Growth of Functions

交通大學 資訊工程系

Asymptotic Notation[1]

- **Asymptotic efficiency of algorithm**
  - **When input sizes are large enough, only the order of growth of the running time is relevant.**
  - **For large enough inputs**
    - **The *multiplicative constants* and *lower-order terms* of an exact running time are dominated by the effects of the input size itself.**
  - **We concern with how the running time of an algorithm increases with the size of the input *in the limit*.**
  - **An algorithm that is asymptotically more efficient will be the best choice for all but very small inputs.**

Asymptotic Notation^2

☞ *Asymptotic notation*

- The asymptotic running time of an algorithm are defined in terms of functions whose domain are the set of *natural numbers*.
- Asymptotic notation actually applies to functions, for example, insertion sort's worst-case running time $an^2 + bn + c$.
- Except running time, asymptotic notation also can be applied to functions that characterize some other aspect of algorithms.
  - Space

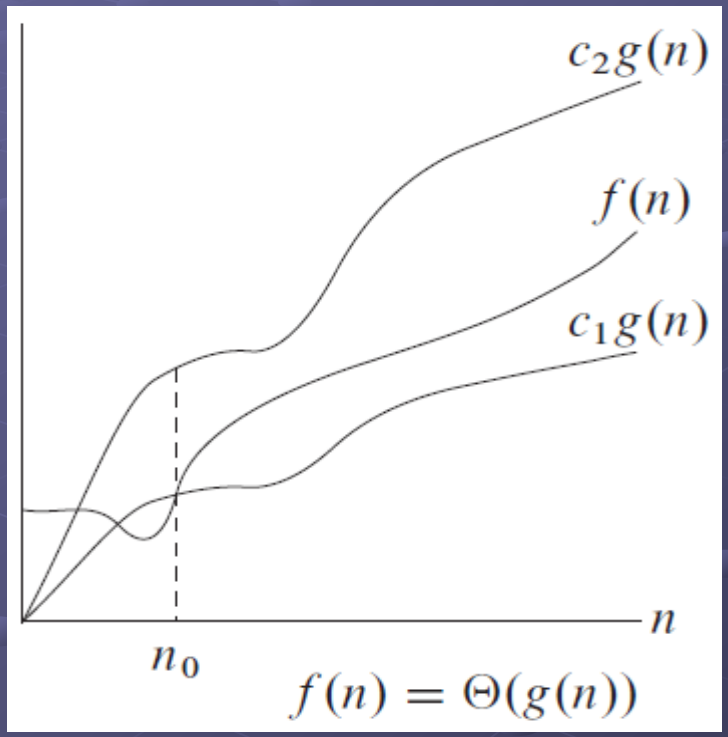
Θ-Notation[1]

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

- $f(n)$ is a member of $\Theta(g(n))$: "$f(n) \in \Theta(g(n))$".
  - We usually abuse "$f(n) = \Theta(g(n))$" to express the same expression, but it has its advantages.
  - For all $n \ge n_0$, $f(n)$ is equal to $g(n)$ to within a constant factor. (Fig. 3.1a)
  - $g(n)$ is an *asymptotically tight bound* for $f(n)$.

Θ-Notation[2]



☞ The definition of $\Theta(g(n))$ requires every member of $f(n) \in \Theta(g(n))$ be ***asymptotically nonnegative***.

Θ-Notation[3]

- **Example: Show that ½ $n^2 - 3n = \Theta(n^2)$.**

  To do so, we must determine positive constants c_1, c_2, and n_0 such that

  $$c_1 n^2 \le \frac{1}{2} n^2 - 3n \le c_2 n^2$$

  for all $n \ge n_0$. Dividing by n^2 yields

  $$c_1 \le \frac{1}{2} - \frac{3}{n} \le c_2$$

  by choosing c_1 = 1/14, c_2 = ½ , and n_0 = 7, we can verify that ½ $n^2 - 3n = \Theta(n^2)$.

