

1 數學符號

以下有一寫常見的數學符號與定義。不一定要全部都知道，因為這些總是可以用文字符號代替。不過熟知一些常用的符號對於書寫與閱讀數學證明有很好的幫助。

集合

這些可能高中數學就有了。

- 集合是一個可以蒐集各種東西的袋子，通常是用大括號括起來，例如說有 1, 2, 3 的集合會寫成 $\{1, 2, 3\}$ 。
- 集合不一定要裝數字！集合可以裝各種東西，例如多項式的集合 $\{x, x^5, 1 + 4x\}$ 、集合的集合 $\{\{5\}, \{1, 4\}\}$ 都可以。
- 集合有聯集 (\cup)、交集 (\cap)、跟差集 (\setminus)，高中數學可能會用減法當差集，但是好像常見的是 \setminus 或 \backslash 。
- 一個元素 x 在集合 S 裡面會寫作 $x \in S$ ，反之記為 $x \notin S$ 。
- 集合 S 的大小可以用 $\#S$ 或 $|S|$ 表示。
- 有一些常見的集合：整數 \mathbb{Z} 、有理數 \mathbb{Q} 、實數 \mathbb{R} 、複數 \mathbb{C} 。
- 自然數 \mathbb{N} 是一個數學上有歧異的名詞，為了讓所有人都不會誤會，正整數用 $\mathbb{Z}_{>0}$ 、非負整數用 $\mathbb{Z}_{\geq 0}$ 比較安全。

量詞 (quantifier)

- \forall 表示「對於所有」。例如說「對於所有正整數 n ， n 與 $n + 1$ 互質」可以寫成 $\forall n \in \mathbb{Z}_{>0}, \gcd(n, n + 1) = 1$ 。
- \exists 表示「存在一個」。例如說「在 10 到 20 中有一個 5 的倍數」可以寫成「 $\exists 10 \leq n \leq 20, 5 \mid n$ 」。
- $\exists!$ 表示「存在唯一」。與上一個的差異可以想成 \exists 是存在至少一個、 $\exists!$ 必須要存在且只有一個滿足條件。

邏輯符號

- $A \implies B$ 表示「如果 A 是真的，那 B 是真的」。例如說，「如果正整數 n 有奇數個因數 $\implies n$ 是一個完全平方數」。
- $A \longleftarrow B$ 是上面的左右相反，一般來說不會用到。

- \iff 表示「若且唯若」，很常見的英文是 if and only if（也會被簡寫成 iff），表示左右邊只會同時是對的或者是錯的。例如說，上面的敘述事實上是雙箭頭。

例題 1

這個敘述的否定敘述是什麼？

$$\forall x, y, z \in S, (x, y) \in T \text{ and } (y, z) \in T \implies (x, z) \in T$$

答案

「對於所有是對的」的反面是「存在一個是錯的」，所以這句話的反面是

$$\exists x, y, z \in S, (x, y) \in T \text{ and } (y, z) \in T, (x, z) \notin T$$

2 複雜度

複雜度想要描述的東西是什麼呢？在描述一個演算法或資料結構的各種性質時，我們想要描述這些性質的隨著資料量或者我們感興趣的變數成長時的變化。最顯著的兩個例子就是時間複雜度與空間複雜度，也就是一個演算法所需要的時間以及空間（記憶體）。

對於一個執行時間是 $5n$ 與另一個執行時間是 $14n$ 的演算法來說，他們在輸入資料大小 n 倍增時只會成長兩倍，不過對於執行時間是 $n \log n$ 的演算法雖然在 n 比較小的時候比較快，但隨著 n 的增加，有朝一日他會輸給上面兩個演算法。換句話說，在理論演算法的世界裡，我們比較會在意「成長的速度」而非演算法中的常數。

數學上最常用來估計各種成長趨勢的工具是 Big-O。他的正式定義是

$$f(x) \in O(g(x)) \text{ 如果 } \exists x_0 > 0, c > 0, \text{ 使得 } \forall x \geq x_0, |f(x)| \leq c \cdot |g(x)|。$$

在這個定義中可以看到我們允許差一個常數的被看成是相同的成長速度。不過 x_0 是用來做什麼的呢？讀者可以想像有些函數（例如 $\log \log x$ ）在 x 太小的時候是沒有好好定義的，為了避免這種邊界狀況，我們允許把一些小小的部份忽略掉。

因為這個定義是數學上最通用的定義，所以函數是對實數所定義的，而比較的時候有絕對值¹。在演算法分析上，也會使用另一組定義：

$$f(n) \in O(g(n)) \text{ 如果 } \exists n_0 > 0, c > 0, \text{ 使得 } \forall n \geq n_0, n \in \mathbb{Z}_{>0}, f(n) \leq c \cdot g(n)。$$

