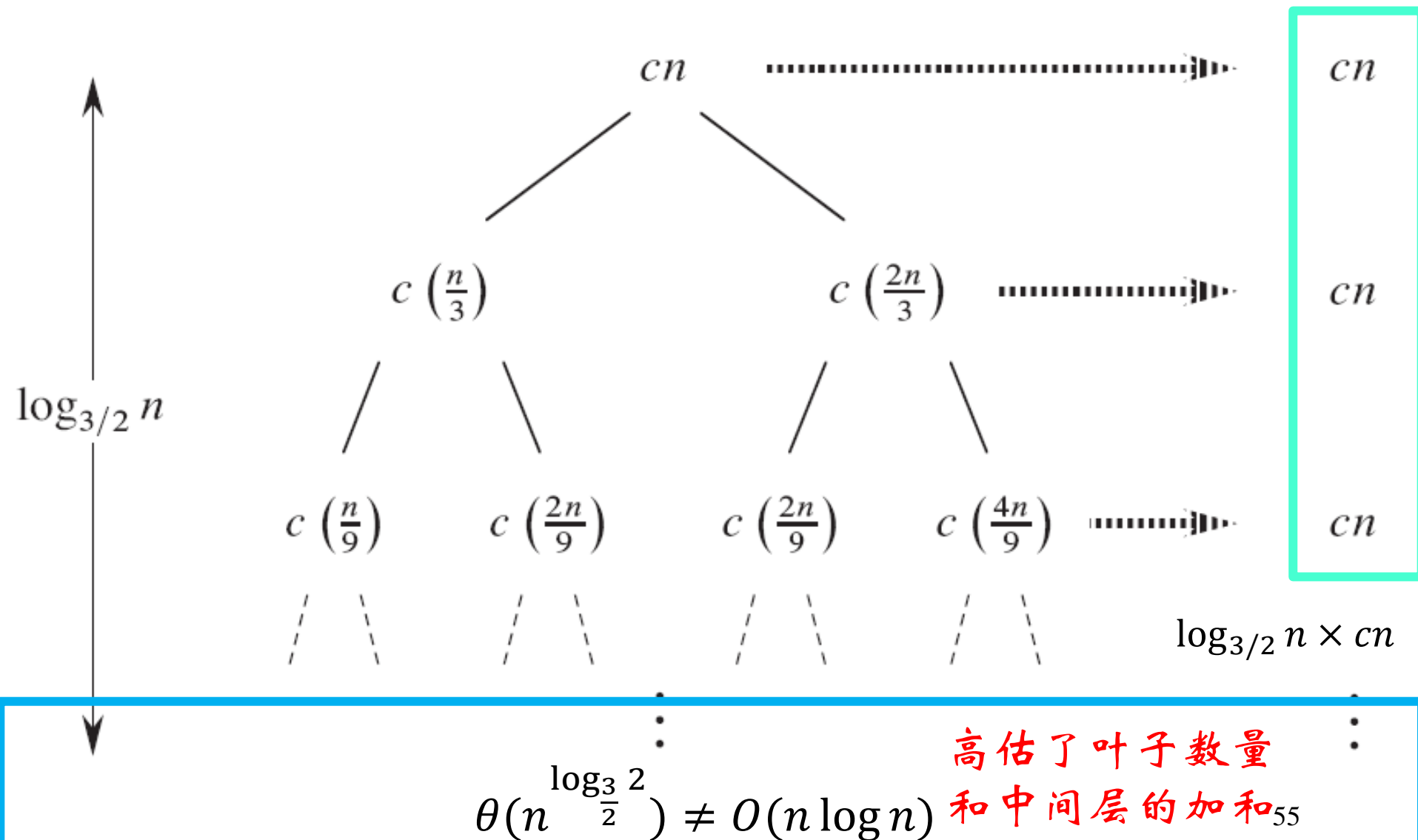
$$\begin{aligned}
 T(n) &= cn^2 + \frac{3}{16}cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2 + \theta(n^{\log_4 3}) \\
 &= \sum_{i=0}^{\log_4 n - 1} \left(\frac{3}{16}\right)^i cn^2 + \theta(n^{\log_4 3}) \\
 &\leq \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \theta(n^{\log_4 3}) = \frac{16}{13}cn^2 + \theta(n^{\log_4 3}) = O(n^2)
 \end{aligned}$$

$$\begin{aligned}
 &cn^2 \\
 &\vdots \\
 &\frac{3}{16}cn^2 \\
 &\vdots \\
 &\left(\frac{3}{16}\right)^2 cn^2 \\
 &\vdots \\
 &\left(\frac{3}{16}\right)^{\log_4 n - 1} cn^2
 \end{aligned}$$

