



算法设计与分析—入门篇

第二讲 算法分析的数学基础

哈尔滨工业大学
王宏志

wangzh@hit.edu.cn

<http://homepage.hit.edu.cn/pages/wang/>



本讲内容

- 2.1 计算复杂性函数的阶
- 2.2 和式的估计与界限
- 2.3 递归方程

增长的阶

- 如何描述算法的效率？
 - 增长率
- 忽略低阶项，保留最高阶项
- 忽略常系数
- 利用 $\Theta(n^2)$ 表示插入排序的最坏运行时间
- $\Theta(1)$, $\Theta(\lg n)$, $\Theta(\sqrt{n})$, $\Theta(n)$, $\Theta(n \lg n)$, $\Theta(n^2)$, $\Theta(n^3)$, $\Theta(2^n)$, $\Theta(n!)$

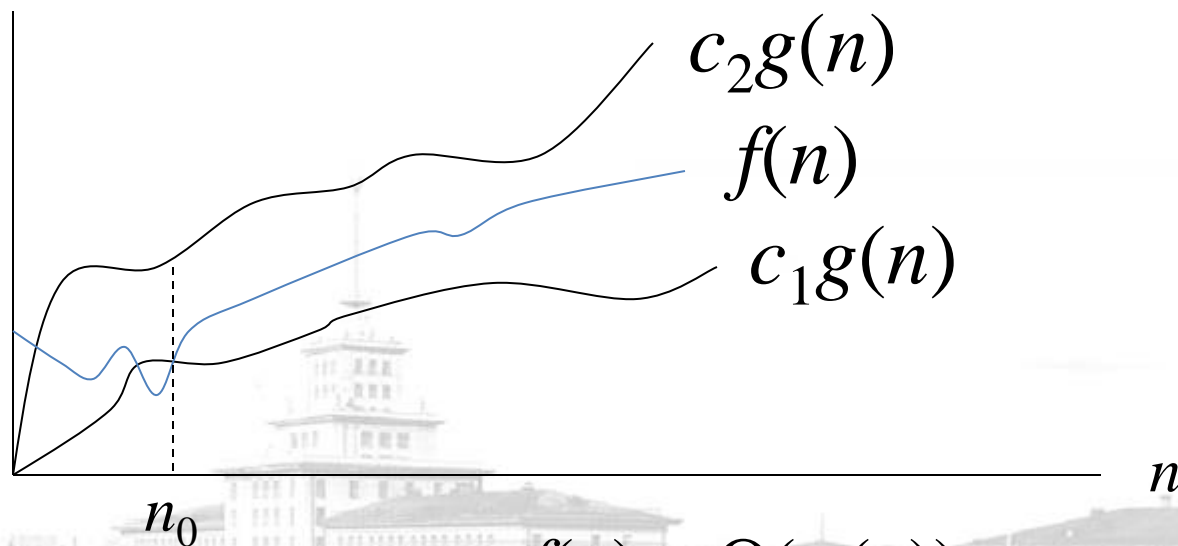
增长函数

- 渐进效率:
 - 输入规模非常大
 - 忽略低阶项和常系数
 - 只考虑最高阶(增长的阶)
- 典型的增长阶:
 - $\Theta(1)$, $\Theta(\lg n)$, $\Theta(\sqrt{n})$, $\Theta(n)$, $\Theta(n \lg n)$, $\Theta(n^2)$, $\Theta(n^3)$,
 $\Theta(2^n)$, $\Theta(n!)$
- 增长的记号: O , Θ , Ω , o , ω .

