高等学校21世纪计算机教材

算法设计与分析

梁田贵　张　鹏　编著

冶金工业出版社

高等学校21世纪计算机教材

算法设计与分析

梁田贵　张　鹏　编著

北　京

冶 [redacted] 社

2004

内 容 简 介

算法研究是计算机科学的核心课题之一，其研究的目的在于设计出运算效率更高、占用空间更小的解决计算机问题的方法。算法设计与分析也是计算机相关专业的核心课程之一，它是程序设计语言以及离散数学课程的后续课程。

目前对于计算机算法介绍的教材通常有两种：一种着重介绍的是数据结构本身的实现，通常称作数据结构与算法；而另一种着重介绍的是算法设计的原理，通常称作算法设计与分析，两者的差异仅仅在于着眼点的不同而已。本书属于后者，是关于算法设计技术与算法分析技术的介绍，涉及到的算法主要是日常生活以及程序设计中常见的一些问题，这对常见问题的解决以及软件开发过程有实用的参考价值。主要内容包括：算法概述、算法设计基础、算法分析基础、排序算法、搜索算法、类搜索算法与字符串匹配算法、图与树相关算法、几何问题算法、数值算法、组合问题算法、加密算法与安全机制以及算法复杂性理论简介。

本书内容丰富、结构合理、语言通俗易懂，不仅可作为高等学校计算机专业教材，也可作为广大工程技术人员与自学读者的学习参考书。

图书在版编目（CIP）数据

算法设计与分析 / 梁田贵等编著. —北京：冶金工业出版社，2004.9
ISBN 7-5024-3614-6

I. 算… II. 梁… III. ①电子计算机—算法设计 ②电子计算机—算法分析 IV. TP301.6

中国版本图书馆 CIP 数据核字（2004）第 086876 号

出版人 曹胜利（北京沙滩嵩祝院北巷 39 号，邮编 100009）
责任编辑 程志宏
湛江蓝星南华印务公司印刷；冶金工业出版社发行；各地新华书店经销
2004 年 9 月第 1 版，2004 年 9 月第 1 次印刷
787mm × 1092mm 1/16； 17.25 印张； 397 千字； 268 页； 1-3500 册
28.00 元
冶金工业出版社发行部 **电话：**（010）64044283 **传真：**（010）64027893
冶金书店 **地址：北京东四西大街 46 号（100711）** **电话：**（010）65289081
（本社图书如有印装质量问题，本社发行部负责退换）

前　言

一、关于本书

历史上，许多伟大的计算机科学家设计了许多很好的算法，而更多的科学家们正在研究传统问题与不断涌现的新问题的解决办法。纵观全书，这里并没有创造出任何新的算法，因为这不是写作目的，作者只是希望把传统的算法设计与分析中最基础、最重要的内容用一种更清晰的语言、更直观的形式展现给读者。只有全面系统地了解与学习算法设计这门学科的基础知识，掌握算法设计的思想，才有可能创造一个新的算法。

二、本书结构

本书总体可以分为两大部分：第一部分为算法设计与分析的理论基础，包括第 1 章、第 2 章、第 3 章以及第 12 章；第二部分为算法的实际应用，包括第 4 章 ~ 第 11 章。

本书具体结构安排如下：

第 1 章：算法概述。主要介绍了算法的基础知识和常见问题的类型以及解决问题的一般步骤。

第 2 章：算法设计基础。主要介绍了算法设计常用到的数据结构以及常用的算法设计方法。其中常用数据结构包括数组与链表、栈与队列、图与树、集合与字典；而常用的算法设计方法有分治法、贪婪法、动态规划法、回溯法以及分支限定法。

第 3 章：算法分析基础。主要介绍了算法分析的基本框架、时间复杂度渐进分析的数学基础、算法分析举例以及递归算法分析再讨论。

第 4 章：排序算法。主要介绍了排序相关的概念以及常用的排序算法。排序算法按照是否需要使用外部存储器可以分为内排序与外排序，而内排序算法按照基本思想的不同可分为交换排序、插入排序、选择排序、堆与堆排序、归并排序和统计排序等。本章对每一类排序算法的思想、实现以及分析都进行了详细介绍。

