



# 演算法複雜度分析

# 這堂課預計上課重點

- 演算法複雜度介紹
- 漸近符號 (Asymptotic Notation)
- 常見分析方法

# 演算法複雜度介紹

- 空間複雜度：演算法所需額外記憶體
- 時間複雜度：演算法執行時間與輸入規模  $n$  的關係
- 常見時間複雜度： $O(1)$ ,  $O(\log n)$ ,  $O(n)$ ,  $O(n \log n)$ ,  $O(n^2)$ ,  $O(2^n)$ ...

# 漸進符號概念

- 一組數學符號，用於描述一個函式的漸近行為，特別是當其變數趨近於無窮大時的成長速度或範圍
- 使用漸進符號來描述演算法在輸入規模  $n \rightarrow \infty$  的成長率
- 常見符號：
  - $O$  : Asymptotically Upper Bound (上界)
  - $\Omega$  : Asymptotically Lower Bound (下界)
  - $\Theta$  : Asymptotically Tight Bound (緊界)
  - $o$  : Strict Upper Bound (嚴格上界)
  - $\omega$  : Strict Lower Bound (嚴格下界)

## 為何使用漸近符號

- 忽略硬體與編譯器差異
  - 不同環境下的實際執行時間可能不同，但漸近分析專注於成長趨勢。
- 忽略常數與低階項
  - 只關注輸入規模  $n$  很大時的主要影響因素。
- 統一比較標準
  - 可以跨演算法、跨平台比較效率。
- 強調 Scalability
  - 演算法在大規模輸入下的表現更具參考價值。

# 漸進符號定義

□ 假設 2 個函數  $f(n)$  與  $g(n)$

□  $n$  是演算法的輸入規模，規模最常使用的的方式是輸入個數

□  $f(n)$  是”預計要評估的演算法”之時間或空間函數

□  $g(n)$  是”基準演算法”之時間或空間函數

Ex. 5筆資料輸入 ( $n=5$ )

$n$  筆輸入



演算法 A



正確輸出

花費  $f(n)$  時間

Ex. 花 20分鐘 ( $f(5)=20$ )

Ex. 5筆資料輸入 ( $n=5$ )

$n$  筆輸入



基準演算法



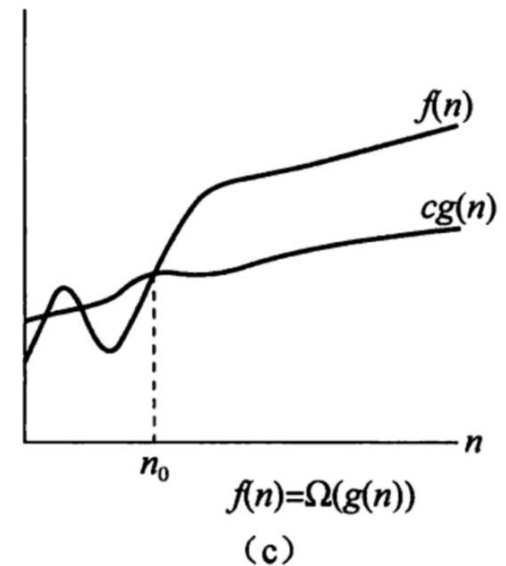
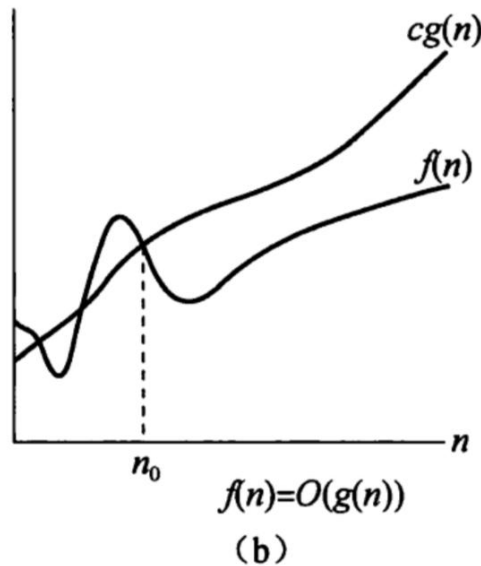
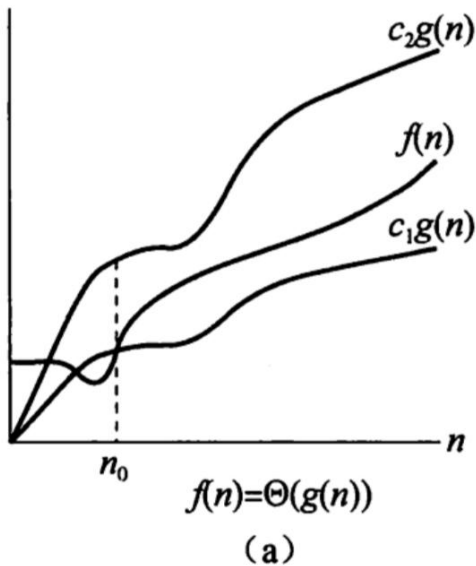
正確輸出

