Discrete Mathematics

离散数学

李 锋 王高丽 苏厚勤◎编著

- 降低理论环节的要求，减少一些繁琐的证明
- 增加应用性内容，提供所讲述知识点的应用实例
- 增加算法章节，在许多概念和定理中加入算法说明
- 每章安排上机练习题，便于实践课程的开展

清華大學出版社

离 散 数 学

李 锋 王高丽 苏厚勤 编著

清华大学出版社

北 京

内 容 简 介

离散数学是计算机专业和部分信息类专业的必修课程，也是IEEE的教育委员会建议的计算机专业本科必修课程。为适应当前教育需求，本书降低了理论环节的难度，强调实践及应用环节。

全书共分为8章，分别介绍了算法基础、集合与序列、关系与函数、逻辑与证明、图论、网络模型、代数系统以及有限状态机和图灵机等内容。为了便于学生上机编写程序，本书在编写的过程中增加了算法章节。

本书可作为高等工科院校计算机专业和信息类专业的教材，也可供高等职业技术学院、高等工业专科学校及其他大专院校的师生以及编程人员参考使用。

图书在版编目（CIP）数据

离散数学/李锋，王高丽，苏厚勤编著. —北京：清华大学出版社，2011.2

ISBN 978-7-302-24647-3

I. ①离… II. ①李… ②王… ③苏… III. ①离散数学 IV. ①O158

中国版本图书馆CIP数据核字（2011）第004932号

责任编辑：朱英彪
封面设计：刘　超
版式设计：侯哲芬
责任校对：张彩凤
责任印制：杨　艳
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦A座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：185×230　**印　张**：8.75　**字　数**：173千字
版　　次：2011年2月第1版　　**印　　次**：2011年2月第1次印刷
印　　数：1～4000
定　　价：22.00元

产品编号：035311-01

前　言

离散数学是计算机类专业和部分信息类专业的必修课程，也是 IEEE 的教育委员会建议的计算机专业本科必修课程。

计算机领域解决问题的一般步骤为：提出问题—建立数学模型—运算对象表示—设计算法—编写代码（如图 1 所示）。在这个过程中，需要用到很多数学知识。由于数字电子计算机只能处理离散的或离散化了的数量关系，因此，无论计算机科学本身，还是与计算机科学及其应用密切相关的现代科学研究领域，都面临着如何解决为离散结构建立相应的数学模型，又如何将已用连续数量关系建立起来的数学模型离散化，从而可由计算机加以处理的问题。

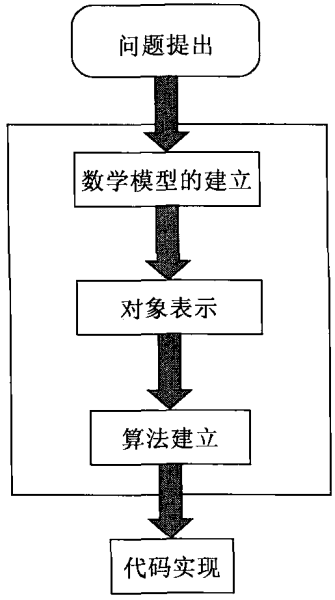

图 1　计算机领域解决问题的一般步骤

离散数学（Discrete Mathematics）是计算机专业的一门重要基础课，它所研究的对象是离散数量关系和离散数学结构模型，主要介绍计算机相关数学领域各分支的基本概念、基本理论和基本方法。这些概念、理论和方法今后会大量地应用在数字电路、编译原理、

数据结构、操作系统、数据库系统、算法的分析与设计、人工智能、计算机网络等专业课程中。同时，该课程所提供的训练十分有益于提高学生的概括抽象能力、逻辑思维能力和归纳构造能力，也有益于培养学生严谨、完整、规范的科学态度。

在计算机研究和应用领域，涉及到的数学知识很多，而一门课程的学时是有限的，所以离散数学也只能涉及到部分内容，不同教材的内容也有所差别，一般包括数理逻辑、代数系统、图论等内容。