第 5 章：搜索算法。主要介绍了搜索相关的概念、静态搜索表的算法、二叉搜索树搜索、AVL 树、2-3 树、最优二叉搜索树、索引结构以及散列方法。

第 6 章：类搜索算法与字符串匹配算法。主要介绍了搜索问题扩展、搜索与排序以及字符串匹配算法。

第 7 章：图与树相关算法。主要介绍了二叉树的遍历、二叉树的计数、图的遍历、图的路径与带权路径、两点之间的最短路径、任意点之间的最短路径、最小生成树、最大流量问题、最小费用最大流量问题、霍夫曼树以及图的应用举例。

第 8 章：几何问题算法。主要介绍了几何形体在计算机中的表示、初等几何问题算法、最近邻点问题算法以及凸包问题算法。

第 9 章：数值算法。主要介绍了杨辉三角、多项式求值、大整数乘法、线性方程组与高斯消元法以及矩阵基本运算。

第 10 章：组合问题算法。主要介绍了排列问题、幂集问题、背包问题以及旅行家问

题。

第 11 章：加密算法与安全机制。主要介绍了加密算法和安全机制的相关知识。加密算法主要包括 DES 算法、RSA 算法以及 MD5 算法，同时还介绍了三个算法在鉴别协议、消息完整性协议以及公开密钥分发协议中的应用。

第 12 章：算法复杂性理论简介。主要介绍了算法问题的相关知识和图灵机的基础知识。

附录：算法伪代码索引。

三、本书特点

本书在内容丰富，除了传统的排序、搜索、字符串匹配和图论问题以外，还介绍了数值计算、几何数学和组合数学的若干问题。在算法的介绍过程中，本书的着眼点在于算法设计与分析的介绍而忽略算法对应的数据结构本身，所以对于算法的实现一般采用的数据结构是数组。同时，本书还十分重视算法效率的分析，除了给出每个算法的分析结果以外，还重点讲述了算法分析的基本方法与框架。

四、本书适用对象

本书不仅可作为高等学校计算机专业教材，也可作为广大工程技术人员与自学读者的学习参考书。

由于编者水平有限，时间仓促，书中难免有疏漏不妥之处，敬请读者批评指正。

虽然经过严格的审核、精细的编辑，本书在质量上有了一定的保障，但我们的目标是力求尽善尽美，欢迎广大读者和专家对我们的工作提出宝贵建议，联系方法如下：

电子邮件：service@cnbook.net

网址：www.cnbook.net

此外，**本书所附送的电子教案也可从该网站免费下载**，该网站还有一些其他相关书籍的介绍，可以方便读者选购参考。

编　者

2004 年 7 月

目　录

第 1 章　算法概述

本章首先介绍算法的一些基本概念，包括算法的定义以及算法必须满足的五个基本要求。然后通过分析求解最大公约数这一经典例子，详细论述算法在实际问题中的应用。接着列举并简要介绍在日常生活以及计算机程序设计中经常遇到的问题的类型、特点以及解决这些问题的一般方法。

1.1　算法简介

1.1.1　算法的定义

什么是算法呢？这个概念没有一个统一的定义。一般而言，对于计算机算法的概念是这样描述的：算法是在有限步骤内求解某一问题所使用的一组定义明确的规则。通俗一点来说，就是计算机解题的过程。在这个过程中，无论是形成解题思路还是编写程序，都是在实施某种算法。前者是“推理”实现的算法，后者是“操作”实现的算法。

一个算法应该具有以下五个重要的特征：

（1）有穷性：一个算法必须保证执行有限步之后结束，不能终止的过程不属于算法的范畴。

（2）确切性：算法的每一个步骤必须具有确切的定义，即每一步要执行的动作是确切的，是无二义性的。

（3）可行性：一个算法是可行的，就是算法中描述的操作都是可以通过已经实现的基本运算执行有限次来实现。并且，在任何条件下，算法只有惟一的一条执行路径，即对于相同的输入只能得出相同的输出。

