算法设计与分析

主讲人：权丽君

Email: ljquan@*suda.edu.cn*

苏州大学 计算机学院







第二讲　函数增长

内容提要：

- 渐进记号
- 常用函数







第二讲　函数增长

内容提要：

- 渐进记号
  - ✓ 定义：O, Ω, Θ, o, ω
  - ✓ 证明例子
- 常用函数

限界函数







- 算法时间复杂度的限界函数常用的三个：

  上界函数、下界函数、渐进紧确界函数

  对应的渐进记号：O　　Ω　　Θ

- 算法的实际执行时间为 $f(n)$，分析所得的限界函数为 $g(n)$

✓ n ：问题规模的某种测度

✓ $f(n)$ ：算法的“实际”执行时间，与机器及语言有关

✓ $g(n)$：是事前分析的结果，一个形式简单的函数，通过对计数时间与频率计数统计分析得到，与机器及语言无关

渐近记号(Asymptotic notation)







- 表示渐近运行时间的记号用定义域为**自然数集的函数定义**。
- 为了方便，可以将定义域扩充至实数域或缩小到自然数集的某个受限子集上。
- **注意**：要清楚这些表示法的准确含义，以保证越规使用但不误用。

渐近上界-O记号







- **O(*g*(*n*))表示以下函数的集合：**

  $O(g(n)) = \{ f(n)$: 存在正常量 c 和 n_0, 使得对所有 $n \geq n_0$ 有 $0 \leq f(n) \leq cg(n) \}$.

  - 若 $f(n)$ 和 $g(n)$ 满足以上关系，记为 $f(n) \in O(g(n))$，表示 $f(n)$ 是集合 $O(g(n)$ 中的一员。通常记为：

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

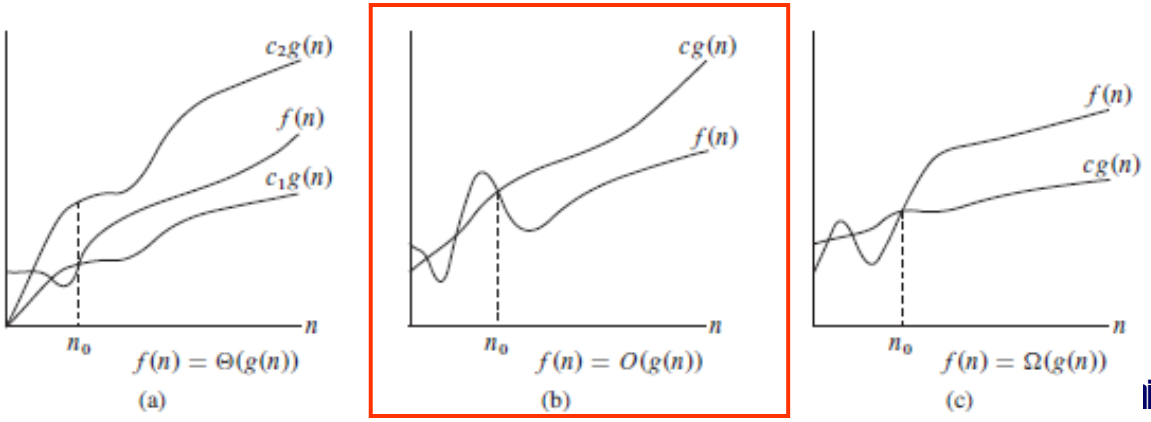
  - $f(n) = O(g(n))$ 表示如果算法用 n 值不变的同一类数据（规模相等，性质相同）在某台机器上运行，所用的时间总小于 $|g(n)|$ 的一个常数倍。

渐近上界-O记号

- O记号给出的是**渐进上界**，称为**上界函数 (upper bound)**。
- 上界函数代表了**算法最坏情况下的时间复杂度**，隐含地给出了在任意输入下运行时间的上界。
- 在确定上界函数时，应试图找**阶最小**的 $g(n)$ 作为 $f(n)$ 的上界函数——**紧确上界 (tight upper bound)**。

✓ 例：若：$3n+2=O(n^2)$，则是松散的界限；

✓ 若：$3n+2=O(n)$，则是紧确的界限。



渐近下界-Ω记号







- **$\Omega(g(n))$表示以下函数的集合：**

  $\Omega(g(n)) = \{ f(n)$: 存在正常量 c 和 n_0, 使得对所有 $n \geq n_0$,

  有$0 \leq cg(n) \leq f(n) \}$.

