算法设计与分析
第二章 算法分析的数学基础

张开旗
海量数据计算研究中心
计算学部

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

函数的阶

- 如何描述算法的效率？
  - 时间/空间复杂性
  - 是原子操作数
  - 是关于输入大小的函数
- 计算复杂性函数的阶
  - 函数的主导项（增长率、增长的阶）

函数的阶

- 渐进效率:
  - 输入规模非常大
  - 忽略低阶项和常系数
  - 只考虑最高阶项(主导项)
- 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.

函数的阶

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

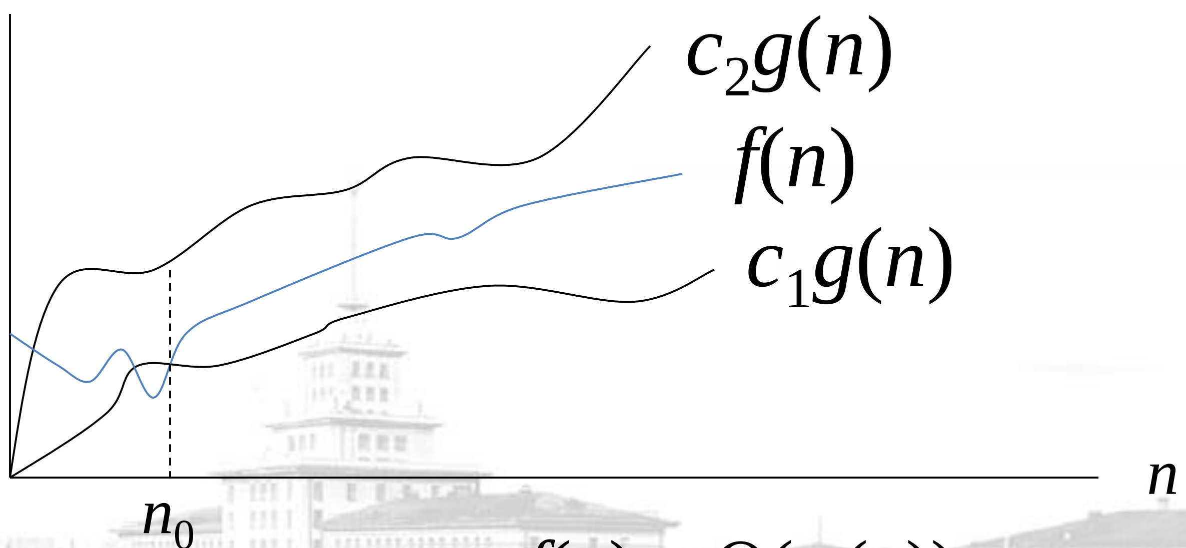
系数a也相对来说意义不大

即：函数$T(n)$的阶为n^2，即$T(n)=\Theta(n^2)$。

同阶函数集合

$\Theta(g(n))=\{f(n) \mid \exists c_1, c_2>0, n_0, \forall n>n_0, c_1g(n)\leq f(n)\leq c_2g(n)\}$ 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