花費  $g(n)$  時間

Ex. 花 10分鐘 ( $g(5)=10$ )

# 漸進符號定義

- Asymptotically : 當  $n$  大於  $n_0$  之後,  $f(n)$  的成長速度和  $g(n)$  一樣快
  - $O(g(n))$  : 存在  $c, n_0$ , 使得  $f(n) \leq cg(n)$ ,  $\forall n \geq n_0$
  - $\Omega(g(n))$  : 存在  $c, n_0$ , 使得  $f(n) \geq cg(n)$ ,  $\forall n \geq n_0$
  - $\Theta(g(n))$  : 存在  $c_1, c_2, n_0$ , 使得  $c_1g(n) \leq f(n) \leq c_2g(n)$ ,  $\forall n \geq n_0$
- Strict : 當  $n$  大於  $n_0$  之後,  $g(n)$  的成長速度一定比  $f(n)$  快
  - $o(g(n))$  : 對所有  $c > 0$ , 存在  $n_0$ , 使得  $f(n) < c \cdot g(n)$ ,  $\forall n \geq n_0$
  - $\omega(g(n))$  : 對所有  $c > 0$ , 存在  $n_0$ , 使得  $f(n) > c \cdot g(n)$ ,  $\forall n \geq n_0$



# 衍生概念 1

□ 函數的 Degree 是函數的最高次項之指數

□  $f(n) = 31n^2 + 5n + 7 \rightarrow f(n)$  的 Degree = 2

□ 考慮  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = L$

□  $L = 0 \rightarrow g(n)$  的 Degree  $> f(n)$  的 Degree  $\rightarrow f(n) = o(g(n)) \rightarrow f(n) = O(g(n))$

□  $L > 0 \rightarrow g(n)$  的 Degree =  $f(n)$  的 Degree  $\rightarrow f(n) = \Theta(g(n))$

□  $L = \infty \rightarrow g(n)$  的 Degree  $< f(n)$  的 Degree  $\rightarrow f(n) = \omega(g(n)) \rightarrow f(n) = \Omega(g(n))$

□ 函數規模由小到大排列



## 衍生概念 2

### □ 函數規模

小 ↓          大	□ $O(1)$	
	□ $O(\sqrt{n})$	
	□ $O(n)$	
	□ $O(n \log \log n)$	→ $n \log_2(\log_2 n)$
	□ $O(n \log n)$	
	□ $O(n \log^2 n)$	→ $n(\log_2 n)(\log_2 n)$
	□ $O(n^{1.5})$	
	□ $O(n^2)$	
	□ $O((\log n)!)$	→ $1 \times 2 \times 3 \times \cdots \times (\log_2 n)$
	□ $O(2^n)$	
	□ $O(n!)$	→ $1 \times 2 \times 3 \times \cdots \times (n-1) \times n$

## 常用公式複習

### □ 算術級數(等差級數, Arithmetic Series)

□ 任何相鄰兩項的差相等，該差值稱為公差 (common difference,  $d$ )

□ 等差數列第  $n$  項  $a_n$  的一般項為

$$a_n = a_1 + (n - 1)d$$

□  $n$  項求和

$$\begin{aligned} S_n &= \frac{n}{2} (a + a_n) \\ &= \frac{n}{2} [2a + (n - 1)d] \\ &= an + d \cdot \frac{n(n - 1)}{2} \end{aligned}$$

## 常用公式複習

### □ 幾何級數(等比級數, Geometric Series)

□ 將一個公比  $r$  恆定的等比數列的各項連接起來所形成的級數，

□ 形式:  $a + ar + ar^2 + \dots + ar^n + \dots$

□ 有窮幾何級數 (Finite geometric series)