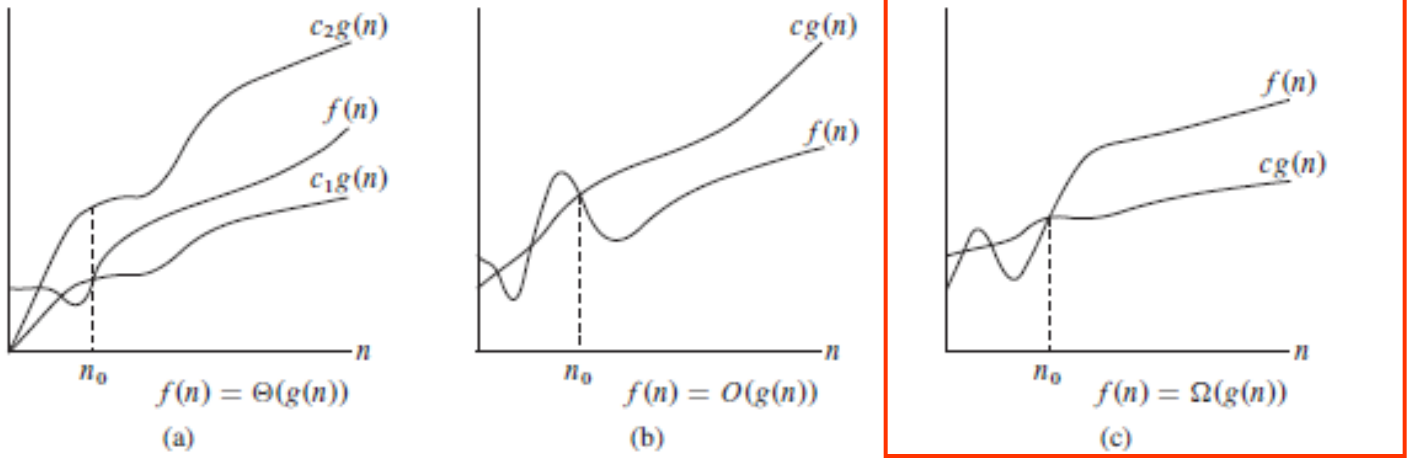
- 若 $f(n)$ 和 $g(n)$ 满足以上关系，记为 $f(n) \in \Omega(g(n))$，表示 $f(n)$是 $\Omega(g(n)$ 中的一员。通常记为：

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

- $f(n)=\Omega(g(n))$ 表示如果算法用 n 值不变的同一类数据在某台机器上运行，所用的时间总不小于 $|g(n)|$ 的一个常数倍。

渐近下界-Ω记号

- Ω 记号给出一个**渐进下界**，称为**下界函数 (lower bound)**。
- 下界函数代表了**算法最佳情况下的时间复杂度**，隐含地给出了在任意输入下运行时间的下界。
- 在确定下界函数时，应试图找出**数量级最大**的 $g(n)$ 作为 $f(n)$ 的下界函数——**紧确下界 (tight lower bound)**。



渐近紧确界-Θ记号







- **Θ(g(n))表示以下函数的集合：**

  $\Theta(g(n)) = \{ f(n)$: 存在正常量 $c_1, c_2,$ 和 $n_0,$ 使得对所有 $n \geq n_0,$ 有 $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \}.$

- 若 $f(n)$ 和 $g(n)$ 满足以上关系，记为 $f(n) \in \Theta(g(n))$，表示 $f(n)$是 $\Theta(g(n)$ 中的一员。通常记为：