（4）输入：一个算法必须具有 0 个或多个输入，以刻画运算对象的初始情况。0 个输入适用于算法本身已确定了初始条件的情况。

（5）输出：一个算法必须具有一个或多个输出，以反映对输入数据加工后的结果。没有输出的算法是毫无意义的。

算法的概念如图 1-1 所示。

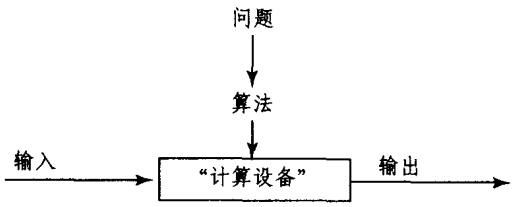

图 1-1

小知识：Algorithm 一词的由来。Algorithm（算法）这一英语单词的本身就十分有趣。Algorithm（算法）一词与 Logarithm（对数）非常相似，只是头四个字母的次序不同。直至 1957 年之前，算法一词在 Webster's New World Dictionary（《韦氏新世界词典》）中还未出现，只能找到带有它的古代涵义的比较古老形式的“Algorism”（算术）。“算术”指的是

用阿拉伯数字进行算术运算的过程。因为在中世纪时，珠算家用算盘进行计算，而算术家用算术进行计算。中世纪之后，对这个词的起源就拿不准了，早期的语言学家推断它是把 algiros（费力的）+arithmos（数字）组合起来派生而成的，但另外一些人则不同意这种说法，认为这个词是从“喀斯迪尔国王 Algor”派生而来的。

最后，数学史学家发现了 algorism（算术）一词的真实起源：它来源于著名的 Persian Textbook（《波斯教科书》）的作者的名字 Abu Ja'far Mohammed ibn Musa al-Khowarizm（约公元前 825 年）——从字面上看，这个名字的意思是“Ja'far 的父亲，Mohammed 和 Musa 的儿子，Khowarizm 的本地人”。Khowarizm 是前苏联 ХИВА（基发）的小城镇。Al-Khowarizm 写了著名的书 Kitab al jabr w'al-muqabala（《复原和化简的规则》）；另一个词“algebra”（代数），是从他的书的标题引出来的，尽管这本书实际上根本不是讲代数的。

逐渐地“algorism”的形式和意义就变得面目全非了。牛津英语字典上提到，这个词是同 arithmetic（算术）相混淆而形成的错拼词，由 algorism 又变成 algorithm。一本早期的德文数学词典 Vollstandiges Mathematisches Lexicon（《数学大全辞典》），给出了 Algorithmus（算法）一词的定义：“在这个名称之下，组合了四种类型的算术计算的概念，即加法、乘法、减法、除法”。拉丁短语 algorithmus infinitesimalis（无限小方法），在当时就用来表示 Leibnitz（莱布尼兹）所发明的以无限小量进行计算的微积分方法。

在 1950 年左右，algorithm 一词经常同欧几里德算法（Euclid's algorithm）联系在一起。欧几里德算法就是在欧几里德的《几何原本》（Euclid's Elements，第 VII 卷，命题 i 和 ii）中所阐述的求两个数的最大公约数的过程（即辗转相除法）。

如图 1-2 所示为 Abu Ja'far Muhammad ibn Musa Al-Khwarizmi（也及 al-Khowarizm）的头像，他出生于约公元 780 年的 Baghdad（现在伊拉克首都巴格达），逝世于约公元 850 年。

图 1-2

1.1.2 算法的重要性

算法的学习是很重要的，可以说算法是计算的核心与灵魂。如果仅仅学习了某一门编程语言的语法、数据结构，那么只是了解了那门语言本身而已，无法把该门编程语言运用到实际问题的解决中。算法正是把这些语法、数据结构运用到问题解决的过程。

算法大师 Donald Knuth 曾经说过：一个人只有当他要把他学的知识教给别人的时候，他才真正了解了这些知识，而设计算法正是把知识教给计算机的过程。因此算法的设计可以使得设计者真正了解这些知识。

1.1.3 一个简单例子的算法表示

说了那么多的概念，这里列举一个大家都很熟悉的例子来解释上面所说的算法概念，

也就是中学阶段关于两个数的最大公约数的求法（greatest common divisor），这个数学问题的表述如下：

对于两个非负整数 m 和 n（其中 m、n 不同时为零），如果整数 d 是同时整除 m 和 n 的最大的整数，那么称 d 是 m、n 的最大公约数，记为 d = gcd（m，n）。