→ 前  $n$  項的和  $S_n = a(1 - r^n) / (1 - r)$ 。

□ 無窮幾何級數

→ 當公比  $r$  的絕對值  $|r| < 1$  時，無窮幾何級數的總和  $S = a / (1 - r)$ 。

# 常用公式複習

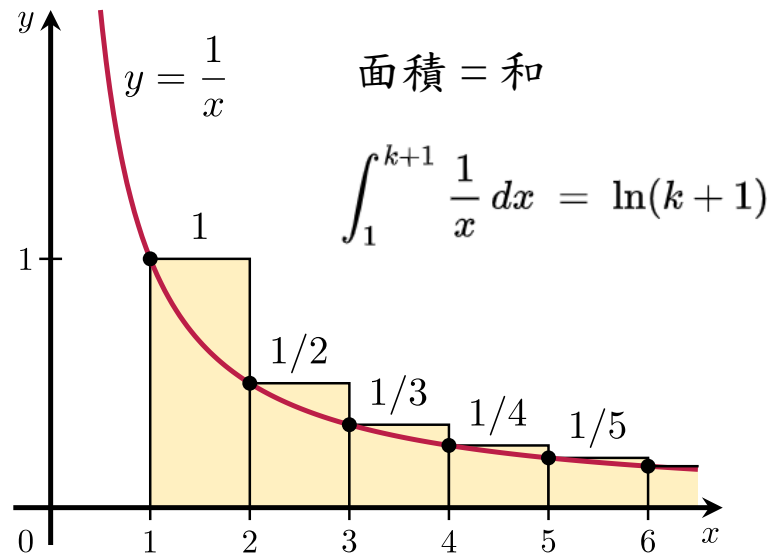
## □ 調和級數 (Harmonic Series)

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots$$

$$\sum_{n=1}^k \frac{1}{n} = \ln k + \gamma + \varepsilon_k$$

$\nearrow \frac{1}{2k}$

歐拉-馬歇羅尼常數  
0.5777.....



# 常用公式複習

## □ 對數基本公式

- \* 1.  $\log_a 1 = 0 \cdot \log_a a = 1$
- \* 2.  $\log_a a^x = x \cdot a^{\log_a y} = y$
- \* 3.  $\log_a xy = \log_a x + \log_a y$
- \* 4.  $\log_a \frac{y}{x} = \log_a y - \log_a x$
- \* 5.  $\log_a x^k = k \log_a x$
- \* 6. 換底公式： $\log_a b = \frac{\log_x b}{\log_x a}$
- \* 7. 鎖鏈公式： $(\log_a b)(\log_b c) = \log_a c$
- \* 8.  $a^{\log b} = b^{\log a}$

## 常見分析方法

- 遞迴樹法 (Recursion-Tree Method)：將遞迴展開為樹，計算成本總和
- Master Method：快速解 divide-and-conquer 遞迴式
- 迭代法：展開成總和式，求閉合解
- 替代法：猜解答 + 數學歸納證明

# 遞迴樹法 (Recursion-Tree Method)

## □ 步驟

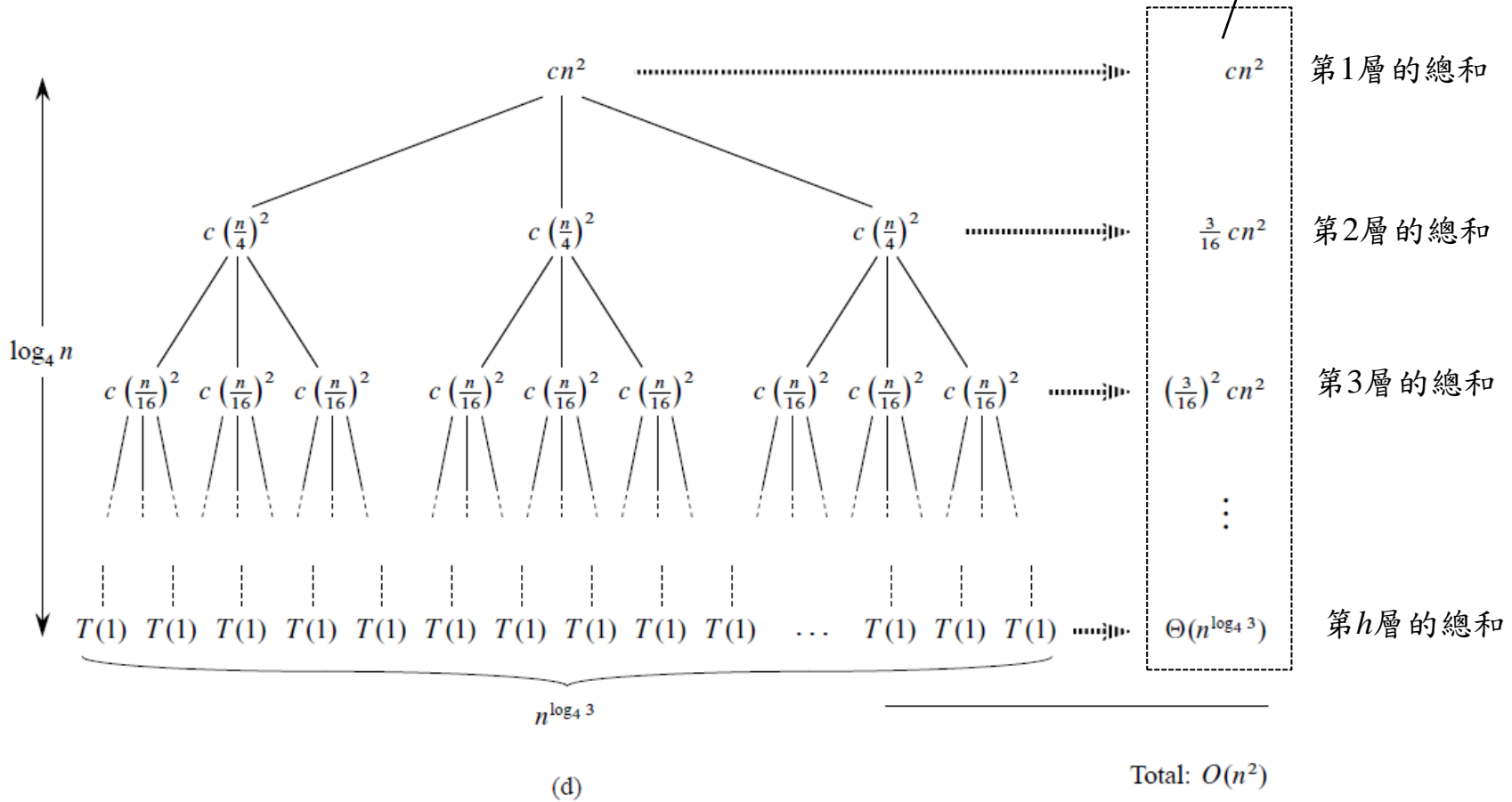
- 依照遞迴定義展開
- 對樹上的每一層進行加總，求得每一層的總和
- 找除層層總和的規律，加總”每一層的總和”，得到最終結果