同阶函数集合

$\Theta(g(n)) = \{f(n) \mid \exists c_1, c_2 > 0, n_0, \forall n > n_0, c_1 g(n) \leq f(n) \leq c_2 g(n)\}$
称为与 $g(n)$ 同阶的函数集合。

- 如果 $f(n) \in \Theta(g(n))$, $g(n)$ 与 $f(n)$ 同阶
- $f(n) \in \Theta(g(n))$, 记作 $f(n) = \Theta(g(n))$



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

$\Theta(g(n))$ 函数的例子

- 证明 $1/2n^2 - 3n = \Theta(n^2)$.
 - $c_1 n^2 \leq 1/2n^2 - 3n \leq c_2 n^2$
 - $c_1 \leq 1/2 - 3/n \leq c_2$
 - 对于任意 $n \geq 1$, $c_2 \geq 1/2$, 且对于任意 $n \geq 7$, $c_1 \leq 1/14$
 - 因此 $c_1 = 1/14$, $c_2 = 1/2$, $n_0 = 7$.
- 证明 $6n^3 \neq \Theta(n^2)$.
 - 如果存在 $c_1, c_2 > 0$, n_0 使得当 $n \geq n_0$ 时, $c_1 n^2 \leq 6n^3 \leq c_2 n^2$ 。
 - 当 $n > c_2/6$ 时, $n \leq c_2/6$, 矛盾。

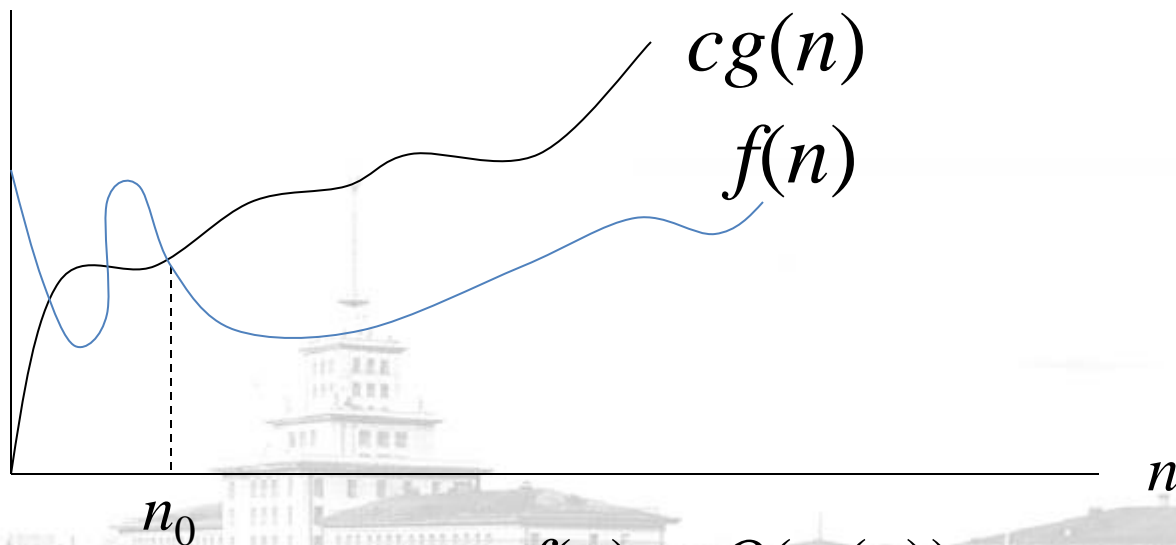
$\Theta(g(n))$ 函数的例子

- 通常 $f(n) = an^2 + bn + c = \Theta(n^2)$, 其中 a, b, c 是常数且 $a > 0$.
- $p(n) = \sum_{i=0}^d a_i n^i$, 其中 a_i 是常数且 $a_d > 0$.
 - $p(n) = \Theta(n^d)$.
- $\Theta(n^0)$ 或者 $\Theta(1)$, 常数时间复杂性.



低阶函数集合

- 对于给定的函数 $g(n)$,
 - $O(g(n)) = \{f(n) : \text{存在正常数 } c \text{ 和 } n_0 \text{ 满足对于所有 } n \geq n_0, \underline{0 \leq f(n) \leq cg(n)}\}$
 - 记作 $f(n) \in O(g(n))$, 或简记为 $f(n) = O(g(n))$.