這個定義通常是在 $f, g \geq 0$ 而且只有在正整數上的時候（也就是一般演算法的假設），在這樣的狀況下與上面的概念是一樣的。

從上面的定義來說， $O(g(n))$ 其實是函數的集合，他蒐集所有 $f(n) \in O(g(n))$ 的 f 。正式的寫法是 $f(n) \in O(g(n))$ ，只不過因為 $f(n) = O(g(n))$ 的寫法實在是太過直覺好用，也可

¹Big-O 絕對不僅限於這兩者，而是可以拿來描述所有函數，如果讀者對於誤差分析有一點概念的話，應該會很容易了解為什麼有絕對值以及為什麼要對所有足夠大的實數都要求要滿足條件。

以寫成這個形式。要特別小心 $O(n) = O(2n)$ 的意思是集合相等（也就是兩個方向都有的意思）。

以下用一些例子讓大家更了解這個定義：

例題 2

請證明 $3n^2 = O(n^2)$ 。

證明

取 $c = 3, n_0 = 0$ 。

$$\forall n > n_0 = 0, 3n^2 \leq 3 \cdot n^2 = c \cdot n^2$$

成立，得證。

例題 3

請證明 $3n^2 \in O(n^3)$ 。

證明

取 $c = 1, n_0 = 3$ ，則

$$\forall n > n_0 = 3, 3n^2 \leq 1 \cdot n^3 = c \cdot n^3$$

成立，得證。

例題 4

請證明 $3n^2 \neq O(n)$ 。

證明

不論 c 取多少，只要 $n > \max(n_0, c)$ ， $3n^2 \leq c \cdot n$ 就不會成立，因此是取不到任何滿足條件的 c 與 n_0 ，所以說 $3n^2 \notin O(n)$ ，得證。

例題 5

請證明或否證 $f(n) = O(n^3) \iff f(n) = O((n+1)^3)$ 。

證明

對於證明雙向箭頭 $P \iff Q$ （若且唯若）的敘述，如果難以使用一些已經有的結果直接推導，另一個方法是直接分別證明 $P \implies Q$ 與 $P \impliedby Q$ 。

(\implies) 根據定義， $f(n) = O(n^3)$ 代表有某個常數 c 與正整數 n_0 滿足

$$\forall n \geq n_0, f(n) \leq c \cdot n^3$$

直接選擇 $c' = c, n'_0 = n_0$ ，那麼

$$\forall n \geq n'_0, f(n) \leq c'n^3 = cn^3 < c(n+1)^3$$

所以再根據定義， $f(n) = O((n+1)^3)$ 。

(\Leftarrow) 根據定義， $f(n) = O((n+1)^3)$ 代表

$$\exists c > 0, n_0 \geq 0, \text{使得 } \forall n \geq n_0, f(n) \leq c(n+1)^3$$

選擇 $c' = 8c, n'_0 = n_0$ ，那麼

$$\begin{aligned} \forall n \geq n'_0, f(n) &\leq c(n+1)^3 = cn^3 + 3cn^2 + 3cn + cn^3 \\ &\leq cn^3 + 3cn^3 + 3cn^3 + cn^3 = 8cn^3 = c'n^3 \end{aligned}$$

根據定義， $f(n) = O(n^3)$ 。

由上述兩點，可以得到 $f(n) = O(n^3) \iff f(n) = O((n+1)^3)$ 。

以上可以發現，要說 $n^2 = O(n^3)$ 是可以的，因為 n^3 的確是 n^2 的上界，只是他是個比較鬆的上界。另外， $O(n+1), O(2n)$ 等等的寫法也都是可以的，只是一般來說都會寫成等價的 $O(n)$ 。

例題 6

請證明或否證 $f(n) = O(2^n) \iff f(n) = O(3^n)$ 。

證明

如果我們直接拿上一個例題的方式來解這個問題，會發現證明途中遇到阻礙，實際上這個敘述是錯誤的，這時候我們可以考慮反證法：

讓 $f(n) = 3^n$ ，假設我們真的找到一個 n_0 以及 c 滿足

$$\forall n \geq n_0, 3^n \leq c \cdot 2^n$$

觀察到右邊的式子是

$$\left(\frac{3}{2}\right)^n \leq c$$

當 $n = \max\left(n_0, \log_{\frac{3}{2}} c\right) + 1$ 的時候，上面的式子不成立。所以對於這個 $f(n)$ ，是不可能找到 c, n_0 滿足 Big-O 的定義的。

只要一個反例就足夠反駁了，所以這個敘述是錯的。

那麼，影片中說的取極限又是怎麼一回事呢？極限是一個很強大的工具，不過使用極限有很多很多需要注意的條件。對於已經知道極限的讀者，可以參考下面的額外內容。

$$\begin{aligned}
 f(n) \in O(g(n)) &\iff \exists c > 0, \exists n_0 > 0, \forall n \geq n_0, f(n) \leq c \cdot g(n) \\
 &\iff \exists c > 0, \exists n_0 > 0, \forall n \geq n_0, \frac{f(n)}{g(n)} \leq c
 \end{aligned}$$

