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 $n \to \infty$ 的成長率
  - 常見符號：
    - O : Asymptotically Upper Bound (上界)
    - Ω : Asymptotically Lower Bound (下界)
    - Θ : Asymptotically Tight Bound (緊界)
    - o : Strict Upper Bound (嚴格上界)
    - ω : 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

↓

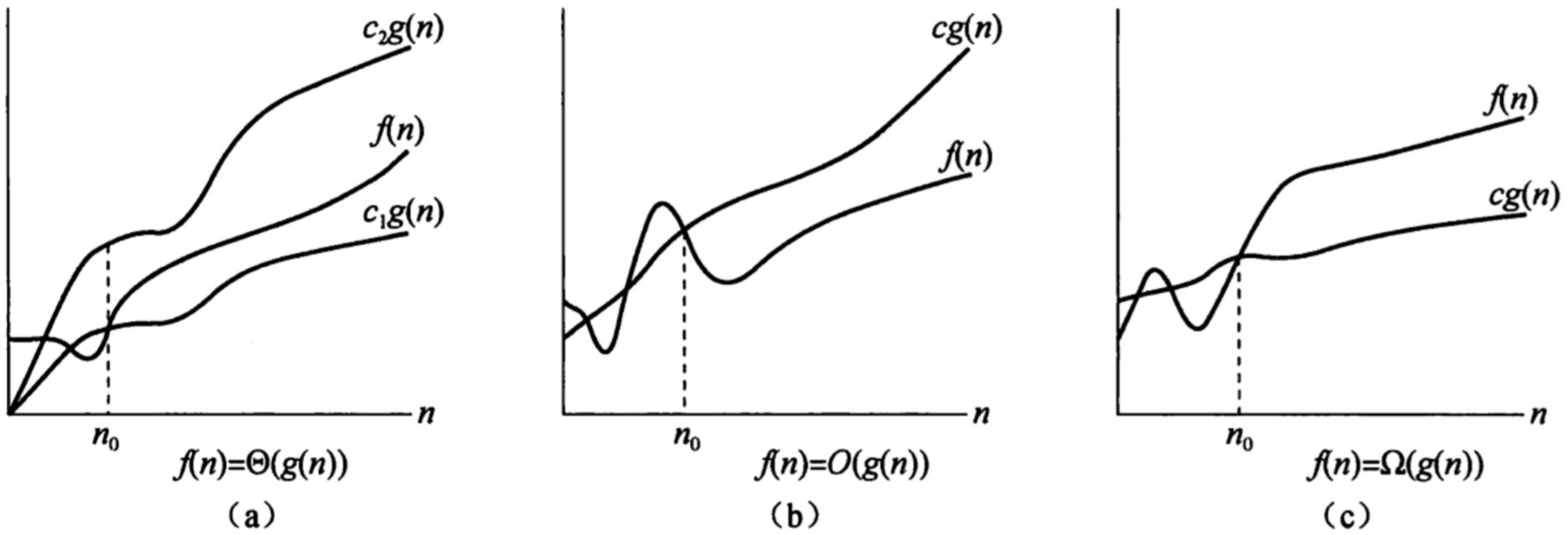
正確輸出

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 $f(n) \le cg(n)$，$\forall\ n \ge n_0$
  - $\Omega(g(n))$：存在 c, n_0，使得 $f(n) \ge cg(n)$，$\forall\ n \ge n_0$
  - $\Theta(g(n))$：存在 c_1, c_2, n_0，使得 $c_1g(n) \le f(n) \le c_2g(n)$ ，$\forall\ n \ge n_0$
- 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 $f(n) < c \cdot g(n)$，$\forall\ n \ge n_0$
  - $\omega(g(n))$：對所有 $c > 0$，存在 n_0，使得 $f(n) > c \cdot g(n)$，$\forall\ n \ge n_0$



衍生概念 1

- 函數的 Degree 是函數的最高次項之指數
  - $f(n) = 31n^2+5n+7$ ➔ $f(n)$ 的 Degree = 2

- 考慮 $\lim_{n\to\infty}\frac{f(n)}{g(n)} = L$
  - $L = 0$ ➔ $g(n)$ 的Degree > $f(n)$的Degree ➔ $f(n) = o(g(n))$ ➔ $f(n) = O(g(n))$
  - $L > 0$ ➔ $g(n)$ 的Degree = $f(n)$的Degree ➔ $f(n) = \Theta(g(n))$
  - $L = \infty$ ➔ $g(n)$ 的Degree < $f(n)$的Degree ➔ $f(n) = \omega(g(n))$ ➔ $f(n) = \Omega(g(n))$

- 函數規模由小到大排列

衍生概念 2

- 函數規模

小

- O(1)
- $O(\sqrt{n})$
- $O(n)$
- $O(n\log\log n)$ → $n\log_2(\log_2 n)$
- $O(n\log n)$
- $O(n\log^2 n)$ → $n(\log_2 n)(\log_2 n)$
- $O(n^{1.5})$
- $O(n^2)$
- $O((\log n)!)$ → $1\times2\times3\times\cdots\times(\log_2 n)$
- $O(2^n)$
- $O(n!)$ → $1\times2\times3\times\cdots\times(n-1)\times n$

大

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: $a + ar + ar^2 + \ldots + ar^n + \ldots$
  - 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

