# 算法基础

主讲人：　庄连生

Email: { *lszhuang@ustc.edu.cn* }

***Spring 2018***，**USTC**

**University of Science and Technology of China**

# 第二讲  函数增长

**内容提要：**

- 渐进记号
- 常用函数
- 级数求和

- $n \to \infty$, 归并排序$\Theta(n\lg n)$, 算法性能优于 插入排序$\Theta(n^2)$
- 利用函数增长率来描述算法效率，并可以用来比较各种算法的相对性能；
- A way to describe behavior of functions in the limit. We're studying asymptotic efficiency.（**函数的渐近效率，即极限情况下的函数行为**）
- Focus on what's important by abstracting away low-order terms and constant factors.（**通常忽略低阶项和常数因子**）
- How we indicate running times of algorithms.（**如何描述算法的运算时间**）
- A way to compare "sizes" of funcitons（**比较函数大小的方法**）

$$o \approx < \;;\; O \approx \le \;;\; \Theta \approx = \;;\; \Omega \approx \ge \;;\; \omega \approx >$$

# 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
- 级数求和

# Asymptotic notation（渐近记号）

- the asymptotic running time are defined in terms of functions whose domains are the set of **natural numbers** $N = \{0,1,2,\ldots\}$.（运行时间函数的定义域为自然数集）
- **Abuse**

  just for convenient

  for example, extended to the real numbers domain
- **Not misused**

  We need understand the precise meaning of the notation when it is abused. It is not misused.

# Θ-notation: asymptotically tight bound（渐近紧界）

- **What this notation $T(n)=\Theta(n^2)$ means**

For a given function $g(n)$, we denote by $\Theta(g(n))$ the set of functions

$\Theta(g(n)) = \{f(n)$: there exist positive constants $c_1$, $c_2$ , and $n_0$ such that

$$0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0\}.$$

we could write "$f(n) \in \Theta(g(n))$" to indicate that $f(n)$ is a member of $\Theta(g(n))$.

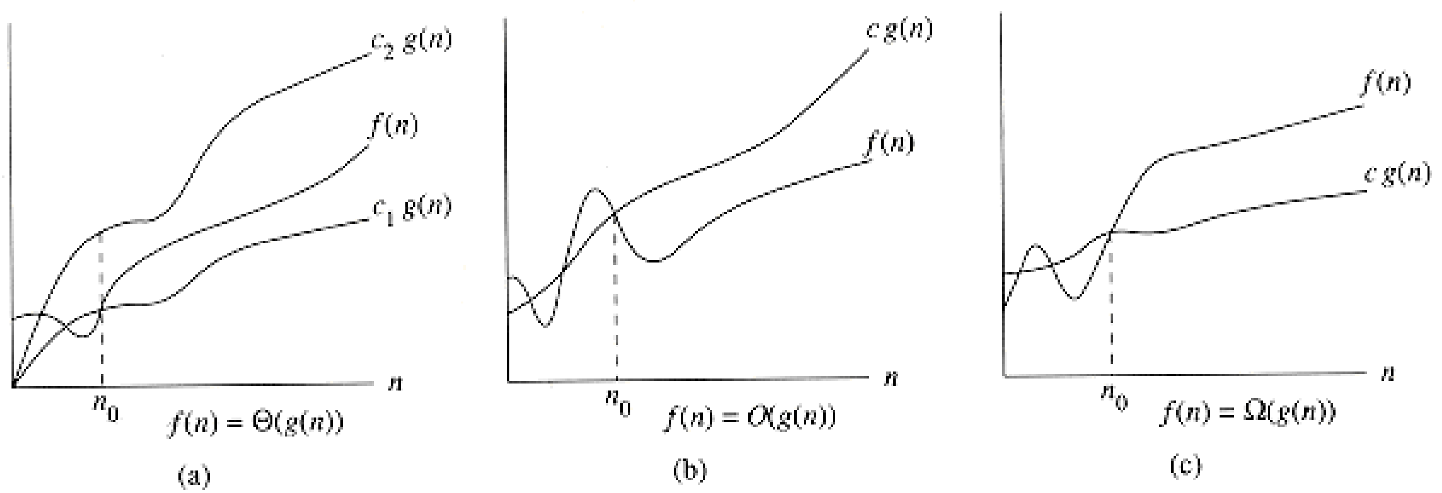
Instead, we will usually write "$f(n)=\Theta(g(n))$" to express the same notion. The abuse may at first appear confusing, but it has advantages.