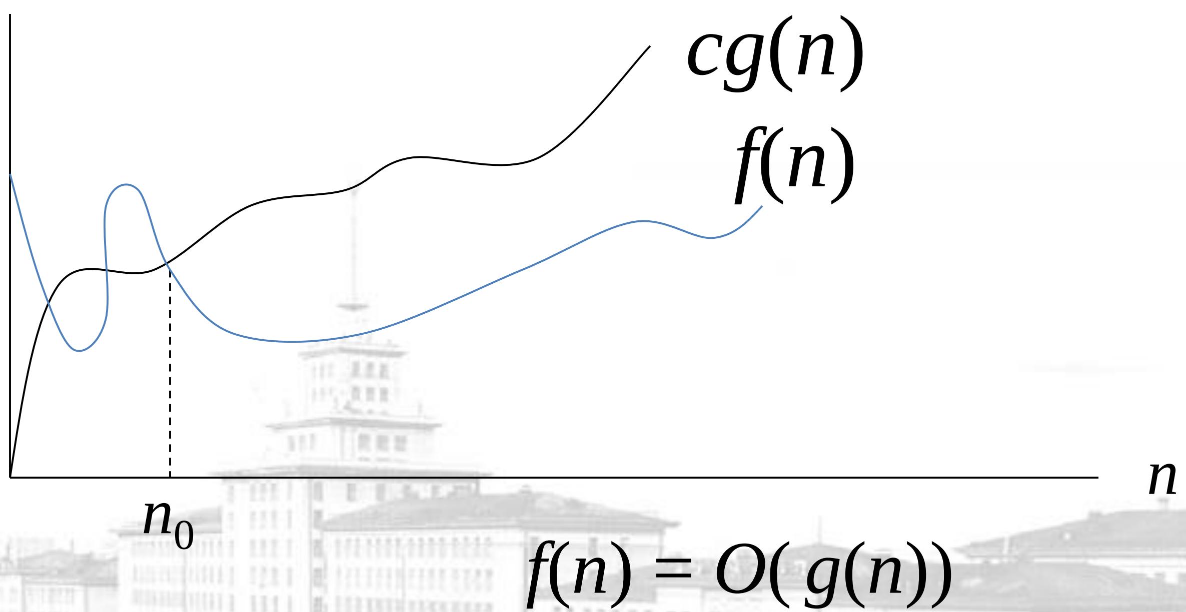
- 证明 $1/2n^2 - 3n = \Theta(n^2)$.
  - $c_1 n^2 \le 1/2n^2 - 3n \le c_2 n^2$
  - $c_1 \le 1/2 - 3/n \le c_2$
  - 对于任意$n \ge 1$, $c_2 \ge ½$, 且对于任意$n \ge 7$， $c_1 \le 1/14$
  - 因此 $c_1 = 1/14$, $c_2 = ½$, $n_0 = 7$.
- 证明 $6n^3 \ne \Theta(n^2)$.
  - 如果存在c_1、$c_2>0$，n_0使得当$n \ge n_0$时， $c_1 n^2 \le 6n^3 \le c_2 n^2$ 。
  - 当$n > c_2/6$时， $n \le c_2/6$， 矛盾。

$\Theta(g(n))$ 函数

- 同阶函数集合$f(n)=\Theta(g(n))$中，所有的$f(n)$函数拥有相同的阶（主导项）。

低阶函数集合

- 对于给定的函数$g(n)$,
  - $O(g(n))=\{f(n) \mid \exists c>0,\ n_0,\ \forall n>n_0,\ 0\le f(n) \le cg(n)\}$
  - 记作$f(n) \in O(g(n))$, 或简记为$f(n) = O(g(n))$.



$\Theta(g(n))$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$=O(n^2)$
- $an+b = O(n^2)$， 但$an+b \neq \Theta(n^2)$
- $n = O(n^2)$!!!
- O标记, 表示渐进上界
- Θ标记, 表示渐进紧界

如果$f(n)=O(n^k)$, 则称$f(n)$是多项式界限的。

时间复杂性实例

例1：计算n个数的平均值

	代价	次数
S=0;	c_1	1
for i=1 to n		
输入a值;	c_2	n
S=S+a;	c_3	n
S=S/n;	c_4	1
输出S值;	c_5	1

- 算法执行时间的计算（时间复杂性）
  - 所有原子操作执行时间的和
  - 每个原子操作假设为常数时间

$$T(n)= n(c_2+c_3) +c_1+c_4+c_5$$

- 度量
  - 原子操作数
  - 输入大小(n)的函数

$$T(n)= 2n + 3$$

用函数的阶来表示

$$T(n)= \Theta(n)$$

$$T(n)= O(n)$$

方法：关注原子操作数最多的项（最高阶项）

时间复杂性实例

例2：插入排序算法

- **最优代价**: 有序的数组
  - $t_j=1$
  - $T(n) = n\text{-}1 = \Theta(n)$
- **最坏代价**: 逆序数组
  - $t_j=j\text{-}1$,
  - $T(n) = \sum_{j=2}^{n} t_j = \sum_{j=2}^{n} (j\text{-}1) = n(n\text{-}1)/2 = \Theta(n^2)$
- **平均代价**:
  - 当读入第k个数x_k时，平均需要的比较次数：$\dfrac{k-1}{2}$
  - 总的平均比较次数为：$\sum_{k=2}^{n}(\dfrac{k-1}{2}) = \dfrac{n^2}{4} - \dfrac{n}{4} = \Theta(n^2)$

