Algorithms
Chapter 3 Growth of Functions

Associate Professor: Ching-Chi Lin
林清池 副教授
chingchi.lin@gmail.com

Department of Computer Science and Engineering
National Taiwan Ocean University

Outline

- **Asymptotic notation** 漸近式表示法
- Standard notations and common functions

The purpose of this chapter$_{1/3}$

- The order of growth of the running time of an algorithm gives us some information about: 演算法複雜度告訴我們
  - the algorithm's efficiency 演算法的效率與其他演算法的效能比較
  - the relative performance of alternative algorithms
- The merge sort, with its $\Theta(n \lg n)$ worst-case running time, beats insertion sort, whose worst-case running time is $\Theta(n^2)$.
merge sort 的效能優於 insertion sort
- For large enough inputs, the following are dominated by the effects of the input size itself. 當input size 夠大以下相對不重要
  - multiplicative constants 乘法的常數
  - lower-order terms of an exact running time 較低的項次

The purpose of this chapter$_{2/3}$

- When the input size *n* becomes large enough, we are studying the **asymptotic** efficiency of algorithms. 我們要的是演算法的漸近式效能
  漸近式
- That is, we are concerned with
  - how the running time of an algorithm increases with the size of the input **in the limit**, as the size of the input increases without bound. 當 input size 很大時，時間複雜度和 size 的關係
- Usually, an algorithm that is asymptotically more efficient will be the best choice for all but very small inputs.

通常漸近式效能較佳的演算法，在實際的效能上也較佳
（如果 size 夠大時）

	linear $5n$	quadratic $5n^2$
10	50	500
100	500	50000
10000	50000	500000000

⇒ 成長速率相差很大

The purpose of this chapter$_{3/3}$

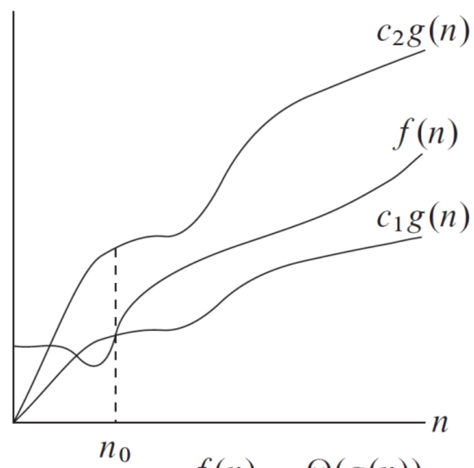
- We will study how to **measure** and **analyze** an algorithm's efficiency for large inputs.
- The next section begins by defining asymptotic notations,
  - Θ-notation 約常數倍
  - O-notation 小於等於常數倍
  - Ω-notation 大於等於常數倍

當n夠大⇔ n≥n₀ 常數倍⇔c倍

Θ-notation

重點：找 $c_1, c_2, n_0 > 0$

- For a given function $g(n)$, we denote by $\Theta(g(n))$ the set of functions {元素|元素的條件} 當n夠大時，f(n)是g(n)的常數倍
  - $\Theta(g(n)) = \{ f(n)$: 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 $n \ge n_0\}$.
    為一集合
- For $n \ge n_0$, the function $f(n)$ is equal to $g(n)$ to within a constant factor. 以漸近式的觀點，g(n)是f(n)的一個緊密界限
- Here, $g(n)$ is an **asymptotically tight bound** for $f(n)$.
- Because $\Theta(g(n))$ is a set, we could write "$f(n) \in \Theta(g(n))$".
- Usually, we write "$f(n) = \Theta(g(n))$".



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

An example proof by construction 建構法

- To show that $n^2/2 - 3n = \Theta(n^2)$, we must determine positive constants c_1, c_2, and n_0 such that

$$c_1 n^2 \le n^2/2 - 3n \le c_2 n^2 \quad \text{for all } n \ge n_0.$$

重点：找 $c_1, c_2, n_0 > 0$ $\Downarrow$ 有很多組

- Dividing by n^2 yields

$$c_1 \le 1/2 - 3/n \le c_2.$$

  - $c_1 \le 1/2 - 3/n$ holds for $n \ge 7$ by $c_1 \le 1/14$ $\Rightarrow c_1 = \frac{1}{2} - \frac{3}{n} = \frac{1}{2} - \frac{3}{7} = \frac{1}{14}$
  - $1/2 - 3/n \le c_2$ holds for $n \ge 1$ by $c_2 \ge 1/2$

- Thus, choosing $c_1 = 1/14$, $c_2 = 1/2$, and $n_0 = 7$, we can verify that $n^2/2 - 3n = \Theta(n^2)$. $n_0 = \max\{7, 1\}$, $c_1 = \frac{1}{14}$, $c_2 = \frac{1}{2}$

- Show that $3n^3 - 2 = \Theta(n^3)$.