# Θ-notation: asymptotically tight bound（渐近紧界）

- $T(n)=\Theta(n^2)$

For a given function $g(n)$, we denote by $\Theta(g(n))$ the set of functions

$$\Theta(g(n)) = \{f(n): \text{there exist positive constants } c_1, c_2 \text{, and } n_0 \text{ such that } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0\}.$$




We say that $g(n)$ is an asymptotically tight bound for $f(n)$.

# Θ-notation: asymptotically tight bound（渐近紧界）

$\Theta(g(n)) = \{ f(n)$: there exist positive constants $c_1$, $c_2$ , and $n_0$ such that $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0$ $\}$.

- In this chapter, assume that every asymptotic notations are asymptotically nonnegative.（设所有的渐近符号为渐近非负）
- Example: How to show that n2/2-3n=Θ(n2) ?

  We must determine positive constants c1, c2, and n0 such that

$$c_1 n^2 \le n^2/2 - 3n \le c_2 n^2$$
$$=> \quad c_1 \le 1/2 - 3/n \le c_2$$

  by choosing c1 =1/14, c2=1/2, and n0=7, we can verify that n2/2-3n=Θ(n2)
- Other choices for the constants may exist. The key is some choice exists.（可能有多个值，只要存在某个值就可以了）

# Θ-notation: asymptotically tight bound（渐近紧界）

$\Theta(g(n)) = \{f(n)$: there exist positive constants $c_1$, $c_2$ , and $n_0$ such that $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0$ }.

- How verify that $6n^3 \neq \Theta(n^2)$ ?

  Suppose for the purpose of contradiction that c2 and n0 exist such that $6n^3 \le c_2 n^2$ for all $n \ge n_0$ . But then $n \le c_2 / 6$ , which cannot possibly hold for arbitrarily large n, since c2 is constant.

# Θ-notation: asymptotically tight bound（渐近紧界）

$\Theta(g(n))$ = { $f(n)$: there exist positive constants $c_1$, $c_2$ , and $n_0$ such that $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0$ }.

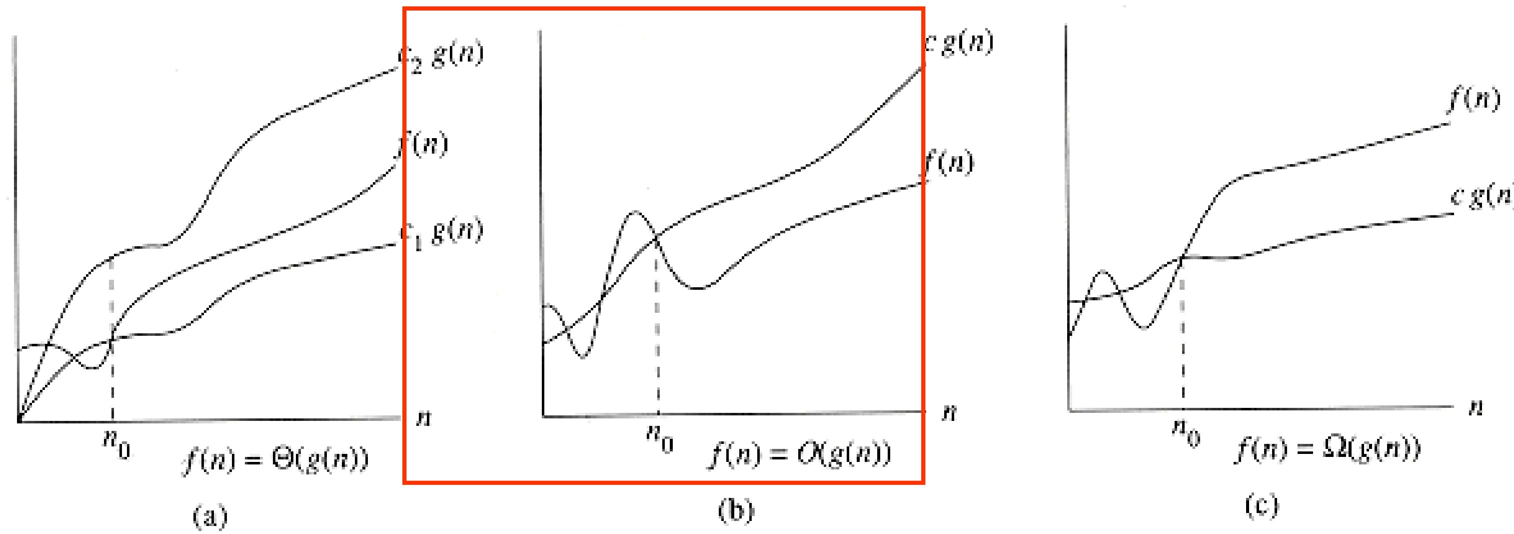
- The lower-order terms, the coefficient of the highest-order term can be ignored.
- Example: $f(n) = an^2 + bn + c$ , where a>0, b, c are constants.