## 舉例

等比級數  $S = a / (1 - r)$

$r = 3/16$ ,  $a = cn^2$

$$T(n) = 3T(n/4) + cn^2 \Rightarrow \text{求 } T(n) = O(?)$$





## 練習

□  $T(n) = T(n/5) + T(7n/10) + n$  求  $T(n) = O(?)$  (90、91 台大資工)

# Divide-and-Conquer

- 將一個問題切割成數個很直覺被解決的小問題，每個小問題解決完後再將結果進行合併即完成解決
- 三大步驟
  - 分割 (Divide)：將問題切割成數個子問題 (Subproblem)。
  - 擊破 (Conquer)：每個子問題以遞迴(Recursively)的方式解決之，直到每個問題都能夠很直覺被解決。
  - 合併(Combine)：將所有子問題的結果合併。

The diagram illustrates the recurrence relation for Divide-and-Conquer with the following annotations:

- $\Theta(1)$ : 處理一個輸入規模小於  $c$  的子問題所需要的執行時間
- $aT(n/b)$ : 處理一個輸入規模為  $\frac{n}{b}$  子問題所需要的執行時間
- $D(n)$ : 切割子問題所需要的執行時間
- $C(n)$ : 合併子問題執行結果所需要的執行時間
- $a$ : 每次切割成  $a$  個子問題

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c, \\ aT(n/b) + D(n) + C(n) & \text{otherwise.} \end{cases}$$

# Master Method

處理一個輸入規模小於  $c$  的子問題所需要的執行時間

處理一個輸入規模為  $\frac{n}{b}$  子問題所需要的執行時間

每次切割成  $a$  個子問題

切割子問題所需要的執行時間

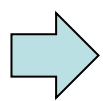
合併子問題執行結果所需要的執行時間

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq c, \\ aT(n/b) + D(n) + C(n) & \text{otherwise.} \end{cases}$$

$$T(n) = aT(n/b) + f(n)$$

$a \geq 1$  and  $b > 1$

1. If  $f(n) = O(n^{\log_b a - \epsilon})$  for some constant  $\epsilon > 0$ , then  $T(n) = \Theta(n^{\log_b a})$ .
2. If  $f(n) = \Theta(n^{\log_b a})$ , then  $T(n) = \Theta(n^{\log_b a} \lg n)$ .
3. If  $f(n) = \Omega(n^{\log_b a + \epsilon})$  for some constant  $\epsilon > 0$ , and if  $af(n/b) \leq cf(n)$  for some constant  $c < 1$  and all sufficiently large  $n$ , then  $T(n) = \Theta(f(n))$ . ■



Step 1. 計算  $\log_b a$

Step 2. 比較  $\log_b a$  和  $f(n)$ ，若不等於需要找  $\epsilon$

# Master Method

- $T(n) = 7T(n/2) + n^2$  (91 交大資工)
- $T(n) = 4T(n/2) + n^2$  (92 交大資工)
- $T(n) = 3T(n/2) + n^2$  (92 交大資工)

## 小結

□ 如果遇到時間函數符合下列公式時

$$T(n) = aT(n/b) + f(n) \quad a \geq 1 \text{ and } b > 1$$

□ 解決方式:

□ Master Method → 需判定大小關係和找  $\epsilon$  → 套公式

□ Recursive Tree → 利用圖形方式找規律 → 費工但簡單