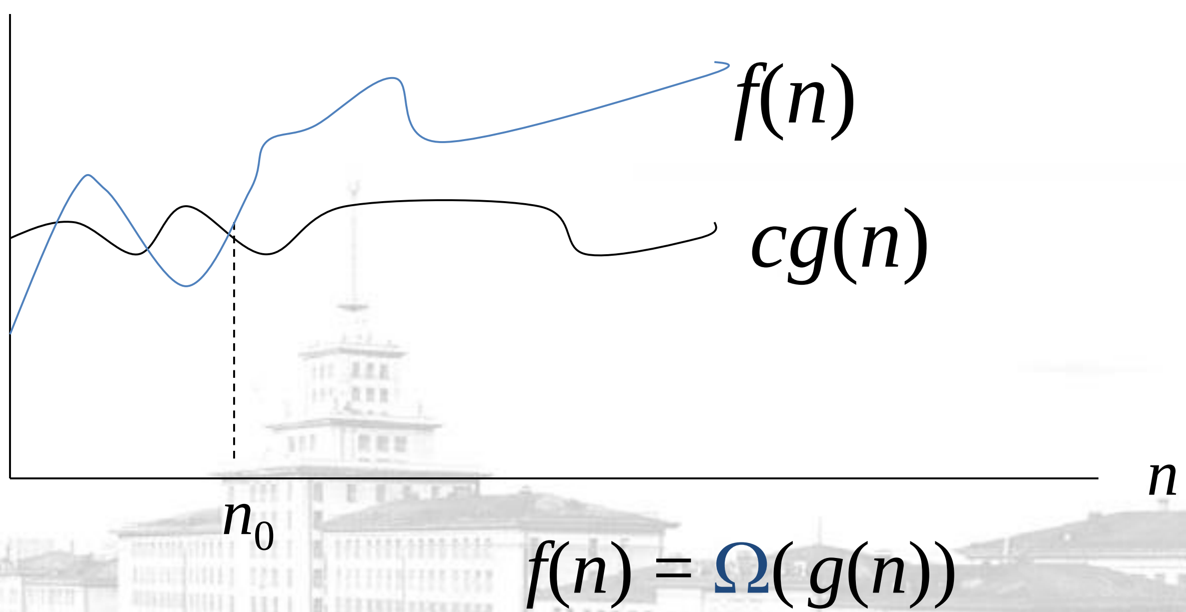
时间复杂性实例

例2：插入排序算法

- 我们可以说插入排序算法的运行时间是$O(n^2)$，因为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$\Theta(n) \in O(n^2)$.
- 也可以说插入排序的最坏运行时间$\Theta(n^2)$。
- 但不能说插入算法的运行时间是$\Theta(n^2)$，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$g(n)$,
  - $\Omega(g(n))=\{f(n) \mid \exists\, c>\mathbf{0}, n_0, \forall n\geq n_0,\ 0\leq cg(n)\leq f(n)\}$
  - 记作$f(n)\in\Omega(g(n))$, 或简记为$f(n)=\Omega(g(n))$.



