第7章 排 序

在这一章，我们讨论元素数组的排序问题。为简单起见，假设例子中的数组只包含整数，当然我们的程序也允许更一般的对象。对于本章的大部分内容，我们还假设整个排序工作能够在主存中完成，因此，元素的个数相对来说比较小(小于几百万)。当然，不能在主存中完成而必须在磁盘或磁带上完成的排序也相当重要。这种类型的排序叫做外部排序(external sorting)，将在本章末尾进行讨论。

我们对内部排序的考查将指出：

- 存在几种容易的算法以 $O(N^2)$完成排序，如插入排序。
- 有一种算法叫做希尔排序(Sellsort)，它编程非常简单，以 $o(N^2)$运行，并在实践中很有效。
- 存在一些稍微复杂的 $O(N \log N)$的排序算法。
- 任何通用的排序算法均需要 $\Omega(N \log N)$次比较。

本章的其余部分将描述和分析各种排序算法。这些算法包含一些有趣的和重要的代码优化和算法设计思想。排序也是使得分析能够得以精确地进行的范例。预先说明，在适当的时机，我们将尽可能多地做一些分析。

7.1 预备知识

我们描述的算法都将是可以互换的。每个算法都将接收包含一些元素的数组；假设所有的数组位置都包含要被排序的数据。我们还假设 N 是传递到排序例程的元素的个数。

正如1.4节所描述的，被排序的对象属于Comparable类型。因此我们使用CompareTo方法对输入数据施加相容的排序。除(引用)赋值运算外，这是仅有的允许对输入数据进行的操作。在这些条件下的排序叫做**基于比较的排序**(comparison-based sorting)。在默认的排序没有或不可接受的情况下，我们很容易用Comparator来重写排序算法。

7.2 插入排序

7.2.1 算法

最简单的排序算法之一是**插入排序**(insertion sort)。插入排序由 $N-1$ 趟排序组成。对于 $p=1$到 $N-1$ 趟，插入排序保证从位置0到位置 p 上的元素为已排序状态。插入排序利用了这样的事实：已知位置0到位置 $p-1$ 上的元素已经处于排过序的状态。图7-1显示一个数组样例在每一趟插入排序后的情况。

原始数组	34	8	64	51	32	21	移动的位置
$p=1$ 趟之后	8	34	64	51	32	21	1
$p=2$ 趟之后	8	34	64	51	32	21	0
$p=3$ 趟之后	8	34	51	64	32	21	1
$p=4$ 趟之后	8	32	34	51	64	21	3
$p=5$ 趟之后	8	21	32	34	51	64	4

图7-1 每趟后的插入排序