  Throwing away the lower-order terms and ignoring the constant yields $f(n) = \Theta(n^2)$ . (Page44)
- In general, for any polynomial $p(n) = \sum_{i=0}^{d} a_i n^i$ , where the ai are constants and ad >0, we have $p(n) = \Theta(n^d)$ .
- We can express any constant function as $\Theta(n^0)$ or $\Theta(1)$ .

  $\Theta(1)$ often mean either a constant or a constant function.

# *O*-notation: asymptotic upper bound（渐近上界）

- ***O* – notation:** For a given function $g(n)$, we denote by $O(g(n))$ the set of functions $O(g(n)) = \{f(n)$: there exist positive constants $c$ and $n_0$ such that $0 \le f(n) \le c\, g(n)$ for all $n \ge n_0\}$.




- "$f(n) = O(g(n))$" indicates "$f(n) \in O(g(n))$"
- $f(n) = \Theta(g(n)) \Rightarrow f(n) = O(g(n))$

  $\Rightarrow \Theta(g(n)) \subseteq O(g(n))$

# $O$-notation: asymptotic upper bound

- **$O$ – notation:** For a given function $g(n)$, we denote by $O(g(n))$ the set of functions $O(g(n)) = \{f(n)$: there exist positive constants $c$ and $n_0$ such that $0 \le f(n) \le c\, g(n)$ for all $n \ge n_0\}$.

- Example: $2n^2 = O(n^3)$, with $c=1$ and $n_0=2$
- Example of functions in $O(n^2)$