一般求最大公约数的办法是首先分别找出 m、n 的素因子，再找出它们公共的素因子，则最大公约数是这些公共素因子的乘积。

例如：求 gcd（24，18）。

其中：

24 = 2 * 2 * 2 * 3

18 = 2 * 3 * 3

可以看出它们都有素因子 2 和 3，故 d = gcd（24，18）= 2 * 3 = 6。

对于一般情况，求 d = gcd（m，n）的过程用自然语言可以描述如下：

（1）找出 m 的素因子。

（2）找出 n 的素因子。

（3）找出 m，n 公共的素因子。

（4）计算所有公共素因子的乘积，结果即为 m，n 的最大公约数。

但是这样的过程能不能称之为算法呢？回顾算法的五大特征可以发现，上面的步骤没有满足确切性的要求，也就是说：计算机没有办法理解如何找出 m、n 的素因子，如何找出 m、n 公共的素因子。或者也可以这样理解，算法的每一个步骤都必须能够使用程序语言来实现。例如如果使用的是 C 语言，那么它本身仅仅提供了四则运算以及循环判断等逻辑控制的机制，而没有提供计算某个数的素因子的机制（也许有人说可以使用子函数，但是子函数的最终实现还是只能靠这些语言本身提供的元素完成），这一点是很重要的。

因此可以说，上面的表示还不能称为算法表示，至少在细化这些步骤之前不能。但并不是说中学阶段的老师所教的求解两个数最大公约数的方式是无法实现的，本章在后面会给出每一步具体做法的讨论。

下面再看看求最大公约数的 Euclid's algorithm，也叫辗转相除法，该算法是根据等式 1-1 而来的：

gcd（m，n）= gcd（n，m mod n）

其中 m mod n 表示 m 除以 n 的余数。

用这个算法重新计算刚才的 gcd（24，18）如下：

gcd（24，18）= gcd（18，6）= gcd（6，0）= 6

结果和上面算法的结果是一样的，至于等式 1-1 的证明，详细内容可参考参考文献【14】的相关内容。

上述的过程用自然语言可以表示如下：

（1）如果 n = 0，计算停止并返回 m，m 即为结果；否则，继续（2）。

（2）记 r 为 m 除以 n 的余数，即 r = m mod n。

（3）把 n 赋值给 m，把 r 赋值给 n，继续（1）。

上面的步骤用伪代码表示如下：

```
ALGORITHM Euclid(m, n)
```

```
//计算 gcd(m, n)
//输入：非负整数 m, n, 其中 m, n 不同时为零
//输出：m, n 的最大公约数
while n ≠ 0 do
    r   ← m mod n
    m   ← n
    n   ← r
return m
```

这里可以验证辗转相除法是否符合算法的五个要求，显然其中输入/输出以及确切性、可行性的要求都是满足的。由于过程使用了循环判断，因此重点检查过程是否满足有穷性。由于每一次循环都把 r 的值赋予 n，而 r 为两个数相除的余数，故每一次 r 总会减小，同时要求 r 是非负的，故 n 最终会达到零值而终止循环。也就是说辗转相除法满足了算法的所有约束，这一过程可以称作辗转相除算法。

通过对同一个问题两种不同计算过程的比较，可以更加深刻地认识到，什么才是一个算法，算法需要满足什么样的约束。中学阶段所学的求最大公约数的过程也是正确的，问题是在于这个过程没有能够清晰地表述成计算机所能执行的一系列指令集合，针对这一点，下面继续对这一过程进行细化。

1.1.4 求最大公约数算法的细化

1. 求整数的素因子的细化

关于求一个整数 x 的素因子，通常的做法是用 x 除以比 x 小的素数，如果能够整除，那么这个素数就是 x 的一个素因子。

用伪代码表示如下：

```
ALGORITHM PrimeFactor(x)
//计算正整数 x 的素因子
//输入：正整数 x
//输出：x 的素因子
p = 1
while p ≤ x and isPrime(p) do
    if x mod p = 0
        print(p)
```