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

$\Theta(g(n))$ 和 $O(g(n))$ 的关系

- $f(n) = \Theta(g(n)) \Rightarrow f(n) = O(g(n))$
- Θ 标记强于 O 标记.
- $\Theta(g(n)) \subseteq O(g(n))$
- $an^2+bn+c = \Theta(n^2)$, 且 $=O(n^2)$
- $an+b = O(n^2)$. 为什么?
- $n = O(n^2)$!!!
- O 标记, 表示渐进上界
- Θ 标记, 表示渐进紧界
- 一些讨论.

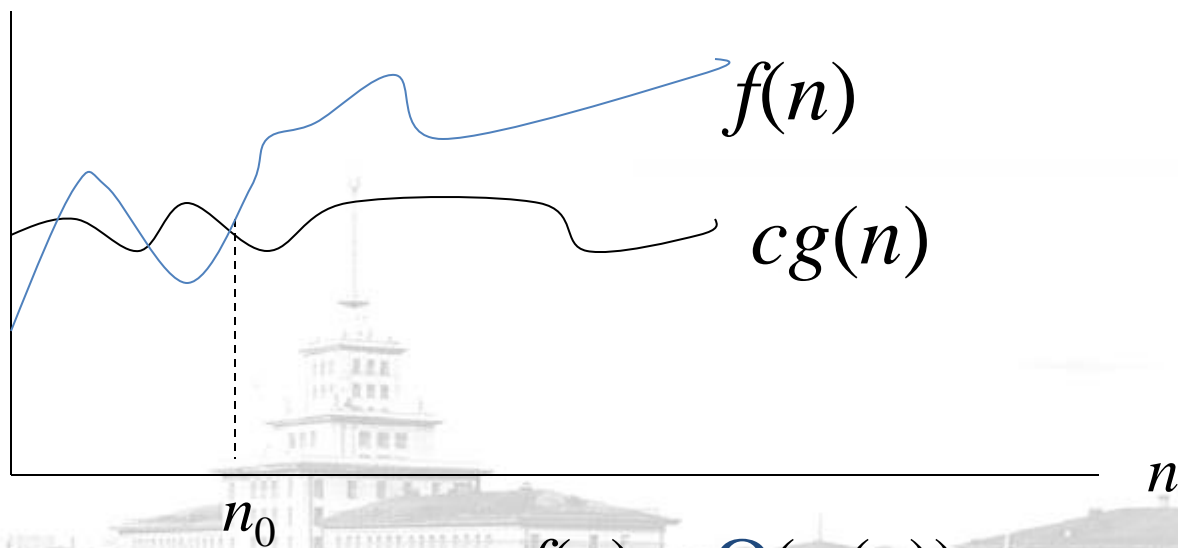
如果 $f(n)=O(n^k)$, 则称 $f(n)$ 是多项式界限的。

的输入, 即使对于已经排序的输入也成立, 因为 $O(n) \in O(n^4)$.

- 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, $\Theta(n) \neq \Theta(n^2)$.

高阶函数集合

- 对于给定的函数 $g(n)$,
 - $\Omega(g(n)) = \{f(n) : \text{存在正常数 } c \text{ 和 } n_0, \text{ 使得对于所有 } n \geq n_0, 0 \leq cg(n) < f(n)\}$
 - 记作 $f(n) \in \Omega(\underline{g(n)})$, 或简记为 $f(n) = \Omega(g(n))$.



