**常用排序算法和查找算法的时间复杂度和空间复杂度**

置顶 2018年04月19日 08:14:55 [chenkaibsw](https://me.csdn.net/chenkaibsw) 阅读数：3321

常用的排序算法的时间复杂度和空间复杂度

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 排序法 | 最差时间分析 | 平均时间复杂度 | 稳定度 | 空间复杂度 |
| 冒泡排序 | O(n2) | O(n2) | 稳定 | O(1) |
| 插入排序 | O(n2) | O(n2) | 稳定 | O(1) |
| 选择排序 | O(n2) | O(n2) | 稳定 | O(1) |
| 二叉树排序 | O(n2) | O(n\*log2n) | 不一顶 | O(n) |
| 快速排序 | O(n2) | O(n\*log2n) | 不稳定 | O(log2n)~O(n) |
| 堆排序 | O(n\*log2n) | O(n\*log2n) | 不稳定 | O(1) |
| 希尔排序 | O | O | 不稳定 | O(1) |

查找算法时间复杂度

|  |  |  |  |
| --- | --- | --- | --- |
| 查找 | 平均时间复杂度 | 查找条件 | 算法描述 |
| 顺序查找 | O(n) | 无序或有序队列 | 按顺序比较每个元素，直到找到关键字为止 |
| 二分查找（折半查找） | O(logn) | 有序数组 | 查找过程从数组的中间元素开始，如果中间元素正好是要查找的元素，则搜素过程结束；如果某一特定元素大于或者小于中间元素，则在数组大于或小于中间元素的那一半中查找，而且跟开始一样从中间元素开始比较。　如果在某一步骤数组为空，则代表找不到。 |
| 二叉排序树查找 | O(logn) | 二叉排序树 | 在二叉查找树b中查找x的过程为：  1. 若b是空树，则搜索失败  2. 若x等于b的根节点的数据域之值，则查找成功；  3. 若x小于b的根节点的数据域之值，则搜索左子树  4. 查找右子树。 |
| 哈希表法（散列表） | O(1) | 先创建哈希表（散列表） | 根据键值方式(Key value)进行查找，通过散列函数，定位数据元素。 |
| 分块查找 | O(logn) | 无序或有序队列 | 将n个数据元素"按块有序"划分为m块（m ≤ n）。  每一块中的结点不必有序，但块与块之间必须"按块有序"；即第1块中任一元素的关键字都必须小于第2块中任一元素的关键字；而第2块中任一元素又都必须小于第3块中的任一元素，……。然后使用二分查找及顺序查找。 |

按数量级递增排列，常见的时间复杂度有：

常数阶O(1),对数阶O( ),线性阶O(n),

线性对数阶O(nlog2n),平方阶O(n^2)，立方阶O(n^3),...，

k次方阶O(n^k),指数阶O(2^n)。

随着问题规模n的不断增大，上述时间复杂度不断增大，算法的执行效率越低。

如当一个算法的空间复杂度为一个常量，即不随被处理数据量n的大小而改变时，可表示为O(1)；当一个算法的空间复杂度与以2为底的n的对数成正比时，可表示为O(log2n)；当一个算法的空间复杂度与n成线性比例关系时，可表示为O(n).若形参为数组，则只需要为它分配一个存储由实参传送来的一个地址指针的空间，即一个机器字长空间；若形参为引用方式，则也只需要为其分配存储一个地址的空间，用它来存储对应实参变量的地址，以便由系统自动引用实参变量。

对于一个算法，其 [1] 时间复杂度和空间复杂度往往是相互影响的。当追求一个较好的时间复杂度时，可能会使空间复杂度的性能变差，即可能导致占用较多的存储空间；反之，当追求一个较好的空间复杂度时，可能会使时间复杂度的性能变差，即可能导致占用较长的运行时间。另外，算法的所有性能之间都存在着或多或少的相互影响。因此，当设计一个算法(特别是大型算法)时，要综合考虑算法的各项性能，算法的使用频率，算法处理的数据量的大小，算法描述语言的特性，算法运行的机器系统环境等各方面因素，才能够设计出比较好的算法。算法的时间复杂度和空间复杂度合称为算法的复杂度。