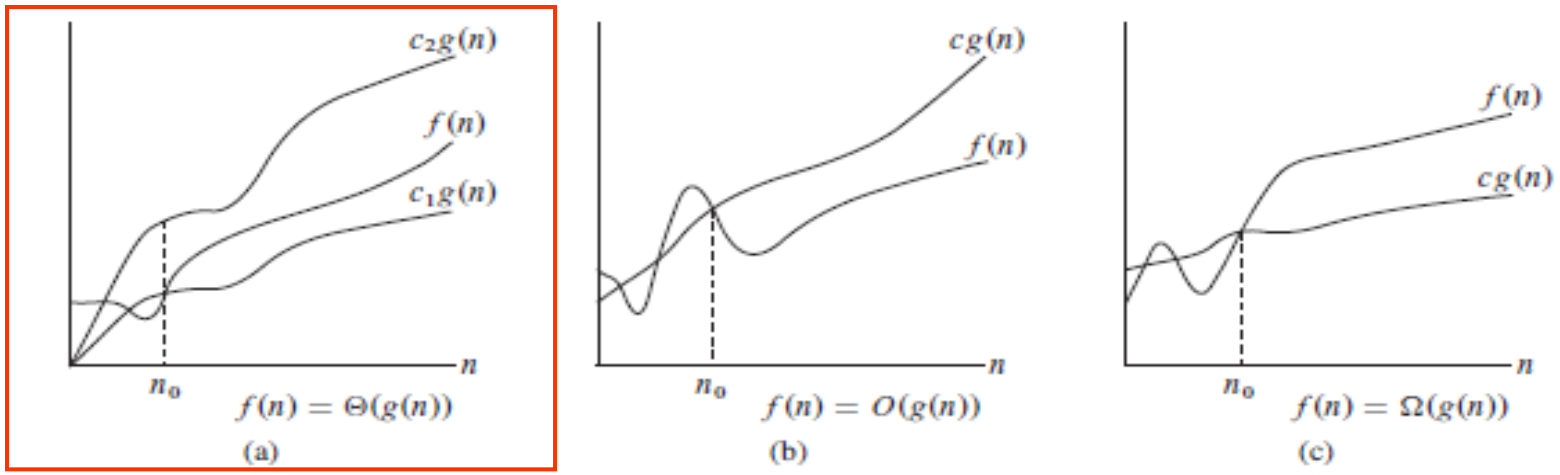
$$f(n) = \Theta(g(n))$$

- $f(n) = \Theta(g(n))$ 表示如果算法用 n 值不变的同一类数据在某台机器上运行，所用的时间既不小于$|g(n)|$的一个常数倍，也不大于$|g(n)|$的一个常数倍，亦即 g 既是 f 的下界，也是 f 的上界。

渐近紧确界-Θ记号

- Θ 记号给出的是**渐进紧确界** (asymptotically tight bound)。
- 从时间复杂度的角度看，$f(n) = \Theta(g(n))$ 表示是算法在最好和最坏情况下的计算时间就一个**常数因子范围内而言**是相同的，可看作：

既有 $f(n) = O(g(n))$, 又有 $f(n) = \Omega(g(n))$



渐近紧确界-Θ记号







例 1：证明 $1/2n^2-3n = \Theta(n^2)$。

分析：根据 Θ 的定义，仅需要确定正常量 c_1，c_2 和 n_0，使得对所有 $n \geq n_0$，有 $c_1n^2 \leq 1/2n^2-3n \leq c_2n^2$。

证明：两边同时除以 n^2 得：$c_1 \leq 1/2-3/n \leq c_2$。

选择任意常量 $c_2 \geq 1/2$，使得右边的不等式对于任何 $n \geq 1$ 成立；同样选择任意常量 $c_1 \leq 1/14$，使得左边不等式对于任何 $n \geq 7$ 成立。因此，通过选择 $c_1 = 1/14$，$c_2 = 1/2$， $n_0= 7$，得证 $1/2n^2-3n = \Theta(n^2)$。