计算机是一门实践性很强的学科，离散数学也应以应用为基础，强调其在解决实际问题中的作用，因此在离散数学的学习中，在培养学生理论知识的同时，更要注重培养学生应用数学知识解决实际问题的能力。但多年来无论是教材编写还是授课，大多仍偏重于抽象的理论和概念，虽然强调“离散数学课程所传授的思想和方法，广泛地体现在计算机科学技术相关专业的各个领域”，但在实际教材中应用型的例子却很少，学生往往是为理论而学理论，懂理论而不会应用，学生在课程中学习到的知识点，无法应用到具体的任务中，教材缺乏必要的实践内容，教学缺乏必要的实践环节。

近些年来，我国高中的数学教学内容发生了很多变化，而大学的《离散数学》教学内容却没有与其进行很好的衔接。例如，集合和函数是高中数学的重要内容，是本科学生已经掌握了的内容，可是多年来，本科离散数学课程中仍一直把集合和函数的基本理论和基本操作当作一个重要的教学内容，造成了教学资源的重复和浪费。鉴于此，在吸收其他优秀教材编写经验的基础上，我们编辑出版了现在这本《离散数学》，该教材的主要特点在于：

（1）强调实践即应用环节。在内容上，本书增加了应用性内容，介绍了一些理论和所讲述知识点的应用实例。让学生体会到理论的实用价值，以提高其学习的积极性。

（2）降低了理论环节的难度要求，减少了一些繁琐的证明，对于理论部分，只要求学生理解即可。即使我们强调一大堆推理、证明和定理，学生也只是在上课和考试时翻看，在以后的学习和工作中很少应用，很快就会忘记。

（3）为便于学生上机编写程序，本书专门增加了一章算法基础，主要让学生理解算法的定义和编写，便于后面章节的实习。

（4）矩阵是研究关系理论、图论等内容的重要工具，所以本书增加了矩阵的简单介绍及其运算实现方法。

（5）在各个章节中增加了算法编写部分，在习题中也专门设置了计算机编程习题，使得部分知识点能由学生自己动手实现，加深了学生对理论的理解，提高其学习兴趣。

（6）对集合和函数的相关内容进行了必要的调整，减少了集合论、函数定义和一般性质说明，增加了集合运算和函数的计算机实现算法说明。

下面简单介绍一下本书各章节的主要内容和难度要求。

第 1 章：算法基础。包括算法的表示和编写方法，为后面章节的算法理解打下基础。

第 2 章：集合与序列。主要讲解集合及其元素的计算机实现算法，另外对计算机常用的序列和串，以及子序列的概念和判定算法也给了说明。本章对于集合的定义和一般运算这些高中已经学过的内容，只进行了简单的介绍，重点在于介绍集合运算的计算机实现算法，还介绍了矩阵的概念、矩阵的加法和乘法的定义和实现算法，这些内容在后面的关系、图论、网络模型等章节中都有应用。

第 3 章：关系与函数。关系是研究个体间联系的重要工具，本章主要介绍了关系的定义、分类和性质，并给出了对关系性质判定的算法，对于常用的等价关系和次序关系，书中也作了主要介绍，给出了根据等价关系对集合进行划分的算法；函数是一种特殊的关系，其基本概念在高中已经讲授过，所以本章涉及内容相对较少，主要从关系的角度对函数进行了阐述。

第 4 章：逻辑与证明。逻辑是信息处理科学的重要工具，本章减少了一般形式逻辑学中的理论证明和繁琐的推理环节，主要强调形式逻辑和自然逻辑的转换、简单逻辑推理等内容。

第 5 章：图论。这是一个重点章节，本教材减少了传统图论中的一些定理和证明内容，主要以图的计算机表示、存储、应用，以及一些算法的实现等内容为主。

第 6 章：网络模型。本章所讲的网络模型是指运输图，也是计算机网络、城市交通网、排水网等网络的数学模型。网络是特殊的图，本章主要介绍了网络的基本概念和最大流量算法。

第 7 章：代数系统。本章主要介绍代数系统的基本概念和几类主要的代数系统——群、环、格。这部分内容是离散数学中的一个难点，在本书中，对其进行了简化，主要介绍一些最基本的概念和定义，在保证系统知识点一致的情况下，基本省略了额外定理和证明。