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

O, Θ, Ω 标记的关系

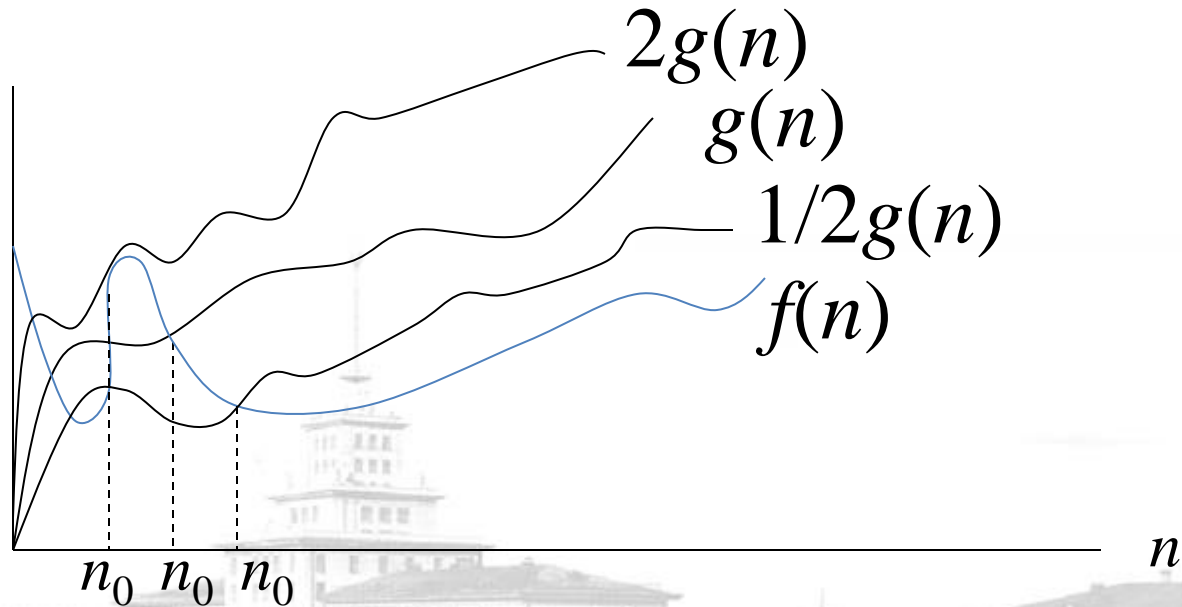
- 对于 $f(n)$ 和 $g(n)$, $f(n) = \Theta(g(n))$ 当且仅当 $f(n) = O(g(n))$ 且 $f(n) = \Omega(g(n))$.
- O : 渐进上界
- Θ : 渐进紧界
- Ω : 渐进下界

关于 Ω 标记

- 用来描述运行时间的最好情况
- 对所有输入都正确
- 比如，对于插入排序
 - 最好运行时间是 $\Omega(n)$,
 - 或者说，运行时间是 $\Omega(n)$.
 - 最坏运行时间是 $\Omega(n^2)$
 - 但是说运行时间是 $\Omega(n^2)$ 则有误
- 可以用来描述问题
 - 排序问题的时间复杂性是 $\Omega(n)$

严格低阶函数

- 给定一个函数 $g(n)$,
 - $o(g(n)) = \{f(n) : \text{对于任意正常数 } c, \text{ 存在一个正数 } n_0, \text{ 从而对所有 } n \geq n_0, \text{ 满足 } 0 \leq f(n) < cg(n)\}$
 - 记作 $f(n) \in o(g(n))$, 或者简写为 $f(n) = o(g(n))$.



$$f(n) = o(g(n))$$

例 1. 证明 $2n = o(n^2)$

证. 对 $\forall c > 0$, 欲 $2n < cn^2$, 必 $2 < cn$, 即 $\frac{2}{c} < n$ 。所以, 当 $n_0 = \frac{2}{c}$ 时,

$2n < cn^2$ 对 $\forall c > 0$, $n \geq n_0$ 。

例 2. 证明 $2n^2 \neq o(n^2)$

证. 当 $c=1>0$ 时, 对于任何 n_0 , 当 $n \geq n_0$, $2n^2 < cn^2$ 都不成立



关于o标记

- 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.

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

证. 由于 $f(n) = o(g(n))$, 对任意 $\varepsilon > 0$, 存在 n_0 , 当 $n \geq n_0$ 时,
 $0 \leq f(n) < \varepsilon g(n)$,

即 $0 \leq \frac{f(n)}{g(n)} < \varepsilon$. 于是, $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$.