其中 isPrime 函数的作用是判断 p 是否是素数，这里只是简单介绍一个称为 sieve of Eratosthenes 的求小于 x 的所有素数的算法，这个算法是公元前三世纪的希腊天文学家、数学家和地理学家埃拉托色尼提出的，主要思想是这样的：

首先在纸上列出所有小于 x 的正整数；然后把 1 划去，把 2 的倍数 4、6 等划去，把 3 的倍数 3、9 等划去……对于纸上剩下的数，继续进行这样的过程，最后纸上剩下的数就是所有小于 x 的素数。

例如，当 x=25 时，上述过程可以表示如下：

```
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
2 3   5   7   9    11    13    15    17    19    21    23    25
2 3   5   7        11    13          17    19          23    25
2 3   5   7        11    13          17    19          23
```

上述过程事实上没有必要循环 x 次才得到所有的素数。观察发现，当轮到把 p 的倍数划去时，因为在 p 的所有倍数中比 pp 小的整数如 2p、3p、…、(p-1)p 都已经在前面过程

中被划去了（如 3p 在 3 那一轮已经被划去），所以第一个需要考虑划去的数是应该是 p*p（显然 p*p 应该小于 x），因此算法只需要做到第$\lfloor\sqrt{n}\rfloor$轮即可以找到所有比 x 小的素数（$\lfloor\sqrt{n}\rfloor$表示不大于$\sqrt{n}$的最大整数）。

用伪代码表示如下：

```
ALGORITHM Sieve(x)
//利用筛选法求解小于任意正整数的素数
//输入：正整数 x
//输出：小于或等于 x 的素数
for p ← 2 to x do
    A[p] ← p        //构建整数序列
for p ← 2 to ⌊√n⌋ do
    if A[p] ≠ 0
        j ← p*p
        while j ≤ x do
            A[j] ← 0
            j ← j + p
i ← 0
for p ← 2 to x do //把所有的素数赋值到新的数组
    if A[p] ≠ 0
        L[i] ← A[p]
        i ← i + 1
return L
```

2. 找出两个数公共素因子的细化

假设得到了 m，n 的素因子，可以把它们按小到大的次序，用一个队列串起来。例如当 m = 120，n = 60 时，队列如图 1-3 所示。

如何才能找出它们公共的素因子呢？可以把 m 的队列作为参考，遍历 m 与 n 的素因子队列，并按照下面规则移动：

（1）如果 $p_m = p_n$，队列 m 与队列 n 同时移动到下一个节点，记录 p 值。

（2）如果 $p_m < p_n$，队列 m 移动到下一个节点。

（3）如果 $p_m > p_n$，队列 n 移动到下一个节点。

如上面例子，按照这样的规则可以得到它们的公共素因子为 2、3、5。比较过程如图 1-4 所示。

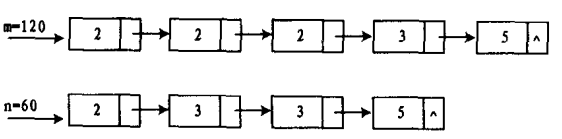

图 1-3

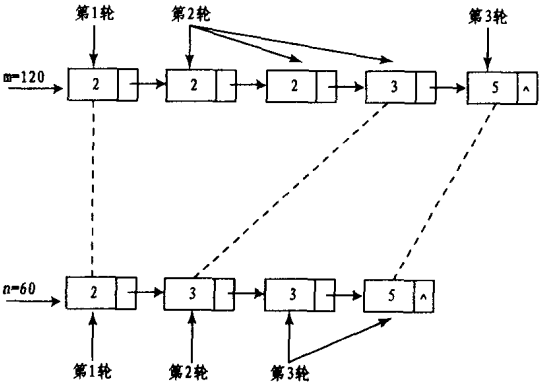

图 1-4

至此，这里把在中学所学到的求两个数的最大公约数的方法细化成为计算机可以实现的算法（当然，Sieve 算法需要使用求$\lfloor\sqrt{n}\rfloor$的函数）。