N：还有其它常量可选，但根据定义，只要存在一组选择（如上述的 c_1，c_2 和 n_0）即得证。

渐近紧确界-Θ记号







例 2：证明 $6n^3 \neq \Theta(n^2)$

采用反证法：假设存在c_2 和 n_0 ，使得对所有 $n \geq n_0$ ，有 $6n^3 \leq c_2 n^2$。

两边同时除以 n^2 得：$n \leq c_2/6$,

而 c_2 是常量，所以对任意大的 n，该式不可能成立。

渐近紧确界-Θ记号

- 渐近正函数的低阶项在确定渐进紧确界时可以被忽略；最高阶项系数同样可以被忽略。
- 例：$f(n) = an^2 + bn + c$, 其中 a, b, c 均为常量，且$a > 0$。
  扔掉低阶项且忽略常量后得 $f(n) = \Theta(n^2)$。
- 一般地，对于任意多项式 $p(n) = \sum_{i=0}^{d} a_i n^i$ ，其中 a_i 为常量且 $a_d > 0$, 有 $p(n) = \Theta(n^d)$。
- 关于 Θ(1) (O(1)、Ω(1) 有类似的含义)：
  - 因为任意常量都可看做是一个0阶多项式，所以可以把任意常量函数表示成 $\Theta(n^0)$ 或 $\Theta(1)$
  - 通常用 $\Theta(1)$ 表示具有常量计算时间的复杂度，即算法的执行时间为一个固定量，与问题的规模 n 没关系。

渐近紧确界-Θ记号







- 定理 3.1 对任意两个函数 $f(n)$ 和 $g(n)$，我们有 $f(n) = \Theta(g(n))$，当且仅当 $f(n) = O(g(n))$ 且 $f(n) = \Omega(g(n))$。

证明： $\Rightarrow$: $f(n) = \Theta(g(n))$, then $\exists c_1 > 0, c_2 > 0, n_0 > 0$,

s.t. $n \geq n_0$, $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$

then $n \geq n_0$, $0 \leq f(n) \leq c_2 g(n) \Rightarrow f(n) = O(g(n))$

then $n \geq n_0$, $0 \leq c_1 g(n) \leq f(n) \Rightarrow f(n) = \Omega(g(n))$

$\Leftarrow$: $f(n) = O(g(n))$, then $\exists c_2 > 0, n_{20} > 0$,

s.t. $n \geq n_{20}$, $0 \leq f(n) \leq c_{20} g(n)$

$f(n) = \Omega(g(n))$, then $\exists c_{10} > 0, n_{10} > 0$,

s.t. $n \geq n_{10}$, $0 \leq c_{10} g(n) \leq f(n)$

let $n_0 = \max\{n_{10}, n_{20}\}$, then $n \geq n_0$,

$0 \leq c_{10} g(n) \leq f(n) \leq c_{20} g(n)$, that is $f(n) = \Theta(g(n))$.

等式和不等式中的渐近记号

□ 如 $n = O(n^2)$，$2n^2 + 3n + 1 = 2n^2 + \Theta(n)$ 等，如何来解释这些公式呢？

- 当渐近记号出现在等式的右边时，则等号表示左边的函数属于右边函数集合中的元素，即等号表示集合的成员关系，即 $n \in O(n^2)$。
- 当渐近记号出现在某个公式中时，将其解释为某个不关注名称的匿名函数，用以消除表达式中一些无关紧要的细节。如公式 $2n^2 + 3n + 1 = 2n^2 + \Theta(n)$ 即表示 $2n^2 + 3n + 1 = 2n^2 + f(n)$, 其中 $f(n)$ 是属于集合 $\Theta(n)$ 中的某个函数。这里假设 $f(n) = 3n+1$，该函数也确实属于 $\Theta(n)$ 中。
- 当渐近记号出现在等式左边时，如 $2n^2 + \Theta(n) = \Theta(n^2)$ ，用如下规则来解释：无论等号左边的匿名函数如何选择，总有办法选取等号右边的匿名函数使等式成立。这样，对于任意函数 $f(n) \in \Theta(n)$，存在函数 $g(n) \in \Theta(n^2)$ ，使得对所有的 n，有 $2n^2 + f(n) = g(n)$ 成立。换而言之，等式右边提供了较左边更少的细节。