Θ-Notation[4]

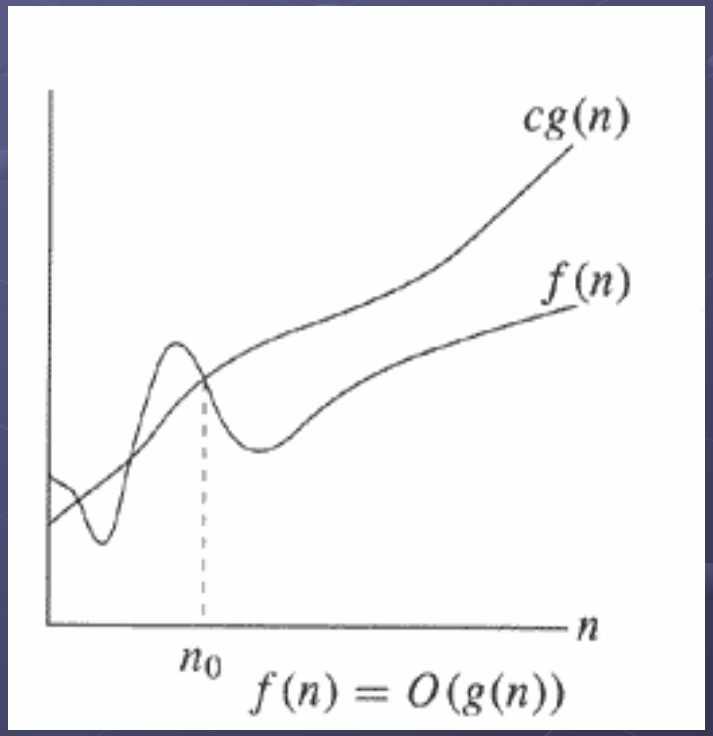
- Example: $6n^3 \neq \Theta(n^2)$.
  - Show there is a contradiction that c_2 and n_0 exist such that $6n^3 \leq c_2n^2$ for all $n \geq n_0$.

    For $n \leq c_2/6$, which can not possibly hold for arbitrarily large n, since c_2 is constant.

O-Notation[1]

☞ ***O-Notation*: *Asymptotic upper bound*.**

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



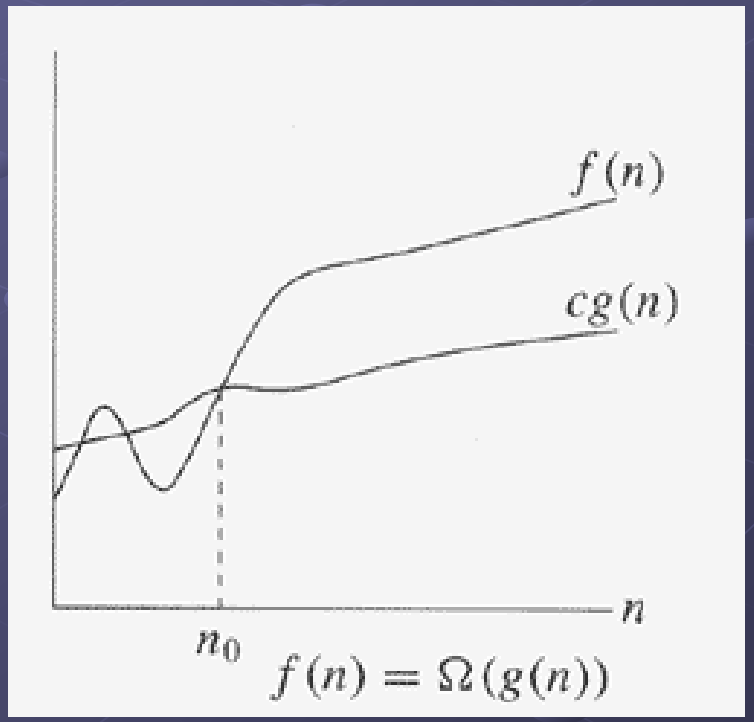
O-Notation^2

- $f(n) = \Theta(g(n))$ implies $f(n) = O(g(n))$, $\Theta(g(n)) \subseteq O(g(n))$.
- Any linear function $an + b$ is in $O(n^2)$.
- When we write $f(n) = O(g(n))$,
  - Claiming some constant multiple of $g(n)$ is an asymptotic upper bound on $f(n)$.
  - With no claim about how tight an upper bound it is.
  - A bound on the worst-case running time of the algorithm ***on every input***.
- An abuse to say that the running time of insertion sort is $O(n^2)$.
  - The actual running varies, depending on the particular input of size n.