然而回过头来比较两个算法，显然辗转相除法无论从可操作性还是从效率上来说都优于中学阶段所学的方法。至于如何评价两个算法的好坏将会在本书算法分析部分介绍。

1.2 常见问题的类型

在日常生活或者平时的程序设计中出现问题的类型会很多，其中有几类问题是特别重要的，也是必须关注的，这里对每个问题给予概要的说明。后面的章节基本上也是按问题的类型划分的，因此每一类问题的具体例子可以参看后面相关的章节。

主要的问题有以下类型：

（1）排序问题（Sorting）。

（2）搜索问题（Searching）。

（3）字符串问题（String problems）。

（4）图论问题（Graph problems）。

（5）组合数学问题（Combinatorial problems）。

（6）几何问题（Geometric problems）。

（7）数值计算问题（Numerical problems）。

（8）加密问题（Encrpytion problems）。

1.2.1 排序问题

所谓的“排序”就是把一组杂乱的数据按照一定的规律顺次排列起来。在日常生活中，特别是在事务处理上，经常需要把一系列的数字按照大小顺序，把一系列的字符或者字符串按照字母表次序排列起来。

排序是很重要的，因为它可以使得以后的工作更加的容易。设想一下，如果平时所用的电话本或者通讯录是杂乱无章的，那么要从大量的记录里面查询一个人的资料将是一件费力的事情。

目前，计算机科学家发明了很多的排序算法，然而这些排序算法中并不存在一种算法是绝对最优的。不同的环境对排序算法有不同的要求，其中某些算法可能很高效，但是很复杂；有些可能更加适合无序的输入而有些则仅仅适合那些已经预先排序的输入。

这里首先介绍几个与“排序”相关的重要概念。

（1）数据表（Data List）：它是待排序数据对象的有限集合。例如学生的考试成绩，通讯录的姓名地址信息等都属于数据表。

（2）关键字（Key）：数据通常由多个数据成员组成，其中有一个数据成员是作为排序的依据，则称这个数据成员（或者称为属性域）为关键字。一个数据对象可能有很多的关键字，因此确定一个排序的关键字是很重要的，否则排序也无法进行。例如假设要排序的对象是学生的考试成绩，里面包含了学生的学号和语文、数学、英语三门考试成绩。那么排序前首先要确定应该按照哪个属性进行排序，是按照学号还是单科成绩，或者是平均分，这个依据就是前面所说的关键字，不同的关键字得到的排序结果往往是不相同的。

（3）主关键字（Primary Key）：如果数据表中各个对象的关键字互不相同，则称这种

关键字为主关键字。某一种排序方法如果按照主关键字进行排序，则排序结果是惟一的。

如果数据表中存在关键字相同的数据对象，则称这种关键字为次关键字。按照次关键字进行排序，排序的结果可能是不惟一的。

例如上面提到的学生成绩的排序例子，由于学号是惟一的，是主关键字，因此如果按照学号排序，则排序结果是惟一的。如果按照学生的成绩排序，由于两个人的成绩有可能相同，所以成绩是次关键字，因而可能导致排序结果不惟一。

（4）排序的稳定性：如果数据表中的两个对象 X[i]与 X[j]，它们的关键字 K[i]=K[j]，且排序前 X[i]排在 X[j]前面（即 i<j）；如果排序后，X[i]仍排在 X[j]前面，那么称这种排序方法是稳定的，否则称这种排序算法是不稳定的。

例如还是考虑上面的学号与成绩的排序。如果学号为 3 的同学的成绩与学号为 5 的同学的成绩是相同的，且输入计算机顺序是按照学号的先后次序。现在按照成绩从高到低排序，如果使用某一种排序方法排序后，学号为 3 号的成绩仍然排在学号为 5 号的成绩前，那么这种排序方法是稳定的。

注意：排序的稳定性指的是排序方法的稳定性，这是由排序方法本身所决定的。

后面的章节将会提到，一般来说，不稳定排序算法的效率比较高，但是有时候问题要求所应用的排序方法必须是稳定的。例如对上面的成绩进行排序时，如果还有班级的这一属性，需要先按班排序，然后每班再按成绩排序，那么在对成绩排序时，所选择的排序方法就必须是稳定的（这里假设所有的排序在一张表上进行）。