$$n^2 \qquad n$$

$$n^2 + n \qquad n/3000$$

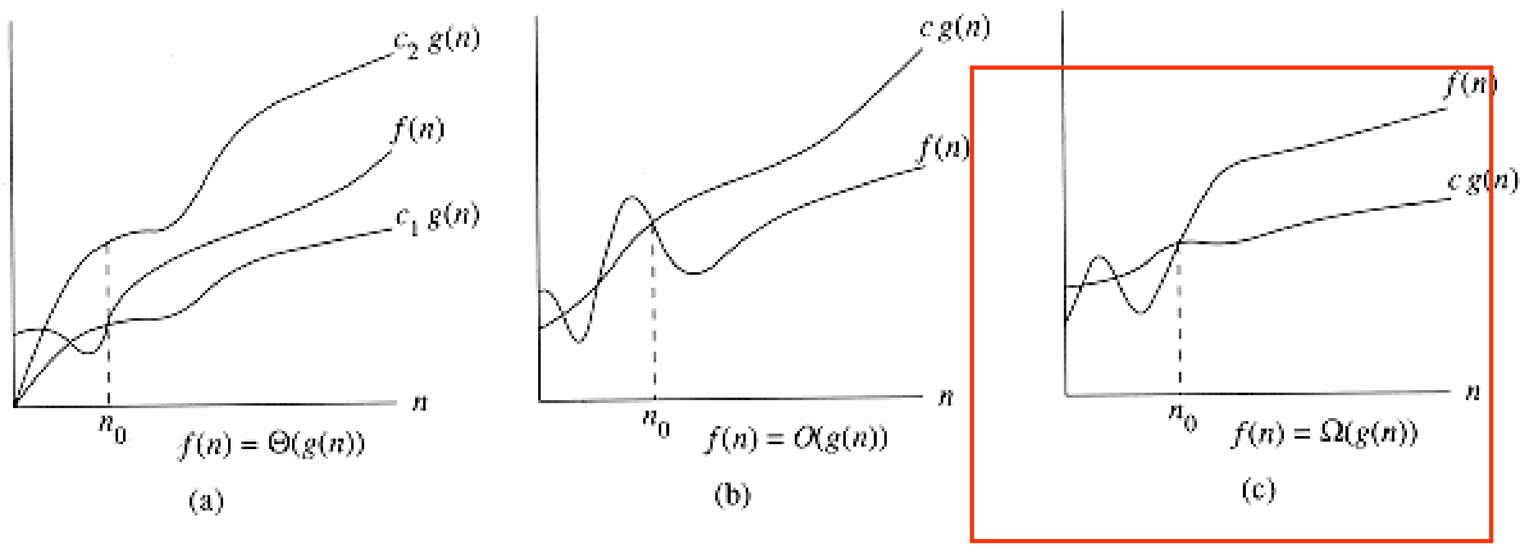
$$n^2 + 2000n \qquad n^{1.99999}$$

$$500n^2 + 1000n \qquad n^2/\lg\lg\lg n$$

# Ω-notation: asymptotic lower bound（渐近下界）

- **Ω – notation: For a given function $g(n)$, we denote by $\Omega(g(n))$ the set of functions $\Omega(g(n)) = \{f(n)$: there exist positive constants $c$ and $n_0$ such that**

$$0 \le c\,g(n) \le f(n) \text{ for all } n \ge n_0\}.$$



# Ω-notation: 渐近下界

- $\Omega$ – notation: For a given function $g(n)$, we denote by $\Omega(g(n))$ the set of functions $\Omega(g(n)) = \{f(n)$: there exist positive constants $c$ and $n_0$ such that

$$0 \le c\, g(n) \le f(n) \text{ for all } n \ge n_0\}.$$

- **Example of functions in $\Omega(n^2)$**

$$n^2$$
$$n^2 + n$$
$$n^2 + 2000n$$
$$500n^2 - 1000n$$

$$n^3$$
$$n^{2.0000001}$$
$$n^2 \lg\lg\lg n$$

# Ω-notation: 渐近下界

- **Theorem 3.1**

  **For any two functions $f(n)$ and $g(n)$, we have $f(n)=\Theta(g(n))$ if and only if $f(n)=O(g(n))$ and $f(n)=\Omega(g(n))$.**

  **Prove:**

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

s.t. $n \ge n_0$ , $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$

then $n \ge n_0$, $0 \le f(n) \le c_2 g(n) \Rightarrow f(n)=O(g(n))$

then $n \ge n_0$, $0 \le c_1 g(n) \le f(n) \Rightarrow f(n)=\Omega(g(n))$

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

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

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

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

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

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

# Ω-notation: 渐近下界

- **Theorem 3.1**

  **For any two functions $f(n)$ and $g(n)$, we have $f(n)=\Theta(g(n))$ if and only if $f(n)=O(g(n))$ and $f(n)=\Omega(g(n))$.**

- **In practice, rather than using the theorem to obtain asymptotic upper and lower bounds from asymptotically tight bounds, we usually use it to prove asymptotically tight bounds from asymptotic upper and tower bounds.**

**（定理作用：实际中，通常根据渐近上界和渐近下界来证明渐近紧界，而不是根据渐近紧界来得到渐近上界和渐近下界。）**

# Ω-notation: 渐近下界

- The running time of insertion sort falls between $\Omega(n)$ and $O(n^2)$, the bounds are asymptotically tight.
- The running time of insertion sort is not $\Omega(n^2)$. Why?
- It is not contradictory to say that the worst-case running time of insertion sort is $\Omega(n^2)$. Why?
- The running time of an algorithm is $\Omega(g(n))$, we mean that no matter what particular input of size $n$ is chosen for each value of $n$, the running time on that input is at **least** a constant times $g(n)$, for large $n$.