关于Ω 标记

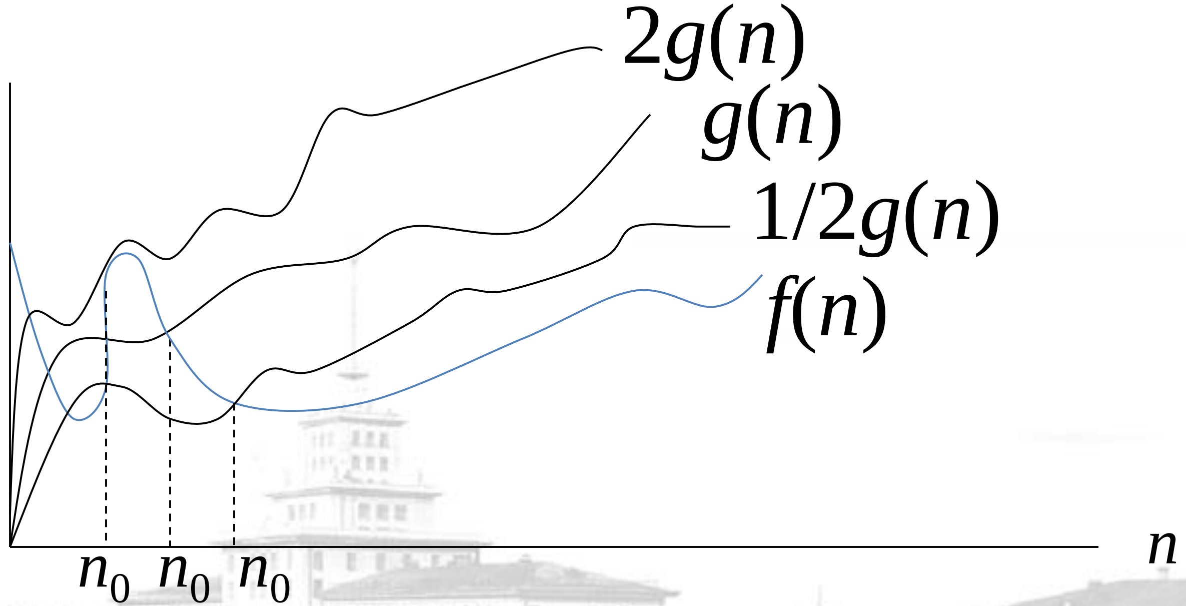
- 用来描述运行时间的最好情况
- 比如，对于插入排序
  - 最好运行时间是 $\Omega(n)$
  - 最坏运行时间是 $\Omega(n^2)$
  - 或者说，运行时间是$\Omega(n)$
  - 或者说，运行时间是$O(n^2)$
  - 但是说运行时间是$\Omega(n^2)$则有误
- 可以用来描述问题
  - 排序问题的时间复杂性是$\Omega(n)$

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: 低阶函数，渐进上界
- Θ : 同阶函数，渐进紧界
- Ω : 高阶函数，渐进下界

严格低阶函数

- 给定一个函数 $g(n)$,
  - $o(g(n))=\{f(n)$ | 对于任意正常数 c，存在一个正数 n_0，从而对所有 $n \geq n_0$, 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 $2n^2 \neq o(n^2)$　　　找反例

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
  - $2n = o(n^2)$, 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.

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

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

严格高阶函数集合

- 对于给定函数$g(n)$,
  - $\omega(g(n))=\{f(n) \mid$ 对于任意正常数c, 存在正数n_0 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$g(n)=o(f(n))$.
- $\lim\limits_{n \to \propto} \dfrac{f(n)}{g(n)} = \propto$