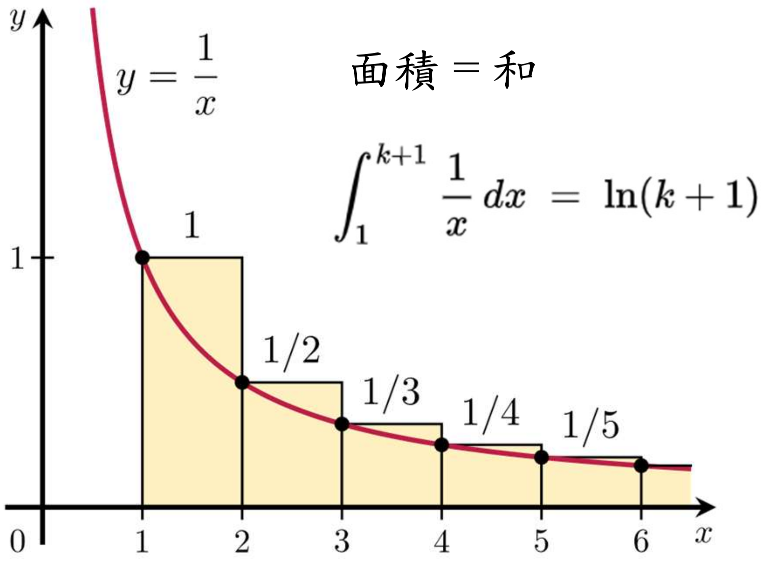
- 調和級數 (Harmonic Series)

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

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

$\frac{1}{2k}$

歐拉-馬歇羅尼常數
0.5777…..



常用公式複習

□ 對數基本公式

* 1. $\log_a 1 = 0$ · $\log_a a = 1$

* 2. $\log_a a^x = x$ · $a^{\log_a y} = y$

* 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)

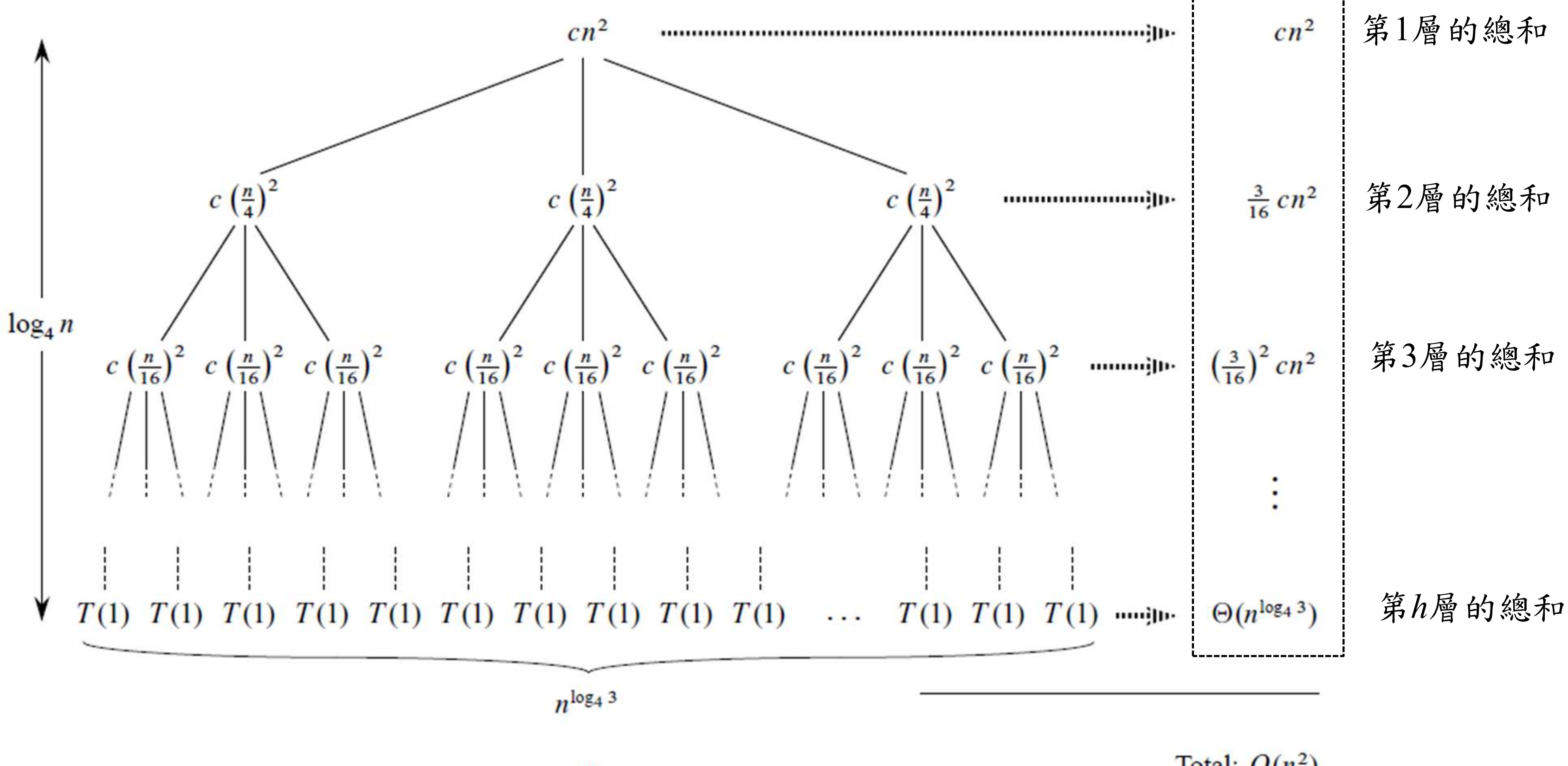
- 步驟
  - 依照遞迴定義展開
  - 對樹上的每一層進行加總，求得每一層的總和
  - 找除層層總和的規律，加總”每一層的總和”，得到最終結果

舉例

等比級數 $S = a / (1 - r)$

$r = 3/16$， $a = cn^2$

$$T(n) = 3T(n/4) + cn^2$$ ⇨ 求 $T(n) = O(?)$



(d)