\downarrow
 $T(1) \quad T(1) \quad T(1) \quad T(1) \quad T(1) \quad T(1) \quad T(1) \quad T(1) \quad T(1) \quad T(1) \quad \dots \quad T(1) \quad T(1) \quad T(1) \quad \dots \Rightarrow \Theta(n^{\log_4 3})$

$\underbrace{\hspace{15em}}_{n^{\log_4 3}}$

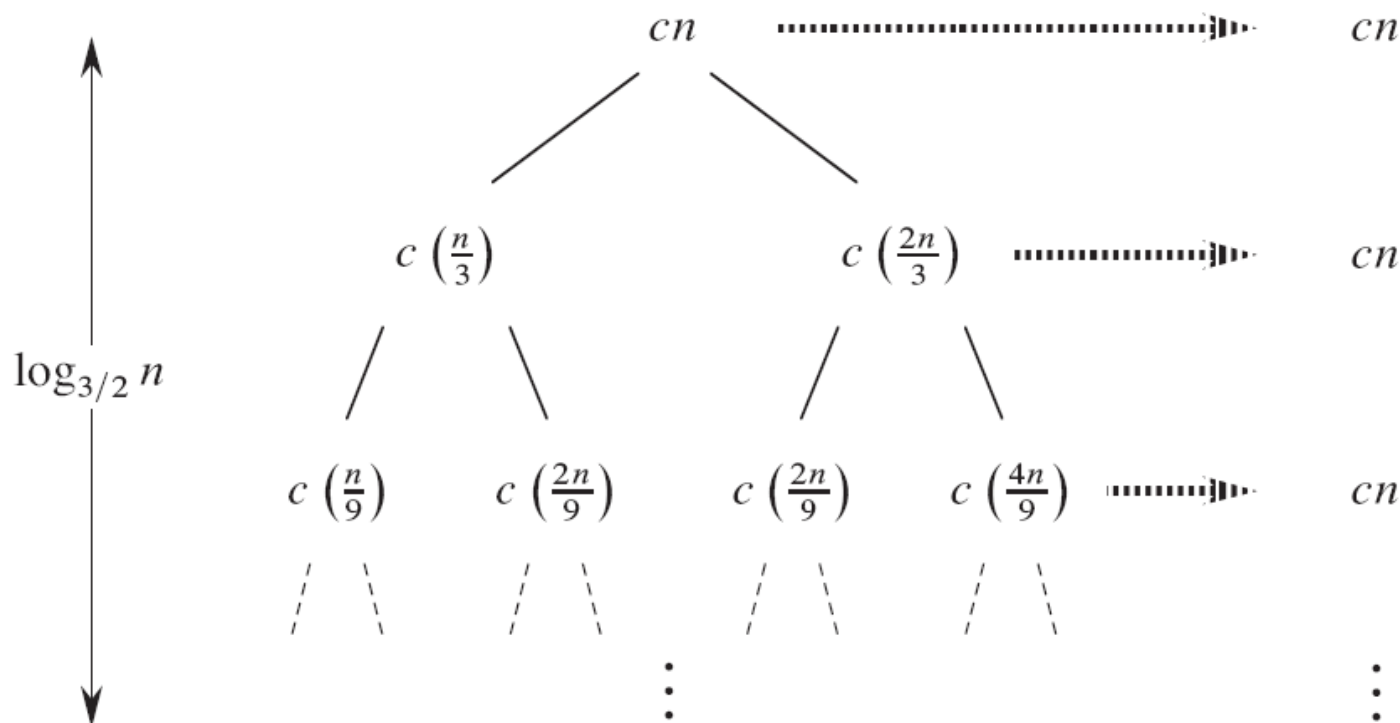
求解 $T(n) = T(n/3) + T\left(\frac{2n}{3}\right) + cn$



求解 $T(n) = T(n/3) + T\left(\frac{2n}{3}\right) + cn$

叶子的数量实际为 $O(n)$

通过数学归纳法可得 $T(n) = O(n \log n)$



2.3.3 Master method

目的：求解 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$ 型方程， $a \geq 1, b > 1$ 是常数， $f(n)$ 是正函数