渐进符号的性质

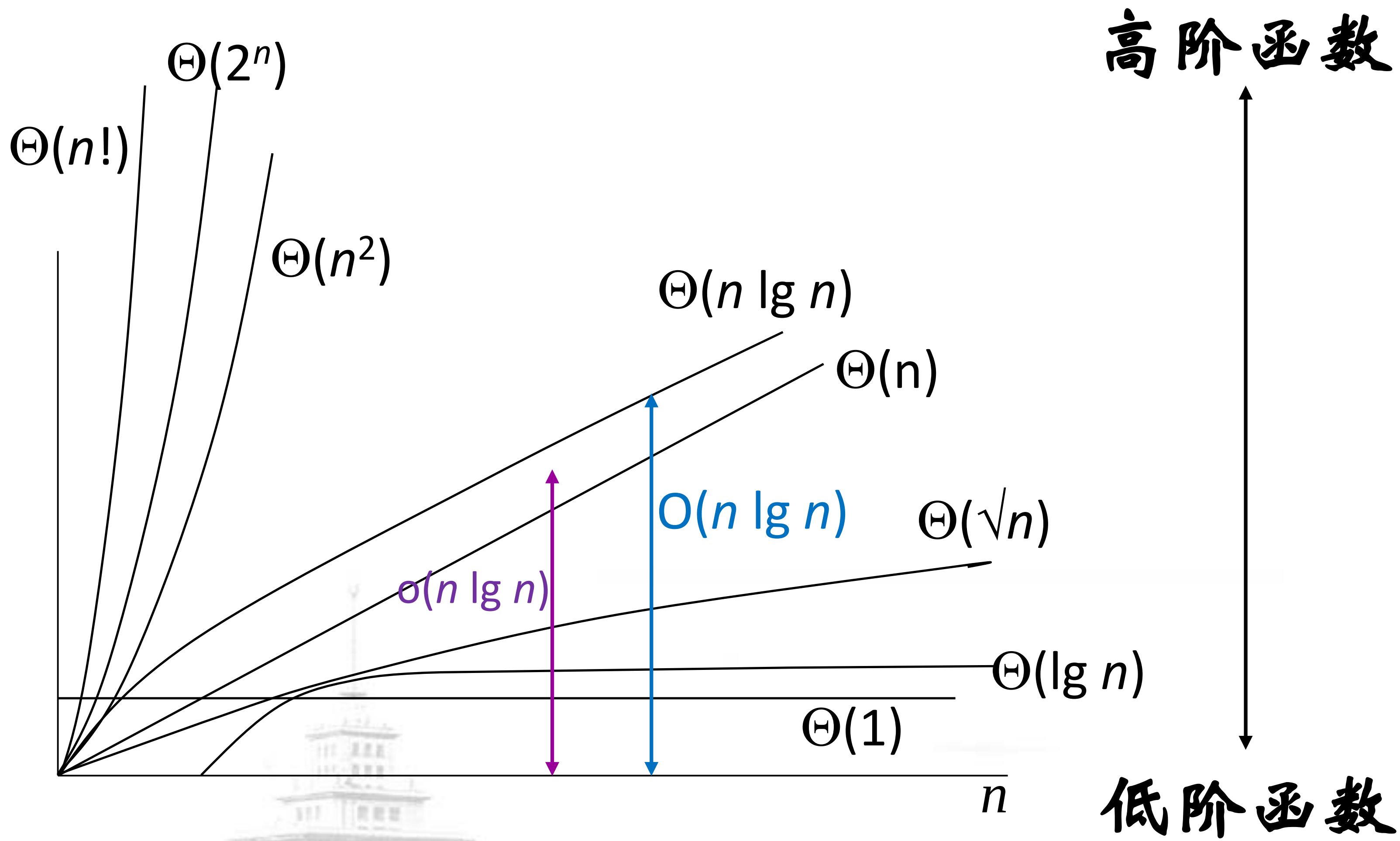
- 传递性: 所有五个标记
  - $f(n)=\Theta(g(n))$ 且$g(n)=\Theta(h(n))$ ➔ $f(n)=\Theta(h(n))$
- 自反性: O, Θ, Ω .
  - $f(n)=\Theta(f(n))$
- 对称性: Θ
  - $f(n)=\Theta(g(n))$ 当且仅当$g(n)=\Theta(f(n))$
- 反对称性:
  - $f(n) = O(g(n))$当且仅当 $g(n)=\Omega(f(n))$.
  - $f(n) = o(g(n))$当且仅当 $g(n)=\omega(f(n))$.

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 $n^{1+\sin n}$．

实例：高阶低阶函数



$o(f(n)) = O(f(n)) - \Theta(f(n))$

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

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

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

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

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

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

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

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

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

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

3. 和的界限

例 1．证明 $\sum_{k=0}^{n} 3^k = O(3^n)$　　　数学归纳法！

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

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

直接求和的界限

例1. $\sum_{k=1}^{n} k \le \sum_{k=1}^{n} n = n^2$ $\sum_{k=1}^{n} k$ =O(n²)

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

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

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

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

$a_3/a_2 \le r \Rightarrow a_3 \le a_2 r \le a_0 r^3$ ……

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

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

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

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

$$\sum_{k=1}^{\infty}\frac{k}{3^{k}}\leq\sum_{k=1}^{\infty}a_1r^{k}=\sum_{k=1}^{\infty}\frac{1}{3}\left(\frac{2}{3}\right)^{k}=\frac{1}{3}\cdot\frac{1}{1-\frac{2}{3}}=1\,.$$