（算法的运行时间为$\Omega(g(n))$意味着对足够大的$n$，对输入规模为 $n$ 的任意输入,其运算时间至少是$g(n)$的一个常数倍。）

# 等式和不等式中的渐近记号

- **How to interpret: "$n=O(n^2)$", "$2n^2+3n+1= 2n^2+\Theta(n)$" , …**
  - **When the asymptotic notation on the right-hand side alone, as $n=O(n^2)$, it means set membership: $n\in O(n^2)$.**
  - **When in a formula, it stands for some anonymous function that we do not care to name. For example, "$2n^2+3n+1=2n^2+\Theta(n)$" means that "$2n^2+ 3n+1=2n^2+f(n)$", where $f(n)$ is some anonymous function in the set $\Theta(n)$.**（代表存在某个匿名函数使得方程成立）
  - **When on the left-hand side , as in "$2n^2 +\Theta(n)=\Theta(n^2 )$", it can be interpretted: *No matter how the anonymous functions are chosen on the left of the equal sign, there is a way to choose the anonymous functions on the equal sign to make the equation valid.*** （任给$f(n)\in\Theta(n)$，存在$g(n)\in\Theta(n^2)$，使得方程成立）
- **From 2) and 3), we have "$2n^2+ 3n+1=2n^2+\Theta(n)=\Theta(n^2)$".**

# *o*-notation: (非渐近紧确上界）

- The bound provided by *O*-notation may or may not be asymptotically tight.
- The bound $2n^2=O(n^2)$ is asymptotically tight, but the bound $2n=O(n^2)$ is not.
- The *o*-notation denotes an **upper bound** that is **not asymptotically tight**. Formally, define $o(g(n))$ as the set
  （渐近非紧上界）

$o(g(n))$ = {$f(n)$: for any positive constants $c>0$, there exits a constant $n_0>0$ such that $0\le f(n)<c\,g(n)$ for all $n\ge n_0$}.

For example, $2n=o(n^2)$, but $2n^2\neq o(n^2)$.

# ω-notation — 非渐近紧确下界

- $\omega$-notation is to $\Omega$-notation as $o$-notation is to $O$-notation.
- The $\omega$-notation denotes an **lower bound** that is **not asymptotically tight**. Formally, define $\omega(g(n))$ as the set

  $\omega(g(n))$ = { $f(n)$: for any positive constants $c>0$, there exits a constant $n_0>0$ such that $0 \le c\, g(n) < f(n)$ for all $n \ge n_0$}.

  One way to define it is by

  $f(n) \in \omega(g(n))$ if and only if $g(n) \in o(f(n))$

  For example, $n^2/2 = \omega(n)$, but $n^2/2 \ne \omega(n^2)$.

- The relation $f(n)=\omega(g(n))$ implies that

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty, \text{ if the limit exists.}$$

# Comparison of functions（函数比较）

- Many of the relational properties of **real number** apply to asymptotic comparisons.

  For the following, Assume that $f(n)$ and $g(n)$ are asymptotically positive.

- Transitivity（传递性）

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

# Comparison of functions（函数比较）

- **Reflexivity（自反性）**

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

- **Symmetry（对称性）**

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

- **Transpose symmetry（反对称性）**

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

# Comparison of functions（函数比较）

- **An analogy between the asymptotic comparison of two functions and the comparison of two real numbers**

  **（函数渐近性比较与实数比较的类比）**

$$f(n)=o(g(n)) \quad \approx \quad a<b,$$

$$f(n)=O(g(n)) \quad \approx \quad a\le b,$$

$$f(n)=\Theta(g(n)) \quad \approx \quad a=b,$$

$$f(n)=\Omega(g(n)) \quad \approx \quad a\ge b,$$

$$f(n)=\omega(g(n)) \quad \approx \quad a>b.$$

# Comparison of functions（函数比较）

- **One property of real numbers, does not carry over to asymptotic notation**
  - **Trichotomy（三分法）: any two real numbers *a* and *b*, one of the following must holds: $a<b$, $a=b$, or $a>b$.**
  - **Not all functions are asymptotically comparable. That is, for two functions $f(n)$ and $g(n)$, it may be the case that neither $f(n)=O(g(n))$ nor $f(n)=\Omega(g(n))$ holds.**

