排序算法

维基百科，自由的百科全书
跳到导航 跳到搜索

> 此條目**没有列出任何参考或来源**。 *(2013年11月10日)*
> 維基百科所有的內容都應該可供查證。请协助添加来自可靠来源的引用以改善这篇条目。无法查证的內容可能被提出异议而移除。

在計算機科學與數學中，一個**排序算法**（英語：Sorting algorithm）是一種能將一串資料依照特定排序方式进行排列的一種算法。最常用到的排序方式是數值順序以及字典順序。有效的排序算法在一些算法（例如搜尋算法與合併算法（英语：Merge algorithm））中是重要的，如此這些算法才能得到正確解答。排序算法也用在處理文字資料以及產生人類可讀的輸出結果。基本上，排序算法的輸出必須遵守下列兩個原則：

1. 輸出結果為遞增序列（遞增是針對所需的排序順序而言）
2. 輸出結果是原輸入的一種排列、或是重組

雖然排序算法是一個簡單的問題，但是從計算機科學發展以來，在此問題上已經有大量的研究。舉例而言，泡沫排序在1956年就已經被研究。雖然大部分人認為這是一個已經被解決的問題，有用的新算法仍在不斷的被發明。（例子：圖書館排序在2004年被發表）

目录

分類

在计算机科学所使用的排序算法通常被分類為：

- 計算的時間複雜度（最差、平均、和最好性能），依據串列（list）的大小( )。一般而言，好的性能是 (大O符号)，壞的性能是 。對於一個排序理想的性能是 ，但平均而言不可能達到。基於比較的排序算法對大多數輸入而言至少需要 。
- 內存使用量（以及其他電腦資源的使用）
- 穩定性：**穩定排序算法**會讓原本有相等鍵值的紀錄維持相對次序。也就是如果一個排序算法是**穩定**的，當有兩個相等鍵值的紀錄 和 ，且在原本的串列中 出現在 之前，在排序過的

串列中 也將會是在 之前。
- 依據排序的方法：插入、交換、選擇、合併等等。

穩定性

當相等的元素是無法分辨的，比如像是整數，穩定性並不是一個問題。然而，假設以下的數對將要以他們的第一個數字來排序。

在這個狀況下，有可能產生兩種不同的結果，一個是讓相等鍵值的紀錄維持相對的次序，而另外一個則沒有：

（維持次序）
（次序被改變）

不穩定排序算法可能會在相等的鍵值中改變紀錄的相對次序，但是穩定排序算法從來不會如此。不穩定排序算法可以被特別地實作為穩定。作這件事情的一個方式是人工擴充鍵值的比較，如此在其他方面相同鍵值的兩個物件間之比較，（比如上面的比较中加入第二个标准：第二个键值的大小）就會被決定使用在原先資料次序中的條目，當作一個同分決賽。然而，要記住這種次序通常牽涉到額外的空間負擔。

排序算法列表

在這個表格中， 是要被排序的紀錄數量以及 是不同鍵值的數量。

穩定的排序

- 冒泡排序（bubble sort）—
- 插入排序（insertion sort）—
- 鸡尾酒排序（cocktail sort）—
- 桶排序（bucket sort）— ；需要 額外空間
- 计数排序（counting sort）— ；需要 額外空間
- 归并排序（merge sort）— ；需要 額外空間
- 原地归并排序— 如果使用最佳的現在版本
- 二叉排序树排序（binary tree sort）— 期望时间； 最坏时间；需要 額外空間
- 鸽巢排序（pigeonhole sort）— ；需要 額外空間
- 基數排序（radix sort）— ；需要 額外空間
- 侏儒排序（gnome sort）—
- 圖書館排序（library sort）— 期望时间； 最坏时间；需要 額外空間
- 塊排序（英语：Block sort）（block sort）—

不穩定的排序

- 選擇排序（selection sort）—
- 希爾排序（shell sort）— 如果使用最佳的現在版本

- 克洛弗排序（Clover sort）— 期望时间， 最坏情况
- 梳排序—
- 堆排序（heap sort）—
- 平滑排序（英语：Smoothsort）（smooth sort）—
- 快速排序（quick sort）— 期望時間， 最壞情況；對於大的、亂數串列一般相信是最快的已知排序
- 內省排序（introsort）—
- 耐心排序（patience sort）— 最坏情況時間，需要額外的 空間，也需要找到最長的遞增子序列（longest increasing subsequence）

不實用的排序

- Bogo排序—