例 5．用分裂和的方法求 $\sum_{k=1}^{n}k$ 的下界．

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

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

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

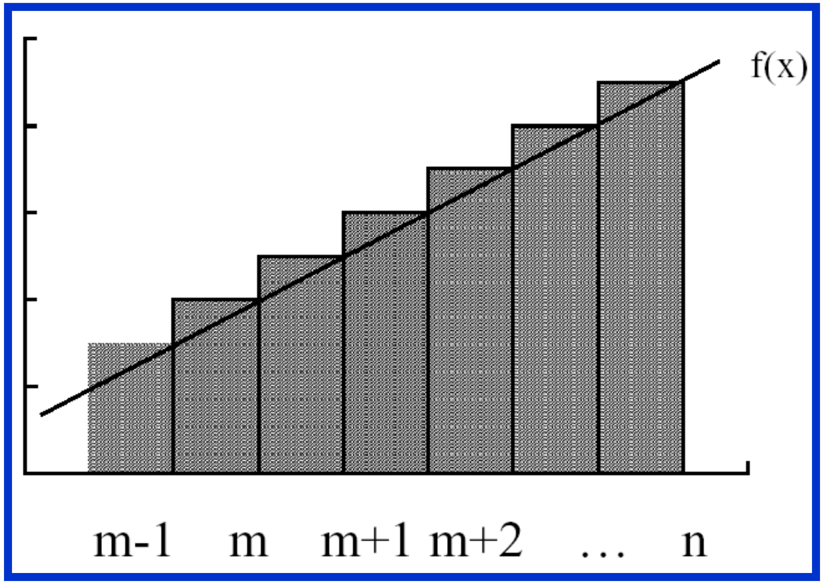
解：

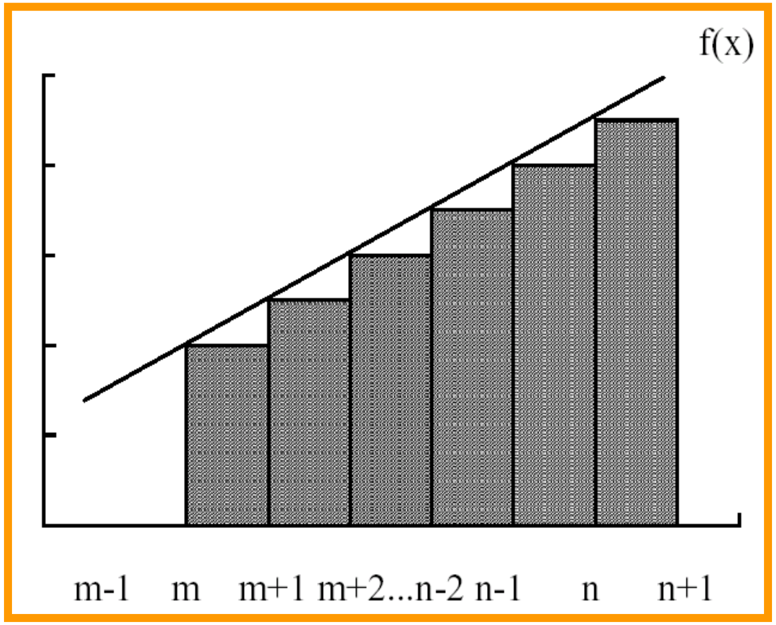
$$\sum_{k=1}^{n} \frac{1}{k} = \frac{1}{1} + \left(\frac{1}{2} + \frac{1}{3}\right) + \left(\frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7}\right)$$

$$+\left(\frac{1}{8} + \frac{1}{9} + \frac{1}{10} + \frac{1}{11} + \frac{1}{12} + \frac{1}{13} + \frac{1}{14} + \frac{1}{15}\right) + \ldots\ldots$$

$$\le \sum_{i=1}^{\lceil \log n \rceil} \sum_{j=0}^{2^i - 1} \frac{1}{2^i + j} \le \sum_{i=0}^{\lceil \log n \rceil} \sum_{j=0}^{2^i - 1} \frac{1}{2^i} \le \sum_{i=1}^{\lceil \log n \rceil} 1 \le \log n + 1 = \mathrm{O}(\log n)$$

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





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

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

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

例 10． $\sum_{k=1}^{n} \frac{1}{k} \ge \int_{1}^{n+1} \frac{dx}{x} = \ln(n+1)$ ， $\sum_{k=2}^{n} \frac{1}{k} \le \int_{1}^{n} \frac{dx}{x} = \ln n$.

