数据结构中内部排序算法的分析

吴红芝，郭麦成，吴　浩

(长江大学计算机科学学院研究生院，湖北 荆州 434023)

摘　要：排序是计算机程序设计中经常遇到的一个重要内容，它的功能是将一个数据集合，按关键字重新排列成一个有序的序列。然而，由于排序算法程序须考虑设计路线、时间复杂度及稳定性等因素，初学者在理解上存在较大的困难，文章针对这些具体问题，提供一些行之有效的解决方法。

关键词：排序；算法；稳定性；时间复杂度

Analysis of Internal Sorting Algorithm in Data Structure

WU Hong-zhi, GUO Mai-cheng, WU Hao

(*Graduate School, College of Computer Science, Yangtze University, Jingzhou, Hubei 434023, China*)

Abstract: Sorting is an important matter often encountered in computer programming, and its function is to rearrange a set of data items into an ordered sequence by keyword. However, because of the factors which must be considered in sorting program, such as design route, time complexity and stability, there is a big problem in understanding for beginners. For these specific problems, some effective solutions are provided.

Key words: sorting; algorithm; stability; time complexity

0 引言

排序依据所涉及的存储器不同，分为两大类：一类是内部排序，指的是对存放在计算机存储器中的待排序记录进行的排序过程；另一类则是外部排序，指的是待排序记录数量很大，以至内存不能容纳全部记录，在排序过程中尚需对外存进行访问的排序过程。本文将主要讨论内部排序中的直接插入排序、简单选择排序两种排序算法，并简要地分析快速排序的稳定性。

与其他算法相比，排序也是数据结构中的一种算法，只不过它所实现的功能比较具体，即实现无序系列的排序。同样，它也涉及到一种算法的设计思想、参数的引用及时间复杂度等相关因素；但与其他算法有所不同，排序算法涉及到了稳定性与不稳定性的概念——关键字相同，排序后它们的顺序不变的排序方法称为稳定排序，顺序有可能发生改变的排序方法称为不稳定排序。例如，1，2^1，3，2^2，5，序列中有两个2，标记为2^1、2^2，用一种排序方法进行排序，排序后，若2^1始终在2^2的前面，则称稳定的排序，若2^1有可能在2^2的后面，则称不稳定的排序。

排序方法大致可分为插入排序、交换排序、选择排序、归并排序和计数排序等五类。它们的实现过程涉及到比较和移位两种操作，即比较关键字的大小，将记录从一个位置移动到另一个位置，最后得到一个有序序列。学习排序的过程，其关键在于理解排序的思想，分析各种排序的特点，利用流程来实现排序过程。

1 几种排序算法的实现过程

内部排序按其算法设计思想的不同，可分为直接插入排序、折半插入排序、希尔排序、冒泡排序、快速排序、简单选择排序、堆排序、归并排序和基数排序等。其中的插入排序有直接插入排序、折半插入排序和希尔排序三种；交换排序有冒泡排序和快速排序；选择排序有简单选择排序和堆排序。

从排序的稳定性来看，稳定的排序有：直接插入排序、折半插入排序、冒泡排序和归并排序，不稳定的排序有：希尔排序、快速排序、简单选择排序和堆排序；从排序算法的时间复杂度来看，直接插入排序、折半插入排序、冒泡排序、简单选择排序的时间复杂度为$o(n^2)$，希尔排序的时间复杂度为$o(n^{3/2})$，快速排序、堆排序和归并排序的时间复杂度为$o(n\log_2 n)$。

本文将从排序算法的设计思想、程序设计、稳定性和时间复杂度等五个方面具体分析，直接插入排序、冒泡排序、快速排序、简单选择排序和堆排序。此外，在排序程序的设计中，将引用一个数组a[n]来保存排序序列，排序序列中的所有元素均为整型元素，且均把序列排成升序。

1.1 直接插入排序

1.1.1 程序设计思想

直接插入排序把序列分成已排序列和未排序列两个部分。开始时，第一个元素为已排序列，后面的所有元素均为未排序列；在排序的过程中，把前面的序列看成一个已排好的序

参考文献：

[1] 贾素玲，王强.XML技术应用[M].清华大学出版社，2007.

[2] 王伟达，芦东昕，孟照星.关系数据库与XML的双向数据传输的机制与实现[J].计算机应用研究，2005.22(2):164~166

[3] 精确计算代码执行时间[EB/OL].http://www.cnblogs.com/aiyagaze/archive/2006/09/23/512507.2006.

[4] 丁跃潮，张涛.XML实用教程[M].北京大学出版社，2006.