Ω-Notation[1]

☞ *Ω-notation*: *Asymptotic lower bound*.

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



Ω-Notation[2]

- **Theorem 3.1**
  - For any two functions $f(n)$ and $g(n)$, we have $f(n) = \Theta(g(n))$ iff $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.
- **The running time of an algorithm is $\Omega(g(n))$**
  - 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 sufficiently large n.

Asymptotic Notation in Equations and Inequalities[1]

- How do we interpret the following formulas?

  $n = O(n^2)$

  $2n^2 + 3n + 1 = 2n^2 + \Theta(n)$

- Stand alone on the right-hand side of an equation
  - The equal sign means "*set membership*".

    For example: $n = O(n^2)$ means $n \in O(n^2)$
- When asymptotic notation appears in a formula:
  - Standing for some *anonymous function*.

    For example: $2n^2 + 3n + 1 = 2n^2 + \Theta(n)$ means $2n^2 + 3n + 1 = 2n^2 + f(n)$.

Asymptotic Notation in Equations and Inequalities[2]

- **Asymptotic notation appears on the left-hand side:**
  - ***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 right of the equal sign to make the equation valid.***
  - **The right-hand side of an equation provides a *coarser level of detail* than the left-hand side.**

Example: For any $f(n) \in \Theta(n)$, $\exists\, g(n) \in \Theta(n^2)$ *s.t.*
$2n^2 + f(n) = g(n)$ **for all** n.

o-Notation and *ω*-Notation

- **o-Notation: Denote an upper bound that is not asymptotically tight.**

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

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

- **ω-Notation: Denote a lower bound that is not asymptotically tight.**

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

$$f(n) = \omega(g(n)) \Leftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$$

Comparison of Functions[1]

- **Transitivity:**

$f(n) = \Theta(g(n)) \wedge g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$

$f(n) = O(g(n)) \wedge g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$

$f(n) = \Omega(g(n)) \wedge g(n) = \Omega(h(n)) \Rightarrow f(n) = \Omega(h(n))$

$f(n) = o(g(n)) \wedge g(n) = o(h(n)) \Rightarrow f(n) = o(h(n))$

$f(n) = \omega(g(n)) \wedge g(n) = \omega(h(n)) \Rightarrow f(n) = \omega(h(n))$

- **Reflexivity**

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

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

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

Comparison of Functions[2]

- **Symmetry**

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

- **Transpose symmetry**

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

  $f(n) = o(g(n)) \Leftrightarrow g(n) = \omega(f(n))$

- **Analogy to the comparison of two real numbers**

  $f(n) = O(g(n)) \approx a \leq b$

  $f(n) = \Omega(g(n)) \approx a \geq b$

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

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

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

Comparison of Functions[3]

- $f(n)$ is *asymptotically smaller* than $g(n)$ if $f(n) = o(g(n))$.
- $f(n)$ is *asymptotically larger* than $g(n)$ if $f(n) = \omega(g(n))$.
- *Trichotomy*: For any two real numbers a and b, exactly one of the following must hold:
  - $a < b$, $a = b$, or $a > b$
- Not all functions are asymptotically comparable.
  - The case that neither $f(n) = O(g(n))$ nor $f(n) = \Omega(g(n))$ holds.
  - For example: n v.s. $n^{1+\sin n}$
    - The exponent of $n^{1+\sin n}$ oscillates between 0 and 2.

Standard Notations and Common Functions[1]

- **Monotonicity**
  - A function f is ***monotonically increasing*** if $m \leq n$ implies $f(m) \leq f(n)$.
  - A function f is ***monotonically decreasing*** if $m \leq n$ implies $f(m) \geq f(n)$.
  - A function f is ***strictly increasing*** if $m < n$ implies $f(m) < f(n)$.
  - A function f is ***strictly decreasing*** if $m > n$ implies $f(m) > f(n)$.