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：递归方程是使用具有小输入值的相同方程来描述一个方程.（用自身来定义自身）
- 递归方程实例: **Merge-sort**排序算法复杂性方程

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

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

T(*n*)的解是Θ(*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(主定理)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 $T(n)= 2T(n/2 + 17) + n$

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

当 n 充分大时 $T\left(\dfrac{n}{2}+17\right)$ 与 $T\left(\dfrac{n}{2}\right)$ 的差别并不大，因为

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

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$\Omega(n\log n)$,
$0(n^2)$的下一个阶是 $0(n\log n)$。

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 $T(n)=\mathrm{O}(n)$ $\quad T(n)\le cn$

证：$T(n)\le c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1 = cn+1 \ne cn$

证不出 $T(n)=\mathrm{O}(cn)$

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

证：设当 $\le n-1$ 时成立

$$T(n)=T(\lfloor n/2 \rfloor)+T\left(\left\lceil \frac{n}{2} \right\rceil\right)+1\le c\left\lfloor \frac{n}{2} \right\rfloor - b + c\left\lceil \frac{n}{2} \right\rceil - b + 1$$

$$= cn-2b+1 = cn-b-b+1 \le cn-b$$ （只要 $b\ge 1$）。

避免陷阱

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

解： 猜 $T(n)=\mathrm{O}(n)$　$T(n)\le cn$

证： 用数学归纳法证明 $T(n)\le cn$ 。

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

--错！！

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

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

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

令 $S(m)=T\left(2^m\right)$ 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，$S(m)=\mathrm{O}(m\lg m)$，即 $T\left(2^m\right)=\theta(m\lg m)$.

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。

例 1. $T(n) = n + 3T\left(\left\lfloor n/4 \right\rfloor\right)$

$$= n + 3\left(\left\lfloor n/4 \right\rfloor + 3T\left(\left\lfloor n/16 \right\rfloor\right)\right)$$

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

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

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

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

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

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

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

设$a \geq 1, b > 1$为常数，$f(n)$为函数， $T(n)$为非负整数

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

则有以下结果：

1. $f(n) = O(n^{\log_b a-\varepsilon}), \exists \varepsilon > 0,$ 那么$T(n) = \Theta(n^{\log_b a})$
2. $f(n) = \Theta(n^{\log_b a}),$ 那么$T(n) = \Theta(n^{\log_b a} \log n)$
3. $f(n) = \Omega(n^{\log_b a+\varepsilon}), \exists \varepsilon > 0,$ 且对于某个常数$c < 1$和所有的充分大的 n有 $af(n/b) \leq cf(n),$ 那么

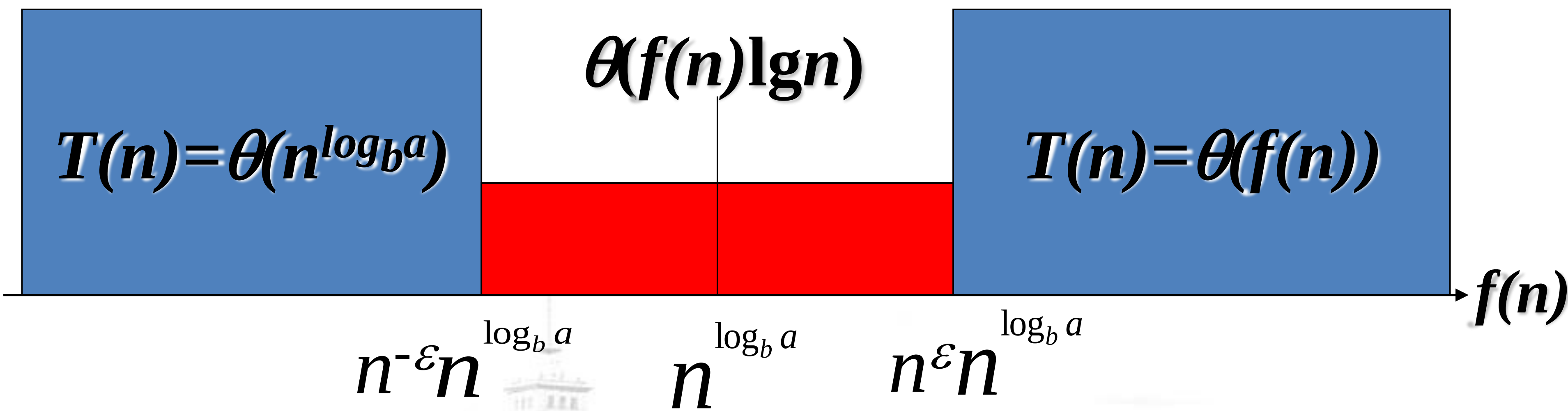
$$T(n) = \Theta(f(n))$$

*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

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

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

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

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

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

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

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

Master定理的使用（续）

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

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

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

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

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 $T(n)$.