非渐近紧确上界-o记号

- 大 O 记号所提供的渐近上界可能是、也可能不是渐近紧确的；例如 $2n^2 = O(n^2)$是渐近紧确的, 但 $2n = O(n^2)$ 却不是。
- 利用小 o 记号来表示**非渐近紧确的上界**，其定义如下：

  $o(g(n)) = \{ f(n)$: 对任意正常量 $c > 0$，存在常量 $n_0 > 0$，使得对所有 $n \geq n_0$，有$0 \leq f(n) < cg(n) \}$。
- **含义**：在 o 表示中，当 n 趋于无穷时，$f(n)$ 相对于 $g(n)$ 来说变得微不足道了，即 $\lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} = 0$。
- 例如 $2n = o(n^2)$, 但 $2n^2 \neq o(n^2)$。

非渐近紧确下界-ω记号







- ω 记号与 Ω 记号类似于 o 记号与 O 记号的关系。
- 利用小 ω 记号来表示**非渐近紧确的下界**，其定义如下：

  $\omega(g(n)) = \{f(n)$: 对任意正常量 $c > 0$，存在常量 $n_0 > 0$，使得对 所有 $n \geq n_0$，有$0 \leq cg(n) < f(n)\}$。
- **含义**：在 ω 表示中，当 n 趋于无穷时，$g(n)$ 相对于 $f(n)$ 来说变得微不足道了，即 $\lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} = \infty$。
- **例如 $n^2/2 = \Omega(n^2)$, 但 $n^2/2 \neq \omega(n^2)$。**

非渐近紧确下界-ω记号







- **O 和 o 的区别：**
  - ✓ **O:** $f(n) = O(g(n)) \Leftrightarrow \exists\ c,\ n_0: (n \geq n_0 \Rightarrow f(n) \leq cg(n))$
  - ✓ **o:** $f(n) = o(g(n)) \Leftrightarrow \forall\ c: (\exists\ n_0: n \geq n_0 \Rightarrow f(n) < cg(n))$
- **Ω 和 ω 的区别：**
  - ➢ **Ω:** $f(n) = \Omega(g(n)) \Leftrightarrow \exists\ c,\ n_0: (n \geq n_0 \Rightarrow cg(n) \leq f(n))$
  - ➢ **ω:** $f(n) = \omega(g(n)) \Leftrightarrow \forall\ c: (\exists\ n_0: n \geq n_0 \Rightarrow cg(n) < f(n))$

限界函数的性质







- 实数的许多关系性质也适用于渐近比较。下面假定 $f(n)$ 和 $g(n)$ 渐近为正 。

- 传递性 (Transitivity)

渐近紧界

$$f(n)=\Theta(g(n)) \text{ and } g(n)=\Theta(h(n)) \text{ imply } f(n)=\Theta(h(n)),$$
$$f(n)=O(g(n)) \text{ and } g(n)=O(h(n)) \text{ imply } f(n)=O(h(n)),$$
$$f(n)=\Omega(g(n)) \text{ and } g(n)=\Omega(h(n)) \text{ imply } f(n)=\Omega(h(n)),$$
$$f(n)=o(g(n)) \text{ and } g(n)=o(h(n)) \text{ imply } f(n)=o(h(n)),$$
$$f(n)=\omega(g(n)) \text{ and } g(n)=\omega(h(n)) \text{ imply } f(n)=\omega(h(n)).$$

渐近非紧上界

渐近非紧下界

- 自反性 (Reflexivity)

$$f(n)=\Theta(f(n)),$$
$$f(n)=O(f(n)),$$
$$f(n)=\Omega(f(n)).$$

渐近非紧界无自反性

限界函数的性质







口对称性 (Symmetry)