Another example 要證不成立，先假設成立⇒產生矛盾⇒得證

- We show that $6n^3 \neq \Theta(n^2)$ by contradiction. 反證法
  - Suppose c_2 and n_0 exist such that $6n^3 \leq c_2 n^2$ for all $n \geq n_0$.
  - Then $n \leq c_2/6$, a contradiction.
  - Since c_2 is constant, it cannot possibly hold for arbitrary large n.

  因為c_2是常數，n是變數，所以不可能對任意n都成立

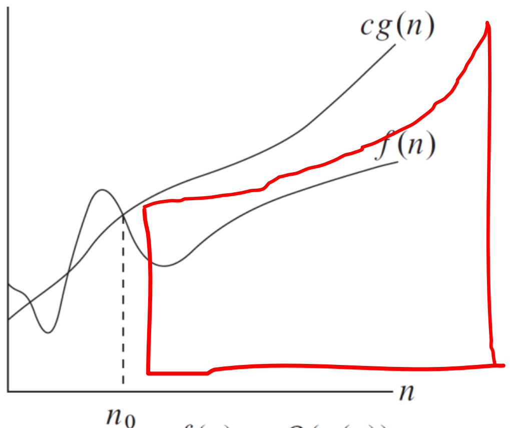
Summary 用Θ表示時

- The lower-order terms can be ignored 較低項次不重要
  - because they are insignificant for large *n*.
- The coefficient of the highest-order term can likewise be ignored 常數不重要，因為可調整 c_1、c_2
  - since it only changes c_1 and c_2 by a constant factor equal to the coefficient.
- In general, for any polynomial $p(n) = a_d n^d + \ldots + a_1 n + a_0$, where a_i are constants and $a_d > 0$, we have $p(n) = \Theta(n^d)$.
- For example, $f(n) = an^2 + bn + c$, where *a*, *b*, and *c* are constants and $a > 0$. Then, we have $f(n) = \Theta(n^2)$.

O-notation

- For a given function $g(n)$, we denote by $O(g(n))$ the set of functions $f(n) \le g(n)$ 的常數倍，對 $n \ge n_0$
  - $O(g(n)) = \{ f(n)$: there exist positive constants c and n_0 such that $0 \le f(n) \le cg(n)$ for all $n \ge n_0 \}$.
- We write $f(n) = O(g(n))$ implies $f(n)$ is a member of the set $O(g(n))$. 原本 $f(n) \in O(g(n)) \Rightarrow$ 通常 $f(n) = O(g(n))$
- Note that $f(n) = \Theta(g(n))$ implies $f(n) = O(g(n))$.
  - any proof showing that $f(n) = \Theta(g(n))$ also shows that $f(n) = O(g(n))$.
  - $\Theta(g(n)) \subseteq O(g(n))$.
- Show that $3n^2 - 2 = O(n^2)$.