Standard Notations and Common Functions[2]

- **Floors and Ceilings**

$$x-1<\lfloor x\rfloor\le x\le\lceil x\rceil<x+1$$

$$\lceil n/2\rceil+\lfloor n/2\rfloor=n$$

$$\lceil\lceil n/a\rceil/b\rceil=\lceil n/ab\rceil$$

$$\lfloor\lfloor n/a\rfloor/b\rfloor=\lfloor n/ab\rfloor$$

$$\lceil a/b\rceil\le(a+(b-1))/b$$

$$\lfloor a/b\rfloor\ge(a-(b-1))/b$$

Standard Notations and Common Functions[3]

Modular Arithmetic

- For any integer a and any positive integer n, the value ***a mod n*** is the ***remainder*** (or ***residue***) of the quotient a/n :

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

- If $(a \bmod n) = (b \bmod n)$. We write $a \equiv b \pmod{n}$ and say that a is equivalent to b modulo n.
- We write $a \not\equiv b \pmod{n}$ if a is not equivalent to b modulo n.

Polynomials: A polynomial in n of degree d.

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

Standard Notations and Common Functions[4]

- A function $f(n)$ is polynomially bounded if $f(n) = O(n^k)$ for some constant k.

☞ Exponentials

- The rates of growth of polynomials and exponentials:

$$\lim_{n\to\infty} \frac{n^b}{a^n} = 0, \quad \text{for } a > 1.$$

$$\therefore n^b = o(a^n)$$

Any positive exponential function grows faster than and polynomial function.

Standard Notations and Common Functions[5]

- **Let *e* denote 2.71828…, the base of the natural logarithm function, for all real *x* :**

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

$$1 + x \le e^x \le 1 + x + x^2 \quad \text{when } |x| \le 1.$$

$$\lim_{n \to \infty} \left(1 + \frac{x}{n}\right)^n = e^x, \quad \textit{for all } x.$$

Standard Notations and Common Functions[6]

Logarithms

$$\lg n = \log_2 n \quad \text{(binary logarithm)}$$

$$\ln n = \log_e n \quad \text{(natural logarithm)}$$

$$\lg^k n = (\lg n)^k \quad \text{(exponentiation)}$$

$$\lg\lg n = \lg(\lg n) \quad \text{(composition)}$$

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} - \cdots \quad \text{when } |x| < 1$$

$$\frac{x}{1+x} \le \ln(1+x) \le x, \quad x > -1.$$

Standard Notations and Common Functions[7]

- $f(n)$ is ***polylogarithmically bounded*** if $f(n) = O(\lg^k n)$ for some constant k.
- Any positive polynomial function grows faster than any polylogarithmic function.

$$\lim_{n\to\infty} \frac{\lg^b n}{\left(2^a\right)^{\lg n}} = \lim_{n\to\infty} \frac{\lg^b n}{n^a} = 0$$

$$\lg^b n = o\left(n^a\right), \quad \text{for any constant } a > 0.$$

Standard Notations and Common Functions[8]

- **Factorials**
  - A weak upper bound $n! < n^n$.
- **Stirling's approximation**

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

$$n! = o(n^n)$$

$$n! = \omega(2^n)$$

$$\log(n!) = \Theta(n \log n)$$

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n e^{\alpha_n}, \quad \text{where} \quad \frac{1}{12n+1} < \alpha_n < \frac{1}{12n} \quad \text{for all } n \geq 1.$$

Standard Notations and Common Functions[9]

☞ Functional iteration

- $f^{(i)}(n)$ denote the function f(n) iteratively applied i times to an initial value of n.

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

- Example: if f(n) = 2n, then $f^{(i)}(n) = 2^i n$.

☞ The *iterated logarithm function*: $lg^* n$.

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

Standard Notations and Common Functions[10]

- **The iterated logarithm is a very slowly growing function:**

$$lg^* 2 = 1$$

$$lg^* 4 = 2$$

$$lg^* 16 = 3$$

$$lg^* 65536 = 4$$

$$lg^* (2^{65536}) = 5$$

Standard Notations and Common Functions[11]

- **Fibonacci numbers**
  - **Each Fibonacci number is the sum of the two previous ones.**

$$F_0 = 0$$

$$F_1 = 1$$

$$F_i = F_{i-1} + F_{i-2}, \quad \text{for } i \geq 2.$$

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, …