紧致界有对称性

$$f(n) = \Theta(g(n)) \text{ if and only if } g(n) = \Theta(f(n)).$$

口转置对称性 (Transpose symmetry)

$$f(n) = O(g(n)) \text{ if and only if } g(n) = \Omega(f(n)),$$

$$f(n) = o(g(n)) \text{ if and only if } g(n) = \omega(f(n)).$$

//大O与大Ω对调时，f、g对称

//f的非紧上界g等价于g是f的非紧下界







- 由上述4个性质， 可将两函数间的渐近比较类比于两个实数间 的比较。

$$f(n)=O(g(n))\approx a\le b\ //\text{“}\approx\text{”相似于}f\sim a,g\sim b$$
$$f(n)=\Omega(g(n))\approx a\ge b$$
$$f(n)=\theta(g(n))\approx a=b$$
$$f(n)=o(g(n))\approx a<b$$
$$f(n)=\omega(g(n))\approx a>b$$

但实数的三岐性(三分性质)不能类比到渐近表示中

三岐性 $\forall a,b\in\mathbb{R}$, 下述三种情况必有一个成立: $a<b, a=b, or\ a>b$

即任意两实数间是可比较的







- 并非所有函数都是渐近可比较的

即 $\exists f(n)$ 和 $g(n)$，

可能 $f(n)=O(g(n))$ 不成立，

而 $f(n)=\Omega(g(n))$ 也不成立

$\therefore$ 由 $Th2.1$ 知，$f \neq \theta(g)$

- 例：函数n和$n^{1+\sin n}$之间是无法渐近比较的

$1+\sin n \Rightarrow 0 \sim 2$

即 $n^{1+\sin}$ 在 $O(1) \sim O(n^2)$ 之间波动







第二讲 函数增长

内容提要：

- 渐进记号
- 常用函数

相关定理







- **定理 3.2 多项式定理** 若$\mathrm{A}(n)=a_m n^m+\cdots+a_1 n+a_0$是一个 n 的 m 次项式，其中 a_i 为常量，$i=0,\ldots,m$，且 $a_m>0$，则有 $\mathrm{A}(n)=\Theta(n^m)$。

证明：需要证明 $c_1 n^m \leqslant a_m n^m+\cdots+a_1 n+a_0 \leqslant c_2 n^m$。

取 $n_0=1$，当 $n \geq n_0$ 时，有

$$
\begin{aligned}
|\mathrm{A}(n)| &\leqslant |a_{\mathrm{m}}| n^m+\cdots+|a_1| n+|a_0| \\
&= \left(|a_m|+|a_{m-1}|/n+\cdots+|a_0|/n^m\right) n^m \\
&\leqslant \left(|a_m|+|a_{m-1}|+\cdots+|a_0|\right) n^m
\end{aligned}
$$

令 $c_2=|a_m|+|a_{m-1}|+\cdots+|a_0|$，

即有 $|\mathrm{A}(n)| \leq c_2 n^m_{25}=\mathrm{O}(n^m)$。

相关定理







需要证明 $c_1 n^m \leqslant a_m n^m + \cdots + a_1 n + a_0$。

$$
\begin{aligned}
\mathrm{A}(n) &= a_m n^m + \cdots + a_1 n + a_0 \\
&= \frac{1}{2} a_m n^m + \frac{1}{2} a_m n^m + a_{m-1} n^{m-1} + \cdots + a_0 \\
&= \frac{1}{2} a_m n^m + \left(\frac{a_m}{2m} n^m + a_{m-1} n^{m-1} \right) \\
&\quad + \left(\frac{a_m}{2m} n^m + a_{m-2} n^{m-2} \right) \\
&\quad + \cdots + \left(\frac{a_m}{2m} n^m + a_0 \right)
\end{aligned}
$$