严格高阶函数集合

- 对于给定函数 $g(n)$,
 - $\omega(g(n)) = \{f(n) : \text{对于任意正常数 } c, \text{ 存在正数 } n_0 \text{ 对于 } n \geq n_0, 0 \leq cg(n) < f(n)\}$
 - 记作 $f(n) \in \omega(g(n))$, 或者简记为 $f(n) = \omega(g(n))$.
- ω 标记, 类似 o 标记, 表示不紧的下界.
 - $n^2/2 = \omega(n)$, 但 $n^2/2 \neq \omega(n^2)$
- $f(n) = \omega(g(n))$ 当且仅当 $g(n) = o(f(n))$.
- $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$

渐进符号的性质

- 传递性: 所有五个标记
 - $f(n) = \Theta(g(n))$ 且 $g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$
- 自反性: O, Θ, Ω .
 - $f(n) = \Theta(f(n))$
- 对称性: Θ
 - $f(n) = \Theta(g(n))$ 当且仅当 $g(n) = \Theta(f(n))$
- 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 $g(n) = \omega(f(n))$.

注意



* 并非所有函数都是可比的，即对于函数 $f(n)$ 和 $g(n)$ ，可能 $f(n) \neq O(g(n))$, $f(n) \neq \Omega(g(n))$ 。例如， n 和 $n^{1+\sin n}$ 。



本讲内容

- 2.1 计算复杂性函数的阶
- 2.2 和式的估计与界限
- 2.3 递归方程

为什么需要和式的估计与界限

```
FOR  $l=2$  TO  $n$   
  FOR  $i=1$  TO  $n-l+1$  DO  
     $j=i+l-1$ ;  
     $m[i,j]=\infty$ ;  
    FOR  $k \leftarrow i$  To  $j-1$  DO  
       $q=m[i,k]+m[k+1,j]+p_{i-1}p_kp_j$   
      IF  $q < m[i,j]$  THEN  $m[i,j]=q$ ;
```



和式的估计

1. 线性和

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



2. 级数

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

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

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

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

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

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

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

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

3. 和的界限

例 1. 证明 $\sum_{k=0}^n 3^k = O(3^n)$

证 证明对于 $c \geq 3/2$, 存在一个 n_0 , 当 $n \geq n_0$ 时 $\sum_{k=0}^n 3^k \leq c3^n$.

当 $n=0$ 时, $\sum_{k=0}^n 3^k = 1 \leq c = c3^n$.

设 $n \leq m$ 时成立. 令 $n = m+1$, 则

$$\sum_{k=0}^{m+1} 3^k = \sum_{k=0}^m 3^k + 3^{m+1} \leq c3^m + 3^{m+1} = c3^{m+1} \left(\frac{1}{3} + \frac{1}{c} \right) \leq c3^{m+1}.$$



例 2. 错误证明: $\sum_{k=1}^n k = \theta(n)$

证. $n=1$ 时, $\sum_{k=1}^1 k = 1 = O(1)$, $\sum_{k=1}^{n+1} k = \sum_{k=1}^n k + (n+1) = O(n) + (n+1) = O(n)$.

*错在 $O(n)$ 的常数 c 随 n 的增长而变化, 不是常数。

*要证明 $\sum_{k=1}^n k = O(n)$ 需要证明: 对某个 $c > 0$, $\sum_{k=1}^n k = c(n)$.



直接求和的界限

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

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

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

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

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

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

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

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

解. 使用例 3 的方法. $\frac{k+1}{3^{k+1}} / \frac{k}{3^k} = \frac{1}{3} \cdot \frac{k+1}{k} = \frac{1}{3} \left(1 + \frac{1}{k}\right) \leq \frac{2}{3} = r$. 于是

$$\sum_{k=1}^{\infty} \frac{k}{3^k} \leq \sum_{k=1}^{\infty} a_1 r^k = \sum_{k=1}^{\infty} \frac{1}{3} \left(\frac{2}{3}\right)^k = \frac{1}{3} \cdot \frac{1}{1 - \frac{2}{3}} = 1.$$

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, $\frac{(k+1)^2/2^{k+1}}{k^2/2^k} = \frac{(k+1)^2}{2k^2} \leq \frac{8}{9}$