我們不知道 n_0 是多少，但是一定是個常數，那麼把 n 趨近到無限大就一定會超過 n_0 了！於是就得到

$$f(n) \in O(g(n)) \iff \exists k > 0, \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} \leq k$$

也就是說，只要 $\frac{f(n)}{g(n)}$ 的極限值是一個常數，則 $f(n) \in O(g(n))$ ，而這就是影片中講解的判斷方法了。

為什麼上面的敘述沒有雙向箭頭呢？其中一件事是當 $k = 0$ 的時候也對，而第二件事是因為極限可以不存在，不過在發散到無窮大的時候足夠反駁上面的敘述，所以其實還有（對於值域是正的函數）：

$$f(n) \in O(g(n)) \iff \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

$$f(n) \notin O(g(n)) \iff \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$$

單純的極限其實不容易證明某個東西是某個複雜度，不過在證否的時候相當容易派上用場。

習題

以下問題均**不得使用極限**作答。

1. 請回答以下問題：

(a) (10 pts) 找到最小的 k 滿足 $n^5 + 14n = O(n^k)$ 。

(b) (10 pts) 找到最小的 k 滿足 $n\sqrt{n} + n \log^2 n = O(n^k)$ 。

(c) (10 pts) 以下的敘述是不是對的？請證明你的答案。

$$f(n) = O(n!) \implies f(n) = O((n+1)!)$$

(d) (10 pts) 以下的敘述是不是對的？請證明你的答案。

$$2^n = O(n)$$

(e) (10 pts) 以下的敘述是不是對的？請證明你的答案。

$$f(n) = O(n) \implies 2^{f(n)} = O(2^n)$$

2. (10 pts) 有一個函數 $T(n) : \mathbb{Z}_{>0} \rightarrow \mathbb{Z}$ 使用下列的遞迴定義：

$$T(n) = \begin{cases} 1 & \text{如果 } n = 1 \\ 2T(\lfloor \frac{n}{2} \rfloor) + n^2 & \text{其他狀況} \end{cases}$$

證明 $T(n) = O(n^2)$ 。

3. 有一個函數 $T(n) : \mathbb{Z}_{>0} \rightarrow \mathbb{Z}$ 使用下列的遞迴定義：

$$T(n) = \begin{cases} 1 & \text{如果 } n = 1 \\ 2T(\lfloor \frac{n}{2} \rfloor) + 2n & \text{其他狀況} \end{cases}$$

(a) (10 pts) 證明所有 $n = 2^m$ ，而 $m \in \mathbb{Z}_{>0}$ ， $T(n) \leq 3n \log_2 n$ 。

(b) (10 pts) 證明 $T(n) = O(n \log n)$ 。

(c) (20 pts) 如果遞迴的時候改成定義（不同之處在取上高斯）：

$$T(n) = \begin{cases} 1 & \text{如果 } n = 1 \\ 2T(\lceil \frac{n}{2} \rceil) + 2n & \text{其他狀況} \end{cases}$$

$T(n) = O(n \log n)$ 還是對的嗎？請證明你的答案。

4. (20 pts) 有一個函數 $T(n) : \mathbb{Z}_{>0} \rightarrow \mathbb{Z}$ 使用下列的遞迴定義：

$$T(n) = \begin{cases} 1 & \text{如果 } n = 1 \\ 2T(\lfloor \frac{n}{2} \rfloor) + \log_2 n & \text{其他狀況} \end{cases}$$

$T(n) = O(n)$ 是正確的嗎？請證明你的答案。

提示：對於遞迴的各種函數，不妨嘗試數學歸納法。

以下的題目不計分、不須繳交為作業，不過有興趣的學員可以自己嘗試。

5. 請回答以下問題：

(a) (0 pts) 以下的敘述是不是對的？請證明你的答案。

$$2^{\sqrt{\log_2 n}} = O(\sqrt{n})$$

(b) (0 pts) 以下的敘述是不是對的？請證明你的答案。

$$3^{(2^n)} = O(2^{(3^n)})$$

(c) (0 pts) 以下的敘述是不是對的？請證明你的答案。

$$n^{\log_2 n} = O(2^n)$$

6. (0 pts) 有一個函數 $T(n) : \mathbb{Z}_{>0} \rightarrow \mathbb{Z}$ 使用下列的遞迴定義：

$$T(n) = \begin{cases} 1 & \text{如果 } n = 1 \\ \sqrt{n} \cdot T(\lfloor \sqrt{n} \rfloor) + n & \text{其他狀況} \end{cases}$$

證明 $T(n) = O(n \log \log n)$ 。