# 第五章:演算法分析

5-1函數成長率的比較

5-2演算法的複雜度



## 5-1函數成長率的比較

# 大 O 符號定義

令 $f$ 與 $g$ 為由整數集合或實數集合對應至實數集合的函數。我們說 $f(x)$ 是 $O(g(x))$，如果存在常數 $C$ 與 $k$，使得

$$|f(x)| \le C|g(x)|$$

每當 $x > k$ 時。〔讀作 $f(x)$ 是 $g(x)$ 的大 $O$。〕

$f(x) = 2x^2 + 3x + 1, g(x) = x^2, h(x) = x^3$

(1) $f(x)$ 是 $O(g(x))$

$|f(x)| = |2x^2 + 3x + 1| \underbrace{\le}_{x>1} |2x^2 + 3x^2 + x^2| \le 6x^2 = 6|g(x)| \quad \Rightarrow C = 6, k = 1$

$(C, k$ 不唯一$)$

(2) $f(x)$ 是 $O(h(x))$

$|f(x)| = |2x^2 + 3x + 1| \underbrace{\le}_{x>1} |2x^3 + 3x^3 + x^3| \le 6x^3 = 6|h(x)| \quad \Rightarrow C = 6, k = 1$

3

# 大 O 符號的表示

注意：$f(x)$ 是 $O(g(x))$ 這個事實，有時會記為 $f(x) = O(g(x))$。只是，此處的等號並不是真正的等號。這樣的符號是在說明，當定義域中的值夠大時，關於 $f$ 與 $g$ 的不等式成立。不過，寫成 $f(x) \in O(g(x))$ 是能接受的，因為能將 $O(g(x))$ 視為所有為 $O(g(x))$ 之函數所成的集合。

(1) $f(x)$ 是 $O(g(x))$, $f(x) = O(g(x))$, 與 $f(x) \in O(g(x))$ 三者同義。

(2) $O(g(x))$ 表所有滿足存在 $k, C \in R^+$ 使得當 $x \ge k$ 時, $|f(x)| \le C|g(x)|$ 的 $f(x)$ 所成的集合。

4

# 大 O 符號 不存在的例子

證明 $n^2$不是 $O(n)$

假設存在 $k, C \in R^+$ 使得當 $n \geq k$ 時, $n^2 \leq C\,n$

即 $n \leq C, \forall n \geq k$ ，此為矛盾。

因此$n^2$不是 $O(n)$。

5

# 定理1 – 多項式函數的大 O估 計

令$f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$，其中$a_0, a_1, \ldots, a_{n-1}$，$a_n$皆為實數，則$f(x)$為 $O(x^n)$。

**證明**：利用三角不等式，若 $x > 1$，我們有

$$\begin{aligned}|f(x)| &= |a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0| \\ &\leq |a_n||x^n| + |a_{n-1}||x^{n-1}| + \cdots + |a_1||x| + |a_0| \\ &= x^n\left(|a_n| + |a_{n-1}|/x + \cdots + |a_1|/x^{n-1} + |a_0|/x^n\right) \\ &\leq x^n\left(|a_n| + |a_{n-1}| + \cdots + |a_1| + |a_0|\right).\end{aligned}$$

這些不等式證明了 $|f(x)| \leq Cx^n$,

其中，每當 $x > 1$ ，$C = |a_n| + |a_{n-1}| + \ldots + |a_1| + |a_0|$ 。

所以，$C = |a_n| + |a_{n-1}| + \ldots + |a_1| + |a_0|$ 與 $k = 1$，證明$f(x)$為 $O(x^n)$。

6

## 大 O 符號 – 例子(1),(2),(3)

(1)$1+2+\cdots+n$是 $O(n^2)$ $|1+2+\cdots+n|\le|\underbrace{n+n+\cdots+n}_{n個n}|=n^2$ $(C=1,k=1)$

(2)$n!$是 $O(n^n)$ $|n!|=|1\cdot 2\cdots\cdots n|\le|\underbrace{n\cdot n\cdots\cdots n}_{n個n}|=n^n$ $(C=1,k=1)$

(3)$\log n!$是 $O(n\log n)$ $n!\le n^n \Rightarrow \log(n!)\le\log(n^n)=n\log n$ $(C=1,k=1)$