一般的分治递归：把问题分成一些更小 (或许有重叠) 的子问题，递归地求解这些子问题，然后用所得到的子问题的解去求解原始问题。

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$ ：将一个大小为 n 的问题分成大小为 n/b 的 a 个子问题，递归地求解这些子问题，然后用所得到的子问题的解以 $f(n)$ 的代价求解原始问题。

Master 定理

定理2.4.1 设 $a \geq 1$, $b > 1$ 为常数, $f(n)$ 是一个函数, $T(n)$ 是定义在非负整数上的递归式 $T(n) = aT(n/b) + f(n)$. 这里 n/b 可以解释为 $\lfloor n/b \rfloor$ 或者 $\lceil n/b \rceil$. 则 $T(n)$ 的渐进界为:

(1). 若对于某个常数 $\varepsilon > 0$, 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 $T(n) = \theta(n^{\log_b a} \lg n)$.

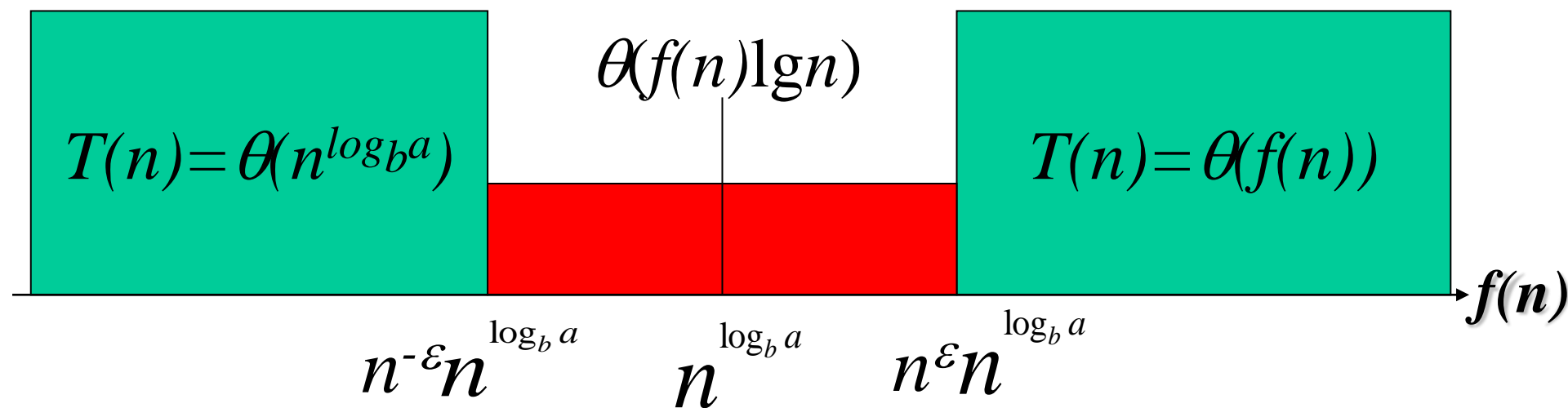
(3). 若对于某个常数 $\varepsilon > 0$, 有 $f(n) = \Omega(n^{\log_b a + \varepsilon})$, 而且对于某个常数 $c < 1$ 和足够大的 n 有 $af(n/b) \leq cf(n)$, 则 $T(n) = \theta(f(n))$.

*直观地：我们用 $f(n)$ 与 $n^{\log_b a}$ 比较

(1). 若 $n^{\log_b a}$ 大，则 $T(n) = \theta\left(n^{\log_b a}\right)$

(2). 若 $f(n)$ 大，则 $T(n) = \theta(f(n))$

(3). 若 $f(n)$ 与 $n^{\log_b a}$ 同阶，则 $T(n) = \theta\left(n^{\log_b a} \lg n\right) = \theta(f(n) \lg n)$.



对于红色部分，Master定理无能为力

更进一步:

- (1). 在第一种情况, $f(n)$ 不仅小于 $n^{\log_b a}$, 必须多项式地小于, 即对于一个常数 $\varepsilon > 0$, $f(n) = O(\frac{n^{\log_b a}}{n^\varepsilon})$.
- (2). 在第三种情况, $f(n)$ 不仅大于 $n^{\log_b a}$, 必须多项式地大于, 即对一个常数 $\varepsilon > 0$, $f(n) = \Omega(n^{\log_b a} \cdot n^\varepsilon)$.