第 8 章：有限状态机和图灵机。本章是选学内容，只要求一般性的了解，主要是对确定性有限状态机、非确定性有限状态机和图灵机作一些说明性介绍。

另外，由于树是数据结构的重要内容，所以在本教材中没有涉及到。书中的每一章节都设计了上机实习题，便于大家利用所学知识自己动手解决实际问题。

在本书即将付梓之际，感谢清华大学出版社为本书出版所付出的艰辛劳动。由于水平有限，书中难免存在不足之处，恳请读者不吝赐教。

作者的联系方式是 lifeng@dhu.edu.cn。

目　录

第1章 算法基础

1.1 算法简介

算法（Algorithm）是一系列解决问题的清晰指令，也就是说，能够对一定规范的输入，在有限时间内获得所要求的输出。如果一个算法有缺陷，或不适合于某个问题，执行这个算法将不会解决问题。

算法（即演算法）的中文名称出自《周髀算经》，英文名称 Algorithm 则出自 9 世纪的波斯数学家 al-Khwarizmi。Algorithm 原写为 algorism，意思是阿拉伯数字的运算法则，于 18 世纪演变为 algorithm。欧几里得算法被人们认为是史上第一个算法。第一个程序是 Ada Byron 于 1842 年为巴贝奇分析机编写的求解伯努利方程的程序，因此 Ada Byron 被大多数人认为是世界上第一位程序员。但是因为查尔斯·巴贝奇（Charles Babbage）未能完成他的巴贝奇分析机，这个程序最终也未能在巴贝奇分析机上执行。因为对“well-defined procedure”定义严谨的过程缺少数学上的精确定义，19 世纪和 20 世纪早期的数学家、逻辑学家在定义算法上出现了困难。20 世纪的英国数学家图灵提出了著名的图灵论题，并提出一种假想的计算机抽象模型，这个模型被称为图灵机。图灵机的出现解决了算法定义的难题，图灵的思想对算法的发展起到了重要作用。

如果没有人们编制的软件去指挥计算机工作，计算机将一无所用。软件是计算机的灵魂，而软件的核心是算法。用计算机解决问题的方法就是算法，计算机是执行算法的机器，让计算机解决各种问题主要就是创造各种算法。

计算机科学家尼克劳斯·沃思写过一本著名的书《数据结构+算法=程序》，可见算法在计算机科学界与计算机应用界的地位。

算法可以理解为由基本运算及规定的运算顺序所构成的完整的解题步骤，或者看成按照要求设计好的、有限的、确切的计算序列，并且这样的步骤和序列可以解决一类问题。

下面以数学家华罗庚的“泡茶”为例，来说明算法的特性。

初始情况：没有开水，开水壶没洗，茶壶、茶杯没洗，火已生了，有茶叶。

最终情况：用开水泡茶喝。

解决方法：洗开水壶，洗茶壶，洗茶杯，将茶叶放入茶壶，用开水壶烧开水，水开后泡茶喝。

一个算法应该具有以下 4 个重要的特征：

- 有限性：也叫可终止性，指一个算法必须保证在执行有限步之后结束。
- 确切性：算法的每个步骤都必须有确切的定义，不能有类似于“执行语句 A 或语句 B”的指令。
- 输入：一个算法有 0 个或多个输入，以刻画运算对象的初始情况。所谓 0 个输入是指算法本身已定义了初始条件。这里，不要刻板地认为输入一定是键盘的输入，也可以是传递的参数、文件或数据等。
- 输出：一个算法有 1 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。

1.2 算 法 表 示

为了表示一个算法，可以采用各种各样的方式，包括自然语言、图形和伪代码。

1. 自然语言

用自然语言描述“A、B 为整数，求 A 除以 B 的余数”这一算法。结果如下：

首先，获得 A 和 B 的值。

然后，判断 B 是否为零。

如果 B 为零，则说明这是非法的除法运算，无法得到余数，输出“除数为零”的出错信息；如果 B 不为零，那么就可以通过除式计算，取得余数。

最后，输出计算出的余数。

用自然语言能够表示一些简单的算法，但对于一些复杂的算法，使用自然语言来描述容易出现“二义性”，或者必须用冗长的语言才能说清楚。

2. 图形

除了自然语言外，在描述算法时还可以采用图形来更精确地表示。其中，流程图是一种常用的图形算法表示法，它是人们经常用来描述算法的工具。流程图是用规定式样的图形、指向线和文字说明组合起来表示算法的。

