七大排序算法性能的分析

原创　2016年06月17日 16:05:26　标签：排序算法 / 性能 / 算法　2789

这里我来集中分析一下七大排序算法的性能问题。如果不当之处，敬请指正。

冒泡排序（Bubble）

排序算法	平均情况下	最好情况	最坏情况	稳定性	空间复杂度
冒泡	O（n^2）	O（n）	O（n^2）	稳定	1

冒泡排序算法里面含有2层循环，显然最坏时间复杂度为O（n^2），这里可能很多人不懂平均情况下的复杂度是什么意思，下面引用一段话：

> - 而平均运行时间也就是从概率的角度看，这个数字在每一个位置的可能性是相同的，所以平均的查找时间为n/2次后发现这个目标元素。平均情况更能反映大多数情况下算法的表现。平均情况分析就是对所有输入尺寸为n的输入，让算法运转一遍，然后取它们的平均值。当然，实际中不可能将所有可能的输入都运行一遍，因此平均情况通常指的是一种数学期望值，而计算数学期望值则需要对输入的分布情况进行假设。
> - 平均运行时间是所有情况中最有意义的，因为它是期望的运行时间。也就是说，我们运行一段程序代码时，是希望看到平均运行时间的。可现实中，平均运行时间很难通过分析得到，一般都是通过运行一定数量的实验数据后估算出来的。

那么为什么冒泡排序最好情况下的复杂度为O（n）呢？其实用我们最初的冒泡排序代码，其最好情况下复杂度依然为O（n^2）。如果冒泡排序算法在原来的基础上改进如下：

```
1   public static void b_getsort(int[] a)
2   {
3       int len = a.length;
4       int i,j;
5       boolean flag = true;
6       for(i=0;i<len-1&&flag;i++)
7       {
8           flag = false;
9           for(j = 0;j<len-1-i;j++)
10          {
11              if (a[j]>a[j+1])
12              {
13                  int temp = a[j];
14                  a[j] = a[j+1];
15                  a[j+1] = a[j];
16                  flag = true;
17              }
18          }
19      }
20  }
```

设置了一个flag变量用来标示我们操作的那部分数据是否发生了数据交换。
比如说，在我们第一次“冒”的时候，均未发生过交换（即原数组本身是有序的），那么flag就是为false，这也就意味着我们根本不必再进行第二次、第三次的“冒泡”。其实这也就是最好的情况，即只进行了一次对数组所有元素的访问。

简单选择排序（Select Sort）

排序算法	平均情况下	最好情况	最坏情况	稳定性	空间复杂度
简单选择	O（n^2）	O（n^2）	O（n^2）	不稳定	1

简单选择排序也是2层循环，因而也是O（n^2）。

插入排序（Insert Sort）

排序算法	平均情况下	最好情况	最坏情况	稳定性	空间复杂度
插入排序	O（n^2）	O（n）	O（n^2）	稳定	1

插入排序跟冒泡排序的性能是一样的。

希尔排序（Shell Sort）

排序算法	平均情况下	最好情况	最坏情况	稳定性	空间复杂度
希尔排序	O（$nlogn$）	依赖步长	依赖步长	稳定	1

写到这儿，我感觉有必要普及一下几种常用的算法复杂度。
常见的算法时间复杂度由小到大依次为：**O(1) < O($logn$) < O(n) < O($nlogn$) < O(n^2) < O(n^3) < ... < O(2^n) < O(n!)**

> 在希尔排序中，增量的选取十分关键。“可究竟选取什么样的增量才是最好，目前还是一个数学难题，迄今为止还没有人找到一种最好的增量序列。不过大量的研究表明，当增量序列为 $dlta[k] = 2^{t-k+1} - 1, 0 \le k \le t \le \lfloor log_2(n+1) \rfloor$“时，可以获得不错的效率，其时间复杂度为O（$n^{3/2}$），要好于直接排序的O（$n^2$）。需要注意的是，增量序列的最后的一个增量值必须等于1才行。另外由于记录是跳跃式的移动，希尔排序并不是一种稳定的排序算法。不管怎么说，希尔排序算法的发明，使得我们终于突破了慢速排序的时代（超越了时间复杂度为O（n^2））”——>引自《大话数据结构》。

不过至于平均情况复杂度为什么会为O（$nlogn$），我也没搞清楚。反正它的复杂度位于O(n)和O(n^2)之间。

快速排序

排序算法	平均情况下	最好情况	最坏情况	稳定性	空间复杂度
快速排序	O（$nlogn$）	O（$nlogn$）	O（n^2）	不稳定	O（$logn$）

> 快速排序的平均时间为$T_{avg}(n) = knln(n)$,其中n为待排序记录中记录的个数，k为某个常数，经验证明，在所有同数量级的此类（先进的）排序方法中，快速排序的常数因子k最小。因此，就平均时间而言，快速排序是目前被认为是最好的一种内部排序方法。——>《数据结构（C语言版）》，严蔚敏 吴伟民著

由于快速排序的算法复杂度分析涉及太多数学知识，这里不做过多分析。不过我们应该注意快排的空间复杂度以及稳定性。

归并排序

排序算法	平均情况下	最好情况	最坏情况	稳定性	空间复杂度
归并排序	O（$nlogn$）	O（$nlogn$）	O（$nlogn$）	稳定	O（n）

从上表可以看出，三种情况下归并排序的复杂度都为O（$nlogn$），空间复杂度为O（n）。

堆排序

排序算法	平均情况下	最好情况	最坏情况	稳定性	空间复杂度
堆排序	O（$nlogn$）	O（$nlogn$）	O（$nlogn$）	稳定	1

这里我们可以拿堆排序和归并排序比较一下，二者时间复杂度相同，但是稳定性和空间复杂度方面有差别。

最后，我来综合说一下几个面试问的比较多的问题。

- 算法的稳定性？

算法的稳定性是指在序列中关键字相同的元素，经过某种排序算法之后，这些元素之间的顺序保持不变。如果发生改变，那么就称该算法是不稳定的。

- 算法的不稳定和稳定会分别导致什么？

①实际排序中，我们操作的可能不是整数，可能是一些很大的对象，而交换元素 产生的开销对程序性能的影响便会很大
②基数排序（将操作元素划分为多个关键字进行排序，比如很多学生的数据，首先按照班级排序，然后班级相同的再按照年龄排序，如此等等）在不稳定排序算法中无法完成。

最后来个总结图（来自倪升武的博客）

常见的排序算法比较表

排序	平均情况	最好情况	最坏情况	稳定与否	空间复杂度
冒泡排序	O(N^2)	O(N)	O(N^2)	稳定	1
选择排序	O(N^2)	O(N^2)	O(N^2)	不稳定	1
插入排序	O(N^2)	O(N)	O(N^2)	稳定	1
希尔排序	O(NlogN)	（依赖于增量序列）		不稳定	1
快速排序	O(NlogN)	O(NlogN)	O(N^2)	不稳定	O(logN)
归并排序	O(NlogN)	O(NlogN)	O(NlogN)	稳定	O(N)
二叉树排序	O(NlogN)	O(NlogN)	O(N^2)	稳定	O(N)
堆排序	O(NlogN)	O(NlogN)	O(NlogN)	不稳定	1
拓扑排序	O(N+E)	—	—	—	O(N)