第1章 引 论

在这一章，我们阐述本书的目的和目标并简要复习离散数学以及程序设计的一些概念。我们将要

- 看到程序对于合理的大量输入的运行性能与其在适量输入下运行性能的同等重要性。
- 概括为本书其余部分所需要的基本的数学基础。
- 简要复习递归。
- 概括用于本书的 Java 语言的某些重要特点。

1.1 本书讨论的内容

设有一组 N 个数而要确定其中第 k 个最大者。我们称之为**选择问题**(selection problem)。大多数学习过一两门程序设计课程的学生写一个解决这种问题的程序不会有什么困难。“明显的”解决方法是相当多的。

该问题的一种解法就是将这 N 个数读进一个数组中，再通过某种简单的算法，比如冒泡排序法，以递减顺序将数组排序，然后返回位置 k 上的元素。

稍微好一点的算法可以先把前 k 个元素读入数组并(以递减的顺序)对其排序。接着，将剩下的元素再逐个读入。当新元素被读到时，如果它小于数组中的第 k 个元素则忽略之，否则就将其放到数组中正确的位置上，同时将数组中的一个元素挤出数组。当算法终止时，位于第 k 个位置上的元素作为答案返回。

这两种算法编码都很简单，建议读者试一试。此时我们自然要问：哪个算法更好？哪个算法更重要？还是两个算法都足够好？使用一千万个元素的随机文件和 $k = 5\,000\,000$ 进行模拟将发现，两个算法在合理的时间量内均不能结束；每种算法都需要计算机处理若干天才能算完(虽然最后还是给出了正确的答案)。在第 7 章将讨论另一种算法，该算法将在一秒钟左右给出问题的解。因此，虽然我们提出的两个算法都能算出结果，但是它们不能被认为是好的算法，因为对于第三种算法能够在合理的时间内处理的输入数据量而言，这两种算法是完全不切实际的。

第二个问题是解决一个流行的字谜。输入是由一些字母构成的一个二维数组以及一组单词组成。目标是要找出字谜中的单词，这些单词可能是水平、垂直或沿对角线上任何方向放置的。作为例子，图 1-1 所示的字谜由单词 this、two、fat 和 that 组成。单词 this 从第一行第一列的位置即(1,1)处开始并延伸至(1,4)；单词 two 从(1,1)到(3,1)；fat 从(4,1)到(2,3)；而 that 则从(4,4)到(1,1)。

	1	2	3	4
1	t	h	i	s
2	w	a	t	s
3	o	a	h	g
4	f	g	d	t

图 1-1 字谜示例

现在至少也有两种直观的算法来求解这个问题。对单词表中的每个单词，我们检查每一个有序三元组(行、列、方向)验证是否有单词存在。这需要大量嵌套的 for 循环，但它基本上是直观的算法。

也可以这样，对于每一个尚未越出谜板边缘的有序四元组(行、列、方向、字符数)我们可以测试是否所指的单词在单词表中。这也导致使用大量嵌套的 for 循环。如果在任意单词中的最