O-notation 較 Θ-notation 條件寬鬆

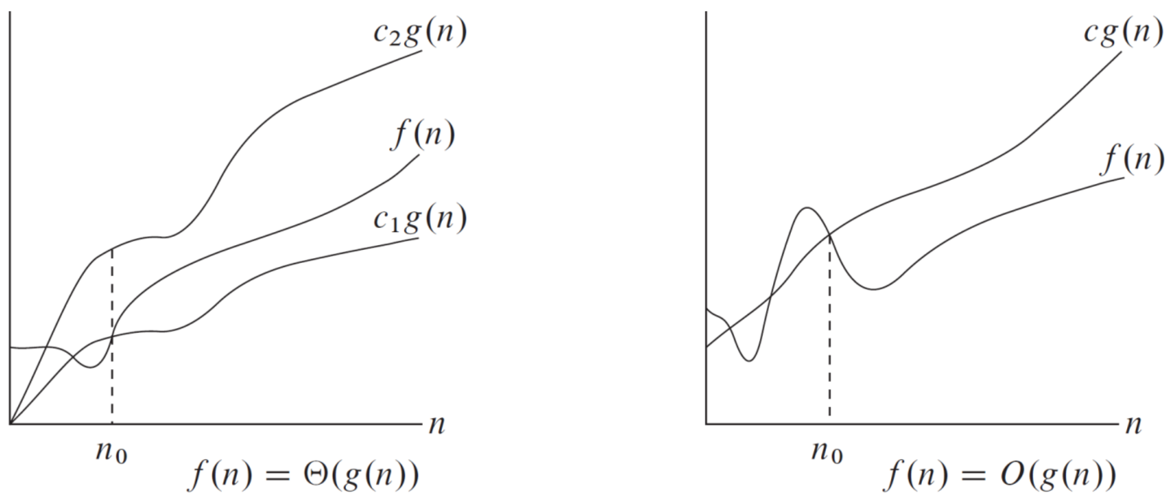


$f(n) = O(g(n))$

皆是 $f(n)$ 可落範圍

The meaning of O-notation$_{1/2}$

- The Θ-notation asymptotically bounds a function from above and below. Θ表示法給定上界和下界
- When we have only an **asymptotic upper bound**, we use O-notation. O表示法只給定上界
- Hence, Θ-notation is a stronger notation than O-notation.
  Θ-notation較O-notation強



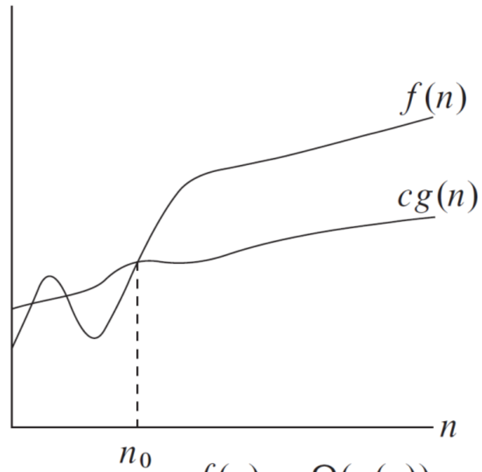
The meaning of O-notation$_{2/2}$

- Any linear function $an + b$ is in $O(n^2)$, which is easily verified by taking $c = a+|b|$ and $n_0 = 1$.
  - $an+ b \leq (a+|b|)\, n^2$ for $n \geq 1$
- $f(n) = O(g(n))$ merely claims that
  - $g(n)$ is an asymptotic **upper** bound on $f(n)$ 只說g(n)是f(n)的一個上界
  - does not claim about how tight an upper bound it is 沒說上界多緊密
- In practical, O-notation is used to describe the **worst-case** running time of an algorithm. 通常用O-notation表示演算法的最差情形
- "an algorithm is $O(g(n))$" means that
  - the running time is at most constant times $g(n)$, for sufficiently large n
  - no matter what particular input of size n is chosen for each value of n

不管輸入的input為何，時間最多為g(n)的常數倍

Ω-notation

- For a given function $g(n)$, we denote by $\Omega(g(n))$ the set of functions $f(n) \geq g(n)$的常數倍，對 $n \geq n_0$
  - $\Omega(g(n)) = \{ f(n)$: there exist positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0\}$.
- We write $f(n) = \Omega(g(n))$ implies $f(n)$ is a member of the set $\Omega(g(n))$. $f(n) \in \Omega(g(n)) \Rightarrow f(n) = \Omega(g(n))$
- Ω-notation provides **asymptotic lower bound.** 給定一個漸近式下界



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

The relationship between Θ, O, and Ω

- Theorem 3.1 多方叙述 Ex: $A \Leftrightarrow B$
  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 $f(n) = \Omega(g(n))$.
- For example:
  - $n^2/2 - 3n = \Theta(n^2) \Rightarrow n^2/2 - 3n = O(n^2)$ and $n^2/2 - 3n = \Omega(n^2)$
  - $n^2/2 - 3n = O(n^2)$ and $n^2/2 - 3n = \Omega(n^2) \Rightarrow n^2/2 - 3n = \Theta(n^2)$

Θ: 找 c_1, c_2, n_0

O: 找 c_2, n_0

Ω: 找 c_1, n_0

The meaning of Ω-notation

- The Ω-notation is used to bound the **best-case** running time of an algorithm. Ω-notation用來描述最佳情況
- "an algorithm is $\Omega(g(n))$" means that
  - the running time is at least constant times $g(n)$, for sufficiently large n
  - no matter what particular input of size n is chosen for each value of n