于是 $\sum_{k=0}^{\infty} \frac{k^2}{2^k} = \sum_{k=0}^2 \frac{k^2}{2^k} + \sum_{k=3}^{\infty} \frac{k^2}{2^k} \leq \theta(1) + \sum_{k=3}^{\infty} \frac{9}{8} \cdot \left(\frac{8}{9}\right)^k = \theta(1).$

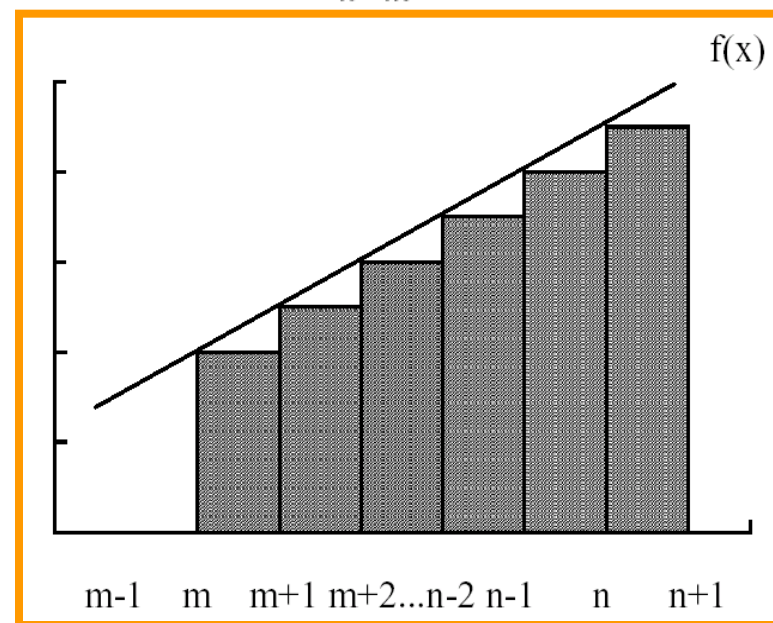
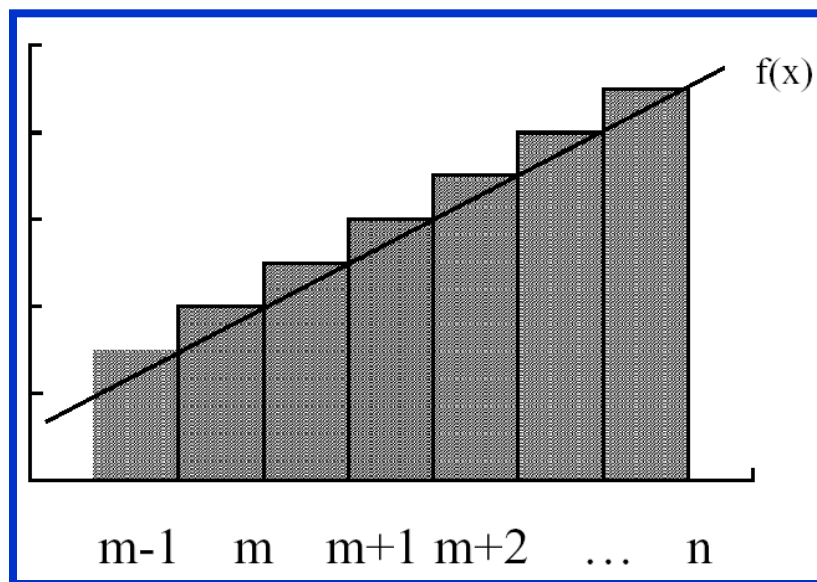


例 7. 求 $H_n = \sum_{k=1}^n \frac{1}{k}$ 的上界

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



例 8. 如果 $f(k)$ 单调递增, 则 $\int_{m-1}^n f(x)dx \leq \sum_{k=m}^n f(k) \leq \int_m^{n+1} f(x)dx$.



$$\sum_{k=m}^n f(k) \geq \sum_{k=m-1}^{n-1} f(k)\Delta x \geq \int_{m-1}^n f(x)dx, \quad f(m-1) < f(n), \quad \Delta x = 1$$

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

例 9. 当 $f(x)$ 单调递减时, $\int_m^{n+1} f(x)dx \leq \sum_{k=m}^n f(k) \leq \int_{m-1}^n f(x)dx$.