7

## 大 O 符號 – 例子(4),(5)

(4)$n$是 $O(2^n)$

(1)當$n=1, 1\le 2^1$

(2)設$n=k, k\le 2^k$成立

當$n=k+1, k+1\le 2^k+1\le 2^k+2^k\le 2\cdot 2^k=2^{k+1}$成立

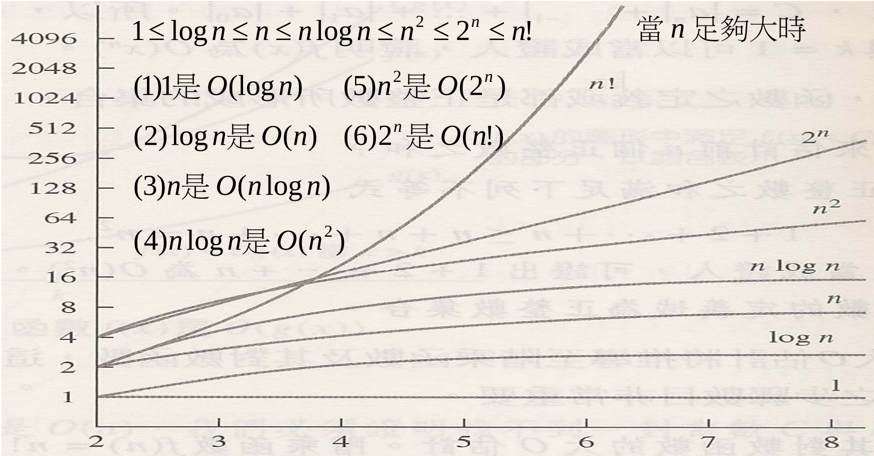
由(1),(2)數學歸納法得證 $n\le 2^n, \forall n\in N, n\ge 1$

$\Rightarrow n$是$O(2^n)$ $(C=1,k=1)$

(5)$\log n$是 $O(n)$ $n\le 2^n \Rightarrow \log(n)\le\log(2^n)=n\underbrace{\log 2}_{C}$ $(C=\log 2, k=1)$

8

## 經常用於大 O估計之函數的成長圖形




$1 \le \log n \le n \le n\log n \le n^2 \le 2^n \le n!$　當 $n$ 足夠大時

(1)1是 $O(\log n)$　(5)$n^2$是 $O(2^n)$

(2)$\log n$是 $O(n)$　(6)$2^n$是 $O(n!)$

(3)$n$是 $O(n\log n)$

(4)$n\log n$是 $O(n^2)$

經常用於大 $O$ 估計之函數的成長圖形



## 定理2 – 組合函數的大 O估 計

(1) 假設$f_1(x)$是$O(g_1(x))$，$f_2(x)$是$O(g_2(x))$，則$(f_1+f_2)(x)$是$O(\max(|g_1(x)|, |g_2(x)|))$。

(2) 假設 $f_1(x)$與$f_2(x)$都是 $O(g(x))$，則$(f_1+f_2)(x)$也是 $O(g(x))$。

(3) 假設 $f_1(x)$是 $O(g_1(x))$，$f_2(x)$是 $O(g_2(x))$，則$(f_1 f_2)(x)$是 $O(g_1(x)g_2(x))$。

求出函數 $f(x) = (x+1)\log(x^2+1)$ 的大 $O$ 估計。

$x+1 \le 2x \Rightarrow x+1$是 $O(x)$;

$\log(x^2+1) \le \log(x^2+x^2) = \log(2x^2) = \log 2 + \log x^2 = \log 2 + 2\log x \le 3\log x$

$\Rightarrow \log(x^2+1)$是 $O(\log x)$

$By(3)$, $f(x)$是 $O(x\log x)$

## 組合函數的大 O估 計 – 例子

(2) 假設 $f_1(x)$與$f_2(x)$都是 $O(g(x))$，則$(f_1+f_2)(x)$也是 $O(g(x))$。

(3) 假設 $f_1(x)$是 $O(g_1(x))$，$f_2(x)$是 $O(g_2(x))$，則$(f_1 f_2)(x)$是 $O(g_1(x)g_2(x))$。

給定一個函數 $f(n) = 3n\log(n!) + (n^2+3)\log n$，當 $n$ 為正整數時，求出其大 $O$ 估計。

$3n \le 4n \Rightarrow 3n$是 $O(n)$;  $n! \le n^n \Rightarrow \log(n!) \le \log(n^n) \Rightarrow \log(n!)$是 $O(n\log n)$

$By(3)$, $3n\log(n!)$是 $O(n^2\log n)$

$n^2+3 \le n^2+n^2 \le 2n^2 \Rightarrow n^2+3$是 $O(n^2)$;  $\log n \le \log n \Rightarrow \log n$是 $O(\log n)$

$By(3)$, $(n^2+3)\log n$是 $O(n^2\log n)$

$By(2)$, $f(n)$是 $O(n^2\log n)$

11

## 大 O 符號與演算法

大 $O$ 符號（big-$O$ notation）在估計函數之成長上有個很大的強項，我們不需要擔心常數倍數和次數較小的項數。

大$O$符號被廣泛地用來估算，當輸入量增加時，演算法使用之運算次數。基於這個緣由，我們能判斷在輸入量增加時，使用某個特殊演算法是否實際。

此外，大$O$符號也能用來判定兩種演算法中，何者較有效率。例如，利用兩種演算法同時求解一個相同的問題，若一個需要 $100n^2 + 17n + 4$ 的運算次數，而另一種需要$n^3$個運算，大$O$符號能告訴我們，在$n$很大時，第一種演算法所需要的運算次數要少得多。儘管當 $n$ 比較小時（例如，當 $n = 10$），第二種演算法用到的運算次數較少。