Master定理的使用

例 1. 求解 $T(n) = 9T\left(\frac{n}{3}\right) + n$.

解: $a = 9, b = 3, f(n) = n, n^{\log_b a} = \theta(n^2)$

$$\because f(n) = n = O\left(n^{\log_b a - \varepsilon}\right), \quad \varepsilon = 1$$

$$\therefore T(n) = \theta\left(n^{\log_b a}\right) = \theta(n^2)$$

例 2. 求解 $T(n) = T\left(\frac{2n}{3}\right) + 1$.

解: $a = 1, b = \left(\frac{3}{2}\right), n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1,$

$$f(n) = 1 = \theta(1) = \theta\left(n^{\log_b a}\right), \quad T(n) = \theta\left(n^{\log_b a} \lg n\right) = \theta(\lg n)$$

例 3. 求解 $T(n) = 3T\left(\frac{n}{4}\right) + n \lg n$

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

$$(1) f(n) = n \lg n \geq n = n^{\log_b a + \varepsilon}, \varepsilon \approx 0.2$$

$$(2) \text{ 对所有 } n, af\left(\frac{n}{b}\right) = 3 \times \frac{n}{4} \lg \frac{n}{4} = \frac{3}{4} n \lg \frac{n}{4} \leq \frac{3}{4} n \lg n = cf(n), c = \frac{3}{4}.$$

于是, $T(n) = \theta(f(n)) = \theta(n \lg n)$

例 4. 求解 $T(n) = 2^n T\left(\frac{n}{2}\right) + n^n$

解: $a=2^n$, 非常数项, 不满足master定理条件,
故master定理不适用。

例 5. 求解 $T(n) = 0.5T(\frac{n}{2}) + \frac{1}{n}$

解： $a < 1$ ，不满足master定理条件，
故master定理不适用。

例 6. 求解 $T(n) = 64T(n/8) - n^2 \log n$

解： $f(n)$ 非正函数，不满足master定理条件，
故master定理不适用。

扩展master定理

定理2.4.2 设 $a \geq 1$, $b \geq 1$ 为常数, $f(n)$ 是一个函数, $T(n)$ 是定义在非负整数上的递归式 $T(n) = aT(n/b) + f(n)$. 这里 n/b 可以解释为 $\lfloor n/b \rfloor$ 或者 $\lceil n/b \rceil$. 则 $T(n)$ 的渐进界为:

(1). 若对于某个常数 $\varepsilon > 0$, 有 $f(n) = O(n^{\log_b a - \varepsilon})$, 则 $T(n) = \theta(n^{\log_b a})$.

(2). 若对于某个非负整数 $k \geq 0$ 有 $f(n) = \theta(n^{\log_b a} (\lg n)^k)$, 则 $T(n) = \theta(n^{\log_b a} (\lg n)^{k+1})$.

(3). 若对于某个常数 $\varepsilon > 0$, 有 $f(n) = \Omega(n^{\log_b a + \varepsilon})$, 而且对于某个常数 $c < 1$ 和足够大的 n 有 $af(n/b) \leq cf(n)$, 则 $T(n) = \theta(f(n))$.

(2). 若对于某个非负整数 $k \geq 0$ 有 $f(n) = \theta(n^{\log_b a} (\lg n)^k)$, 则 $T(n) = \theta(n^{\log_b a} (\lg n)^{k+1})$.

例 7. 求解 $T(n) = 2T(n/2) + n \lg n$.

解: $f(n) = n \lg n$

$a = 2, b = 2$, 根据原始的 master 定理, $n^{\log_b a} = n$
 $\lg n$ 和 n^ϵ 的大小关系: 对于任意 $\epsilon > 0$, $\lg n \in o(n^\epsilon)$

显然无法找到大于零的 ϵ 使得:

$n \lg n = O(n^{1-\epsilon})$ 或者 $n \lg n = \Omega(n^{1+\epsilon})$ 成立

$a = 2, b = 2, k = 1$, 故: $n^{\log_b a} \log^k n = n \lg n = f(n)$

$$T(n) = \theta(n \lg^2 n)$$

- 计算复杂性函数的阶
 - 同阶、低阶、高阶、严格低阶、严格高阶
 - 算法的复杂性与问题的复杂性
- 递归方程
 - 定义
 - 求解方法：替换法、迭代展开、master方法等