相关定理







证明： $\because\ a_m > 0$，对于足够大的 n，有

$$\frac{a_m}{2m} n^m + a_{m-1} n^{m-1} \geqslant 0$$

$$\frac{a^m}{2m} n^m + a_{m-2} n^{m-2} \geqslant 0$$

$$\vdots$$

$$\frac{a^m}{2m} n^m + a_0 \geqslant 0$$

$\therefore$ 对于足够大的 n，有 $\mathrm{A}(n) \geqslant \frac{1}{2} a_m n^m = \Omega(n^m)$，

取 $c_1 = 1/2 a_m$。

相关定理







- **应用**：如果一个算法的时间复杂度函数是多项式形式，则其阶函数(复杂度函数表示)就可取该多项式的最高次项。

$$A(n) = a_m n^m + \cdots + a_1 n + a_0 \Rightarrow A(n) = \Theta(n^m)$$

- 事实上，根据渐近关系，对于足够大的 n，低阶项（包括常数项）是无足轻重的：当 n 较大时，即使最高阶项的一个很小部分都足以“支配”所有的低阶项。所以用阶函数表示限界函数时，低阶项和常数项均被忽略。

相关定理

- **例**：考虑二次函数 $f(n)=an^2+bn+c$，其中a、b、c为常量且 $a>0$

  根据上述思路，去掉低阶项并忽略常系数后即得：$f(n)=\Theta(n^2)$

  对比形式化证明：

  取常量：$c_1=a/4,\ c_2=7a/4,\ n_0=2\cdot\max\left(|b|/a,\sqrt{|c|/a}\right)$

  可以证明对所有的 $n\geq n_0$，有:

$$0\leqslant c_1n^2\leqslant an^2+bn+c\leqslant c_2n^2$$

相关定理

- **定理 3.3 用于估算复杂性（函数阶的大小）的定理**

  对于任意正实数 x 和 ε，有下面的不等式：

1. 存在某个 n_0，使得对于任何 $n \geq n_0$，有 $(\lg n)^x < (\lg n)^{x+\varepsilon}$。
2. 存在某个 n_0，使得对于任何 $n \geq n_0$，有 $n^x < n^{x+\varepsilon}$。
3. 存在某个 n_0，使得对于任何 $n \geq n_0$，有 $(\lg n)^x < n$。
4. 存在某个 n_0，使得对于任何 $n \geq n_0$，有 $n^x < 2^n$。
5. 对任意实数 y，存在某个 n_0，使得对于任何 $n \geq n_0$，有 $n^x (\lg n)^y < n^{x+\varepsilon}$。

相关定理

- **例**：根据定理 3.3，很容易得出：

$$n^3 + n^2 \lg n = O(n^3)\ ;$$

$$n^4 + n^{2.5}\lg^{20} n = O(n^4)\ ;$$

$$2^n n^4 \lg^3 n + 2^n n^5 / \lg^3 n = O(2^n n^5)。$$

相关定理

- **定理 3.4 设 $d(n)$、$e(n)$、$f(n)$ 和 $g(n)$ 是将非负整数映射到非负实数的函数，则：**

1. 如果 $d(n)$ 是 $O(f(n))$，那么对于任何常数 $a>0$，$ad(n)$是 $O(f(n))$；
2. 如果 $d(n)$ 是 $O(f(n))$，$e(n)$是 $O(g(n))$，那么 $d(n)+e(n)$ 是 $O(f(n)+g(n))$ —— 加法法则；
3. 如果 $d(n)$ 是 $O(f(n))$，$e(n)$是 $O(g(n))$，那么 $d(n)e(n)$ 是 $O(f(n)g(n))$ —— 乘法法则；
4. 对于任意固定的 $x>0$ 和 $a>1$，n^x 是 $O(a^n)$ —— 任意底大于1的指数函数比任意多项式函数增长得快；
5. 对于任意固定的 $x>0$，$\lg n^x$ 是 $O(\lg n)$；
6. 对于任意固定的常数 $x>0$ 和 $y>0$，$\lg^x n$ 是 $O(n^y)$ —— 任意正的多项式函数都比任意对数函数增长得快。