例 10. $\sum_{k=1}^n \frac{1}{k} \geq \int_1^{n+1} \frac{dx}{x} = \ln(n+1)$, $\sum_{k=2}^n \frac{1}{k} \leq \int_1^n \frac{dx}{x} = \ln n$.



本讲内容

- 2.1 计算复杂性函数的阶
- 2.2 和式的估计与界限
- 2.3 递归方程



递归方程

- 递归方程：递归方程是使用小的输入值来描述一个函数的方程或不等式。

- 递归方程例：**Merge-sort**排序算法的复杂性方程

$$T(n) = \Theta(1) \quad \text{if } n=1$$

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

$T(n)$ 的解是 $\Theta(n \log n)$

求解递归方程的三个主要方法

- 替换方法：
 - 首先猜想,
 - 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定理方法：
 - 求解型为 $T(n)=aT(n/b)+f(n)$ 的递归方程

替换方法

替换方法 I：联想已知的 $T(n)$

例1. 求解 $2T(n/2 + 17) + n$

解：猜测： $T(n) = 2T\left(\frac{n}{2} + 17\right) + n$ 与 $T(n) = 2T\left(\frac{n}{2}\right) + n$ 只相差一个 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

$\frac{n}{2} + 17$ 与 $\frac{n}{2}$ 相差小. 我们可以猜 $T(n) = O(n \lg n)$.

证明：用数学归纳法



替换方法 I： 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 $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil \frac{n}{2} \rceil) + 1$

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

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

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 $T(n) \leq cn$ 。

--错!!

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

错在那里：过早地使用了 $O(n)$ 而陷入了陷阱应该在证明了 $T(n) \leq cn$ 才可用。从 $T(n) \leq cn + n$ 不可能得到 $T(n) \leq cn$ 因为对于任何 $c > 0$ ，我们都得不到 $cn + n \leq cn$ 。



变量替换方法:

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 $T(2^m) = \theta(m \lg m)$.

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



迭代方法

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。



$$\begin{aligned}
 \text{例 1. } T(n) &= n + 3T\left(\left\lfloor \frac{n}{4} \right\rfloor\right) \\
 &= n + 3\left(\left\lfloor \frac{n}{4} \right\rfloor + 3T\left(\left\lfloor \frac{n}{16} \right\rfloor\right)\right) \\
 &= n + 3\left(\left\lfloor \frac{n}{4} \right\rfloor + 3\left(\left\lfloor \frac{n}{16} \right\rfloor + 3T\left(\left\lfloor \frac{n}{64} \right\rfloor\right)\right)\right) \\
 &= n + 3\left\lfloor \frac{n}{4} \right\rfloor + 9\left\lfloor \frac{n}{16} \right\rfloor + 27T\left(\left\lfloor \frac{n}{64} \right\rfloor\right) \\
 &= n + 3\left\lfloor \frac{n}{4} \right\rfloor + 3^2\left\lfloor \frac{n}{4^2} \right\rfloor + 3^3\left\lfloor \frac{n}{4^3} \right\rfloor + \dots + 3^i T\left(\left\lfloor \frac{n}{4^i} \right\rfloor\right)
 \end{aligned}$$

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

$$\begin{aligned}
 &= n + 3\left\lfloor \frac{n}{4} \right\rfloor + 3^2\left\lfloor \frac{n}{4^2} \right\rfloor + 3^3\left\lfloor \frac{n}{4^3} \right\rfloor + \dots + 3^{\log_4 n} T(1) \\
 &\leq \sum_{i=0}^{\log_4 n} 3^i \frac{n}{4^i} + O(n) \leq n \sum_{i=0}^{\infty} \left(\frac{3}{4}\right)^i = n \times \frac{1}{1 - \frac{3}{4}} = 4n = O(n)
 \end{aligned}$$