列，把后面的元素逐个插入到前面已排好的序列中。例如序列：25，17，8，49，23，15，20。

一次排序：25是一个已排好的序列，17，8，49，23，15，20是未排序列，选择未排序列中第一个数17插入到前面的已排序列中，形成序列：17，25，8，49，23，15，20。此时17，25是已排序列，8，49，23，15，20是未排序列。

二次排序：选择未排序列中第一个数8插入到前面的已排序列，形成已排序列8，17，25，未排序列49，23，15，20，此时整个序列为8，17，25，49，23，15，20。

依次类推，直至整个序列有序。

1.1.2 程序设计

```
InsertSort(int a[n]) //a[i-1]是已排序列的最后一个元素，a[i]是未。
{ int i, j, b ; //排序列的第一个元素，实质是把a[i]插入到
  for(i=1; j<n; i++) //前i-1个元素中
    if(a[i]<a[i-1]) //是否需要插入
    { b=a[i];
      a[i]=a[i-1]; // a[i-1]是最大元素，放到最后，即第i个位置
      for( j=i-2; j>=0;j-- ) //把a[i]往前插
      { if(a[i]<b) break; //若发现已排序列中有一个元素比b小。
        a[j+1]=a[j]; //则找到插入位置，for终止，元素后移。
      }
      a[j+1]=b; //插入元素
    }
}
```

1.1.3 稳定性分析

由于此算法在处理过程中，只是把后序元素往前插，在比较的过程中，若出现两数相同的情况，也不会改变两数的相对位置，固此算法是稳定的。

1.1.4 时间复杂度

根据算法的程序设计可以看出，程序利用了两次for循环，两次循环的最大次数都可能会达到n次，固时间复杂度为$o(n^2)$。

1.2 简单选择排序

1.2.1 程序设计思想

简单选择排序均在未排序列中选择一个最小的元素（开始时整个序列无序，简单选择排序是从最小的元素选到最大的元素，有序序列在前，无序序列在后），插入到该元素所在的位置，直至所有的元素均选择完毕，此时序列已有序。例如序列：25，17，8，49，23，15，20。

首先，序列25，17，8，49，23，15，20是一个无序序列，25与17比较，17小，25与17交换位置，序列变为17，25，8，49，23，15，20；17与8比较，8小，8与17交换位置，序列变为8，25，17，49，23，15，20；用8和后面的元素49，23，15，20依次比较，8始终最小，不需交换位置，固8为最小元素。此时，8为有序序列，25，17，49，23，15，20为无序序列。然后，用同样的方法在无序序列25，17，49，23，15，20中选择一个最小的元素，序列变为8，15，25，49，23，17，20，有序列为8，15，无序序列25，49，23，17，20需再次进行选择，直至整个序列有序（黑色标记的元素为每一次选择的最小元素）。

1.2.2 程序设计

```
SelectSort (into a[n])
{ int i, j, b;
  for (i=1; i<n; i++)  //遍历选择次数，需n-1次
    for(j=i; j<n; j++)  //此for循环实现选择的过程
      if(a[j]<a[j-1])
      { b=a[i-1];
        a[i-1]=a[j]; //实现元素交换
        a[j]=b;
      }
}
```

1.2.3 稳定性分析

简单选择排序属于不稳定的排序，因为在程序的运行过程中，相同元素的相对位置有可能发生改变，例如序列：13^1，49，13^2，10，28。

一次选择，序列变为：13^1，49，13^2，10，28。

二次选择，序列变为：13^1，49，13^2，10，28。

三次选择，序列变为：10，49，13^2，13^1，28。

三次选择以后，13^1和13^2的相对位置发生了改变，故不再稳定。

1.2.4 时间复杂度

从程序的设计过程来看，程序中有两个for循环，且最大参数均为n，固时间复杂度为$o(n^2)$。

1.3 快速排序的稳定性分析

快速排序属于不稳定排序算法，因为在排序的过程中，相同元素的相对位置可能会发生改变。例如，在一个序列中，若有两个相同的元素（均为49^1和49^2），49^1为第一个元素，49^2靠后，且序列中比49小的元素较比49大的元素多，则两个元素的位置一定会发生改变，根据快速排序的思想（49^1把序列分成两个部分，比49^1小的所有元素在前，比49^1大的所有元素在后，49^1在中间），49^2必定在小元素序列中，49^1和49^2的相对位置发生了改变，固不再稳定。

例如序列49^1、58、12、49^2、23、72、40、41、83、20中有两个49，比它大的元素有3个，比它小的元素有5个，一次快速排序以后49^1之前有5个元素，而49^2排在第四位，且在比较的过程中其位置不会发生改变，49^2排在了49^1的前面，相对位置发生了改变。

2 结束语

内部排序是数据结构中很重要的一个章节，也是学习过程中的一个难点，但只要理清算法的思想，合理的设计程序的流程及变量，问题定会迎刃而解。鉴于文章篇幅及个人的能力有限，本文仅仅分析了直接插入排序、简单选择排序两种比较简单且常用的算法，并对快速排序的稳定性进行了分析，其他算法，读者可以借鉴本文的思路进行研究。

参考文献：

[1] Clifford A Shaffer，张铭，刘晓丹译.数据结构与算法分析(Java版)[J].电子工业出版社，2001.

[2] Sartaj Sahni.数据结构算法与应用—C++语言描述(英文版)[M].机械工业出版社，1999.

[3] William ford，William Topp.数据结构C++语言描述(英文版)[M].清华大学出版社，1997.

[4] 严蔚敏，吴伟民.数据结构(C语言版)[M].清华大学出版社，1997.

[5] 黄宇潜，滕少华.数据结构教程[M].华中科技大学出版社，1996.