算法时间复杂度的分类

- 根据**上界函数**的特性，可以将算法分为：**多项式时间算法**和**指数时间算法**。

➢ **多项式时间算法**：可用多项式函数对计算时间限界的算法

常见的多项式限界函数有：

$$o(1) < o(\log n) < o(n) < o(n\log n) < o(n^2) < o(n^3)$$

复杂度越来越高

➢ **指数时间算法**：计算时间用指数函数限界的算法

常见的指数时间限界函数：

$$o(2^n) < o(n!) < o(n^n)$$

复杂度越来越高

算法时间复杂度的分类

- 当 n 取值较大时，**指数时间算法**和**多项式时间算法**在计算时间上非常悬殊。

计算时间的典型函数曲线：

logn	n	nlogn	n_2	n^3	2^n
0	1	0	1	1	2
1	2	2	4	8	4
2	4	8	16	64	16
3	8	24	64	512	256
4	16	64	256	4096	65536
5	32	160	1024	32768	4294967296

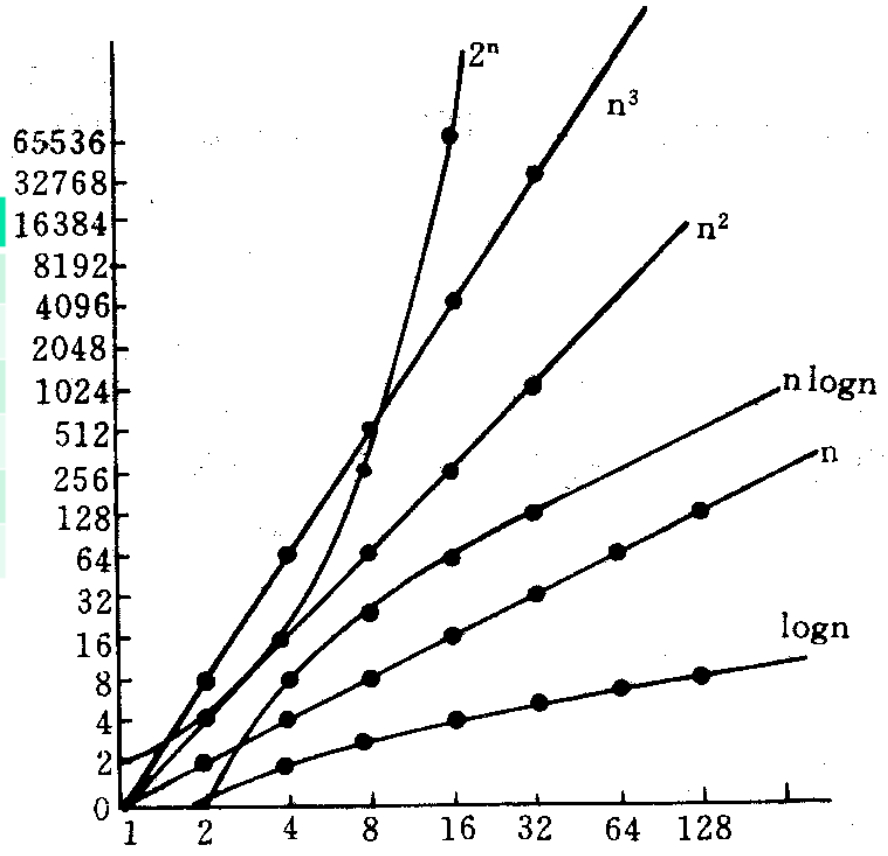


图 1.1 一般计算时间函数的曲线

对算法复杂性的一般认识







- 当数据集的规模很大时，要在现有的计算机系统上运行具有比 $O(n\log n)$ 复杂度还高的算法是比较困难的。
- 指数时间算法只有在 n 取值**非常小**时才实用。
- 要想在**顺序处理机**上扩大所处理问题的规模，有效的途径是**降低算法的计算复杂度**，而不是（仅仅依靠）提高计算机的速度。







谢谢！

作业： 3.2-3

思考题：3-2