18. Sorting Algorithms

2024資訊研究社 語法班
Made with ❤️ by jheanlee

Big O 複雜度

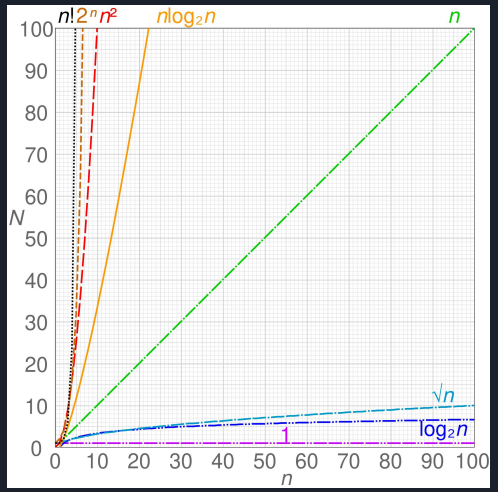
為了標示演算法的效率，我們可以分析程式的Big O複雜度 (通常為 資料數n對複雜度f(n))
會簡化到最顯著層級

常見時間複雜度列表 [編輯]

以下表格統整了一些常用的時間複雜度類別。表中，poly(x) = $x^{O(1)}$，也就是 x 的多項式。

名稱	複雜度類別	執行時間（$T(n)$）	執行時間舉例	演算法舉例
常數時間		$O(1)$	10	判斷一個二進制數的奇偶
反阿克曼時間		$O(\alpha(n))$		併查集的單個操作的平攤時間
迭代對數時間		$O(\log^* n)$		分散式圓環著色問題
對數對數時間		$O(\log\log n)$		有界優先佇列的單個操作[1]
對數時間	DLOGTIME	$O(\log n)$	$\log n$，$\log n^2$	二分搜尋
冪對數時間		$(\log n)^{O(1)}$	$(\log n)^2$	
（小於1次）冪時間		$O(n^c)$，其中 $0 < c < 1$	$n^{\frac{1}{2}}$，$n^{\frac{2}{3}}$	K–d樹的搜尋操作
線性時間		$O(n)$	n	無序陣列的搜尋
線性迭代對數時間		$O(n \log^* n)$		萊姆德·賽德爾的三角分割多邊形演算法
線性對數時間		$O(n \log n)$	$n \log n$，$\log n!$	最快的比較排序
二次時間		$O(n^2)$	n^2	泡沫排序、插入排序
三次時間		$O(n^3)$	n^3	矩陣乘法的基本實現，計算部分相關性
多項式時間	P	$2^{O(\log n)} = n^{O(1)}$	n，$n \log n$，n^{10}	線性規劃中的卡馬卡演算法，AKS質數測試

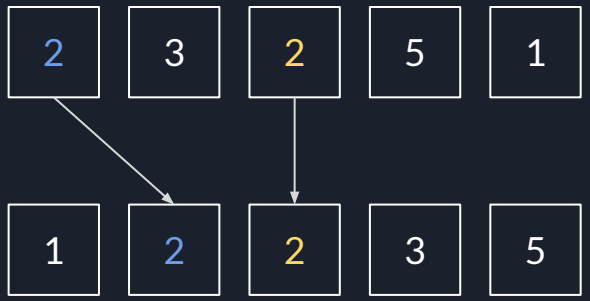
中文維基百科 - 時間複雜度
https://zh.wikipedia.org/zh-tw/%E6%97%B6%E9%97%B4%E5%A4%8D%E6%9D%82%E5%BA%A6



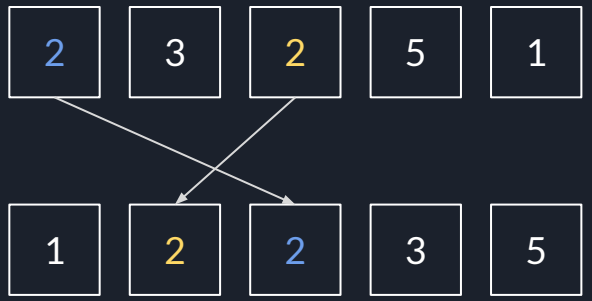
English Wikipedia - Big O notation
https://en.wikipedia.org/wiki/Big_O_notation

排序法的穩定性

排序演算法的討論中包含一個重要項目：穩定性(stability)，指具有相同值或計算值的元素間的相對順序是否改變



穩定



不穩定

Selection Sort

1. 找出最小的元素
2. 將他與最前面的元素交換
3. 重複

時間複雜度: Ω(n^2)/Θ(n^2)/O(n^2)

空間複雜度: Ω(1)/Θ(1)/O(1)

Bubble Sort

1. 每輪的各個元素與下個元素比大小，若此元素比下一個元素大就交換位置，持續到最後一個未排序的元素
2. 此時最大的元素已「浮」至最後
3. 重複直到一輪中完全沒有交換位置

時間複雜度: Ω(n)/Θ(n^2)/O(n^2)

空間複雜度: Ω(1)/Θ(1)/O(1)

Merge Sort

1. 每一次都將資料分成兩群
2. 不斷重複直到資料只有一個
3. 將資料比大小後複寫

- 可以多執行緒(multi-threading)同時處理各段

時間複雜度: Ω(n log n)/Θ(n log n)/O(n log n)

空間複雜度: Ω(n)/Θ(n)/O(n)

Quick Sort

1. 選出一個支點(pivot)
2. 依支點大小將資料分為大小兩群(使用two pointer技巧)
3. 重複直到剩下一個元素-> 回傳

- 一般狀況下QuickSort較Merge Sort快

時間複雜度: Ω(n log n)/Θ(n log n)/O(n^2)

空間複雜度: Ω(log n)/Θ(log n)/O(n) (遞迴堆疊)

其他常見演算法

Introsort - 大部分C++ 的 std::sort使用的演算法

In-Place Merge Sort - 穩定版本的Merge Sort，含有其特性而可以多執行緒同時處理

Shellsort - 程式碼較短，常見於嵌入式(embedded)裝置

Heapsort - 最差時間複雜度較Quicksort好 (O(log n)) 但平均較慢

Bogosort - 使用隨機排序 最佳Ω(n) 最差O(∞)時間複雜度

Divine Sort- ~~在神的世界中為O(1) 時間及空間複雜度~~