    **For example, the functions *n* and $n^{1+\sin n}$ cannot be compared using asymptotic notation.**

$$-1 \le \sin n \le 1 \Rightarrow n^0 \le n^{1+\sin n} \le n^2$$

$$n^{1+\sin n} \le n \le n^{1+\sin n} \quad ???$$

# 第二讲 函数增长

**内容提要：**

- 渐进记号
- 常用函数
- 级数求和

# Standard notation and common function

- **Monotonicity**（单调性）
- **Floors and ceilings** $x-1<\lfloor x \rfloor \le x \le \lceil x \rceil < x+1$
- **Modular arithmetic (remainder or residue)**（模运算）

$$a \bmod n = a - \lfloor a/n \rfloor n$$

- **Polynomials**（多项式）

$$p(n)=\sum_{i=0}^{d} a_i n^i$$

- **Exponentials**（指数）
- **Logarithms**（对数）
- **Factorials**（阶乘）

斯特林近似公式

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1+\theta\left(\frac{1}{n}\right)\right)$$

$$==> n! = o(n^n), n! = \omega(2^n), \lg(n!) = \theta(n \lg n)$$

# Standard notation and common function

- **Functional iteration（迭代函数）**

  **We use the notation $f^{(i)}(n)$ to denote the function $f(n)$ iteratively applied $i$ times to an initial value of $n$. For non-negative integers $i$, we recursively define**

$$f^{(i)}(n) = \begin{cases} n & \text{if } i=0 \\ f(f^{(i-1)}(n)) & \text{if } i>0 \end{cases}$$

  **For example, if $f(n)=2n$, then**

$$f^{(2)}(n) = f(f(n)) = f(2n) = 2(2n) = 2^2 n$$

$$\cdots$$

$$f^{(i)}(n) = 2^i n$$

# Standard notation and common function

- **The iterated logarithm function（多重对数函数）**

  **We use the notation $\lg^* n$ to denote the iterated logarithm. Let $\lg^{(i)} n$ be iterated function, with $f(n)=\lg n$, that is**

  **$\lg^{(i)}(n)=\lg(\lg^{(i-1)}(n))$ . $\lg^{(i)} n$ is defined only if $\lg^{(i-1)} n>0$. Be sure to distinguish $\lg^{(i)} n$ from $\lg^i n$. $\lg^* n$ is defined as**

$$\lg^* n = \min\{i \geq 0 : \lg^{(i)} n \leq 1\}$$

  **The iterated logarithms is a very slowly growing function:**

$$\lg^* 2 = 1, \qquad \lg^* 4 = 2, \qquad \lg^* 16 = 3,$$

$$\lg^* 65536 = 4, \quad \lg^* 2^{65536} = 5.$$

  **$2^{65536} >> 10^{80}$. Rarely encounter an input size $n$ such that $\lg^* n>5$.**

- **Fibonacci numbers**

# 第二讲 函数增长

**内容提要：**

- 渐进记号
- 常用函数
- 级数 求和

# 级数求和

- 定义：有限和、无限和、级数收敛、级数发散的、绝对收敛级数
- 等差级数： $\sum_{k=1}^{n} k = 1+2+\ldots+n = \frac{1}{2}n(n+1) = \Theta(n^2)$
- 平方和与立方和：

$$\sum_{k=0}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}, \sum_{k=0}^{n} k^3 = \frac{n^2(n+1)^2}{4}$$

- 几何级数：

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

- 调和级数：

$$H_n = 1 + \tfrac{1}{2} + \tfrac{1}{3} + \ldots + \tfrac{1}{n} = \sum_{k=1}^{n} \tfrac{1}{k} = \ln n + O(1)$$

- 级数的积分和微分：

$$\sum_{k=0}^{\infty} kx^k = \frac{x}{(1-x)^2}$$

# 确定求和时间的界

## □ 数学归纳法

1）计算级数的准确值

2）计算和式的界，如：证明几何级数 $\sum_{k=0}^{n} 3^k$ 的界是 $O(3^n)$

3）一个容易犯的错误，如：证明 $\sum_{k=1}^{n} k = O(n)$

## □ 确定级数各项的界

1）一个级数的理想上界可以通过对级数中的每个项求界来获得；

2）一个级数实际上以一个几何级数为界时，可以选择级数的最大项作为每项的界（注意防止犯错！）

□ **分割求和：** 可以将级数表示为两个或多个级数，按下标的范围进行划分，然后再对每个级数分别求界。

# 谢谢！

## Q & A