Master定理方法

目的：求解 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$ 型方程， $a \geq 1, b > 0$ 是常数，
 $f(n)$ 是正函数

方法：记住三种情况，则不用笔纸即可求解上述方程。



Master 定理

Master 定理 设 $a \geq 1$ 和 $b > 1$ 是常数, $f(n)$ 是一个函数, $T(n)$ 是定义在非负整数集上的函数 $T(n) = aT\left(\frac{n}{b}\right) + f(n)$. $T(n)$ 可以如下求解:

(1). 若 $f(n) = \underline{O(n^{\log_b a - \varepsilon})}$, $\varepsilon > 0$ 是常数, 则 $T(n) = \theta(n^{\log_b a})$.

(2). 若 $f(n) = \theta(n^{\log_b a})$, 则 $T(n) = \theta(n^{\log_b a} \lg n)$.

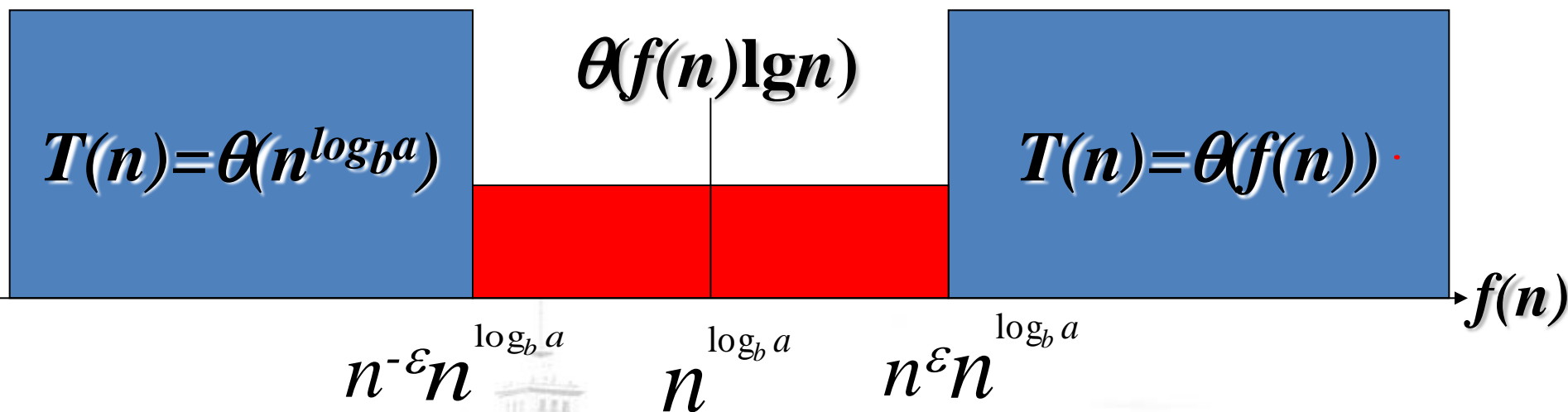
(3). 若 $f(n) = \Omega(n^{\log_b a + \varepsilon})$, $\varepsilon > 0$ 是常数, 且对于所有充分大的 n
 $af\left(\frac{n}{b}\right) \leq cf(n)$, $c > 1$ 是常数, 则 $T(n) = \theta(f(n))$.

*直观地：我们用 $f(n)$ 与 $n^{\log_b a}$ 比较

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

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

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



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 $\varepsilon > 0$, $f(n) = O\left(\frac{n^{\log_b a}}{n^\varepsilon}\right)$.
- (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 $T(n) = 9T(n/3) + n$.

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

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

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

例 2. 求解 $T(n) = T(2n/3) + 1$.

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

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

Master定理的使用（续）

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 $T(n) = 2T\left(\frac{n}{2}\right) + n \lg n$.

解: $a = 2, b = 2, f(n) = n \lg n, n^{\log_b a} = n$. $f(n) = n \lg n$ 大于 $n^{\log_b a} = n$, 但不是多项式地大于, Master 定理不适用于该 $T(n)$.