（5）内排序与外排序：排序方法根据排序过程中数据对象是否完全在内存中可以分为内排序和外排序。如果排序期间数据对象都存放在内存中，这种排序方法称为内排序。如果排序过程中需要把部分数据暂时存放在硬盘缓冲区才能完成的排序方法，称为外排序。这样区分的原因在于对于外排序效率的衡量，除了考虑算法的效率外（如排序交换次数等），还需要考虑硬盘的存取开销问题。

1.2.2 搜索问题

所谓搜索，就是在数据集合中寻找满足某种条件的数据对象。一般的形式是先给出一个特定值，然后在数据集合中找出关键字等于该特定值的对象。

对于搜索问题，通常需要返回的结果可能有两种。一种是简单的搜索成功或者失败的信息；另一种是返回满足搜索条件的对象所在位置，如果找不到则返回不存在信息。

对于搜索问题，同样没有一种最好的办法，不同环境的应用有不同的算法要求。其中有一点是需要重视的，就是待搜索的数据结构是不变的还是变化的。如果只是在一个数据集合找出某个数据，这种情况是很简单的。但是如果搜索过程还伴随着插入与删除的操作，那么就需要考虑怎样组织数据才能够适应数据集合的变化。

1.2.3 字符串问题

对于字符串处理的问题，这里只是对其中一类特殊的问题进行讨论，即字符串模式匹配问题（string matching）。问题是这样表述的：在一篇文章或文章的一部分中找出单词第一次出现的位置。

本书把字符串匹配问题作为一种特殊的搜索问题考虑，因为两者是有相似之处的。最

大的区别在于，一般搜索问题是单一的关键字的搜索，而字符串匹配可以看作是几个关键字组成一种模式的搜索过程，因此对模式的分析有助于算法效率的提高。

1.2.4 图论问题

图论问题主要是与图或树相关的问题。这里所谓的“图”指的是一些顶点（Vertex）与边（Edge）的集合。图有着非常重要与丰富的实际应用，如通信网络、交通网络、工程计划等。本书所涉及到的图论问题主要包括了图的遍历问题（Graph Traversal）、最短路径问题以及图的特例——树结构的排序、搜索中的应用问题。

学习这一部分的前提是掌握了一些最基本的图的概念，本书也会在后面提供一些相关的知识。

1.2.5 组合数学问题

事实上，图论的一些问题其实也是组合数学问题的范畴，例如图论中的着色问题、Hamilton 圈问题以及旅行家问题等。组合数学问题主要是关于排列组合，或者求满足某种约束的子集的一系列问题。

传统的组合问题常常以一种游戏的形式出现，如 Bachet 的法码问题、Kirkman 的女生问题和 Euler 的 36 名军官的问题等等，都是这方面经典的例子。而现代的组合数学所涉及的领域更加广阔，其中的问题出自抽象代数、拓扑学、数学基础、图论、博弈论、线性规划以及其他许多领域。值得注意的是，这些来源不同的问题不能在一个统一的理论体系中得到有效地解决，而且随着问题的输入量的增加，问题的规模急剧增长至计算机的处理能力的极限，所以可以说组合数学问题是最难的一类问题。

目前大部分的计算机科学家认为类似于旅行家问题、Hamilton 圈问题等的最有效算法理论上是不存在的，虽然他们不能证明这一点。

1.2.6 几何问题

几何问题是几何数学上与点、线、多边形相关的问题。当然，这不是一本几何学的书，所以这里只会涉及如直线段的表示、相交判断等基本问题。同时还会给出最近邻点问题与凸包问题这两个重要问题的介绍。它们都有穷举法与分治法的算法版本，通过比较，可以深入体会到其中所用到的算法设计思想。

1.2.7 数值计算问题

数值计算问题是另一大类问题，它要解决的问题包括：求解方程或者方程组和求数值积分等。这些问题常常只能给出一个近似的结果，而不是精确的解析答案。并且这些问题常涉及到具体的数据，由于计算机对数字表示的精度问题，所以还需要考虑可能带来的误差以及这些误差对算法稳定性的影响。

1.2.8 加密问题

加密问题是信息安全的基础，它要解决的问题包括：通信双方的相互鉴定、消息完整性的确定以及公开密钥的分发等。这些安全问题的解决方案依赖于加密算法的发展，如 DES