不管輸入的 input 為何，時間複雜度至少要 g(n) 的常數倍

o-notation 表示小於常數倍 ＊O-notation 表示小於等於常數倍

- For a given function $g(n)$, we denote by $o(g(n))$ the set of functions 對任何常數 c，都存在 n_0，使得 $0 \le f(n) < cg(n)$ for all $n \ge n_0$ 成立
  - $o(g(n)) = \{f(n)$: for **any** positive constant $c>0$, there exists a constant $n_0>0$ such that $0 \le f(n) < cg(n)$ for all $n \ge n_0\}$.
- We use o-notation to denote an upper bound that is **not** asymptotically tight. 給定一個不緊密的上界
- For example, $2n=o(n^2)$, but $2n^2 \neq o(n^2)$.
- Intuitively, the function $f(n)$ becomes insignificant relative to $g(n)$ as n approaches infinity; that is,

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0.$$ $f(n)$ 相較 $g(n)$ 顯得不重要

ω-notation 表示大於常數倍 *Ω-notation表示大於等於常數倍

- For a given function $g(n)$, we denote by $\omega(g(n))$ the set of functions 對任何常數c，都存在n_0，使得$0 \le cg(n) < f(n)$ for all $n \ge n_0$成立
  - $\omega(g(n))=\{f(n)$: for **any** positive constant $c>0$, there exists a constant $n_0>0$ such that $0 \le cg(n) < f(n)$ for all $n \ge n_0\}$.
- We use ω-notation to denote a lower bound that is **not** asymptotically tight. 給定一個不緊密的下界
- For example, $n^2/2=\omega(n)$, but $n^2/2 \ne \omega(n^2)$. 不能用於相等情形
- The relation $f(n)=\omega(g(n))$ implies that

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

f(n)的成長速度較g(n)快

if the limit exists.

Comparison of functions$_{1/4}$

- Transitivity: 遞移性
  - $f(n) = \Theta(g(n))$ and $g(n) = \Theta(h(n))$ imply $f(n) = \Theta(h(n))$,
  - $f(n) = O(g(n))$ and $g(n) = O(h(n))$ imply $f(n) = O(h(n))$,
  - $f(n) = \Omega(g(n))$ and $g(n) = \Omega(h(n))$ imply $f(n) = \Omega(h(n))$,
  - $f(n) = o(g(n))$ and $g(n) = o(h(n))$ imply $f(n) = o(h(n))$,
  - $f(n) = \omega(g(n))$ and $g(n) = \omega(h(n))$ imply $f(n) = \omega(h(n))$.

Comparison of functions$_{2/4}$

- Reflexivity: 自反性(反身性)
  - $f(n) = \Theta(f(n))$,
  - $f(n) = O(f(n))$,
  - $f(n) = \Omega(f(n))$.
- Symmetry: 對稱性
  - $f(n) = \Theta(g(n))$ if and only if $g(n) = \Theta(f(n))$.
- Transpose symmetry:
  - $f(n) = O(g(n))$ if and only if $g(n) = \Omega(f(n))$,
  - $f(n) = o(g(n))$ if and only if $g(n) = \omega(f(n))$.

Comparison of functions$_{3/4}$

- Analogy between the asymptotic comparison and the real number comparison:
  - $f(n) = \Theta(g(n)) \approx a = b.$
  - $f(n) = O(g(n)) \approx a \le b.$
  - $f(n) = \Omega(g(n)) \approx a \ge b.$
  - $f(n) = o(g(n)) \approx a < b.$
  - $f(n) = \omega(g(n)) \approx a > b.$

Comparison of functions$_{4/4}$

- Trichotomy property of real numbers does not carry over to asymptotic notation:
三一律
  - **Trichotomy:** For any two real numbers a and b, exactly one of the following must hold: $a < b$, $a = b$, or $a > b$. 三一律在漸近式表示中不存在
- Not all functions are asymptotically comparable.
  - For two functions $f(n)$ and $g(n)$, it may be the case that neither $f(n) = O(g(n))$ nor $f(n) = \Omega(g(n))$. 對於 $g(n)$ 和 $f(n)$，有可能不是 $f(n) = O(g(n))$，同時也非 $f(n) = \Omega(g(n))$
  - For example, the function n and $n^{1+\sin n}$ cannot be compared, since the value of $n^{1+\sin n}$ oscillates between 0 and 2.

Ex: n 和 $n^{1+\sin n}$ $\Rightarrow$ $\begin{cases} 不是\ n = O(n^{1+\sin n}) \\ 也非\ n = \Omega(n^{1+\sin n}) \end{cases}$

$\Rightarrow$ 因為 $0 \le 1+\sin n \le 2$，n 很大時，此兩函數還是沒有大小關係