12

## 大 Ω 符號與大 Θ 符號的須要

### 大Ω符號與大Θ符號

雖然大$O$記號廣泛地用來描述函數的成長，但仍有其限制。比較具體地來說，當$f(x)$是$O(g(x))$時，對相當大的$x$，函數值$f(x)$能找到一個與$g(x)$有關的上界。然而，大$O$符號並不能提供當$x$相當大時，函數值$f(x)$的下界。此時，我們使用**大Ω符號**（**big Omega notation**）。若打算同時找出當$x$相當大時，函數值$f(x)$的上界與下界，則使用**大Θ符號**（**big-Theta notation**）。大Ω符號與大Θ符號都是高德納於1970年代引介的。他介紹這兩種記號的目的，是因為當同時需要知道函數的上下界時，大$O$記號經常遭到誤用。

13

## 大 Ω 符號定義

令$f$和$g$為由整數集合（或實數集合）對應到實數集合的函數。我們說$f(x)$是$\Omega(g(x))$，如果存在正的常數$C$和$k$，使得當$x > k$時，$|f(x)| \geq C|g(x)|$。〔讀作$f(x)$是$g(x)$的大Ω。〕

$f(x)$是$\Omega(g(x))$，若且唯若$g(x)$是$O(f(x))$。

$f(x) = 2x^2 + 3x + 1, g(x) = x^2, h(x) = x^3$

(1) $f(x)$是 $\Omega(g(x))$

$$|f(x)| = |2x^2 + 3x + 1| \geq 2x^2 = 2|g(x)| \qquad (C = 2, k = 1)$$

(2) $h(x)$是 $\Omega(g(x))$

$$|h(x)| = x^3 \geq x^2 = |g(x)| \qquad (C = 1, k = 1)$$

14

# 大 Θ 符號定義

令 $f$ 和 $g$ 為由整數集合（或實數集合）對應到實數集合的函數。我們說 $f(x)$ 是 $\Theta(g(x))$，如果 $f(x)$ 是 $O(g(x))$，而且同時 $f(x)$ 也是 $\Omega(g(x))$。當 $f(x)$ 是 $\Theta(g(x))$，我們說 $f(x)$ 是 $g(x)$ 的大 $\Theta$，也可以說 $f(x)$ 與 $g(x)$ 同階（$f(x)$ is of order $g(x)$）。

我們能證明 $f(x)$ 是 $\Theta(g(x))$，如果可以找到兩個正常數 $C_1$ 與 $C_2$ 和一個正整數 $k$，使得當 $x > k$ 時，$C_1|g(x)| \le |f(x)| \le C_2|g(x)|$，不等式證明了 $f(x)$ 是 $O(g(x))$ 以及 $f(x)$ 是 $\Omega(g(x))$。

$$f(x) = 2x^2 + 3x + 1, g(x) = x^2$$

$$|g(x)| = x^2 \underbrace{\le}_{x>1} |f(x)| = |2x^2 + 3x + 1| \underbrace{\le}_{x>1} |2x^2 + 3x^2 + x^2| = 6x^2 = 6|g(x)|$$

$$\Rightarrow f(x) \text{是 } \Theta(g(x))$$

15

# 定理 - 同階

令 $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$，其中 $a_0, a_1, \ldots, a_n$ 皆為實數，且 $a_n \ne 0$，則 $f(x)$ 與 $x^n$ 同階。

多項式 $3x^8 + 10x^7 + 221x^2 + 1444$、$x^{19} - 18x^4 - 10{,}112$ 和 $-x^{99} + 40{,}001x^{98} +$ $100{,}003x$ 的階數分別為

$x^8$、$x^{19}$ 和 $x^{99}$。

16

# 隨堂練習:1

(1) 證明 $x^3$ 是 $O(x^4)$，但 $x^4$ 不是 $O(x^3)$。

(2) 證明 $3x^4 + 1$ 是 $O(x^4/2)$，而且 $x^4/2$ 是 $O(3x^4 + 1)$。

17

# 隨堂練習:2

盡可能找出下列函數最好的大 $O$ 估計。

**a)** $(n^2 + 8)(n + 1)$

**b)** $(n \log n + n^2)(n^3 + 2)$

18

# 隨堂練習:3

a) 證明 $3x + 7$ 為 $\Theta(x)$。
b) 證明 $2x^2 + x - 7$ 為 $\Theta(x^2)$。

19