下面用流程图来描述“A、B 为整数，求 A 除以 B 的余数”这一算法，如图 1.1 所示。

流程图等图形法存在书写复杂的问题，因此，现在常用的是伪代码法。

3. 伪代码

伪代码（Pseudocode）是一种算法描述语言。使用伪代码的目的是为了使被描述的算法可以非常方便地以任何一种编程语言（如 Pascal、C、Java 等）实现。因此，伪代码必

须结构清晰、代码简单、可读性好，并且类似自然语言。

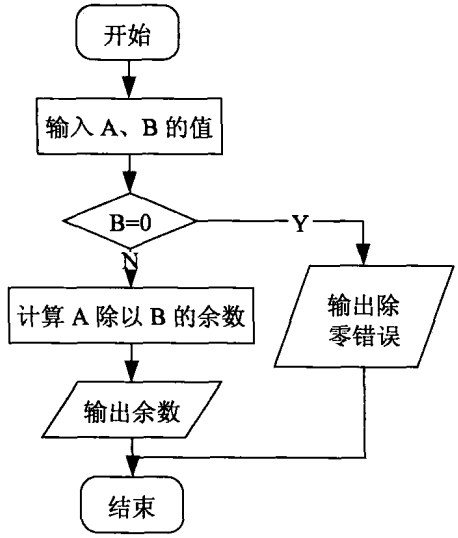

图 1.1　算法流程图

人们在用不同的编程语言实现某个算法时发现，不同语言环境下的实现方法（注意：这里是实现，不是功能）会有所不同。这对于那些只熟知某一种编程语言的程序员来说，要理解一个用其他编程语言编写的程序会比较困难，因为程序语言的形式限制了程序员对程序关键部分的理解。这样伪代码就应运而生了。

当考虑算法的功能而不是其语言实现时，常常应用伪代码，在计算机科学的教学中也通常使用伪代码，以使所有的程序员都能理解。

伪代码介于自然语言与编程语言之间，以编程语言的书写形式指明算法的职能。相比于程序语言（如 Java、C++、C、Dephi 等），它更类似于自然语言，是一种半形式化的、不标准的语言。可以将整个算法运行过程的结构用接近自然语言的形式（可以使用任何一种你熟悉的文字，关键是把程序的意思表达出来）描述出来。使用伪代码可以帮助我们更好地表述算法，而不必拘泥于具体的实现和语法。

例如，类 Pascal 语言的伪代码的语法规则是：在伪代码中，每一条指令占一行（else if 例外），指令后不跟任何符号（Pascal 和 C 中，语句要以分号结尾）。书写上的缩进表示程序中的分支程序结构，这种缩进风格也适用于 if-then-else 语句。用缩进取代传统 Pascal 中的 begin 和 end 语句来表示程序的块结构可以大大提高代码的清晰性：同一模块的语句有相同的缩进量，次一级模块的语句相对于其父级模块的语句缩进。

伪代码只是像流程图一样用在程序设计的初期，帮助写出程序流程。编程序是把流程写下来，从总体上考虑整个功能如何实现。写完以后不仅可以作为以后测试、维护的基础，还可用来与他人交流。但是，如果把全部的东西写下来必定会浪费很多时间，那么这个时候可以采用伪代码方式，例如：

```
if 九点以前 then
do 私人事务;
else if 9 点到 18 点
  then
    工作;
 else
    下班;
```

这样不但可以达到文档的效果，同时可以节约时间，更重要的是使结构比较清晰，表达方式更加直观。因为大家对 C 语言比较熟悉，所以这里采用了类 C 语言的伪代码表示法。

使用伪代码表示算法，主要强调的是逻辑的表达，对语法要求较低。

例 1.2.1 算法：在整数 iData1、iData2、iData3 中寻找最大数。

解：

```
//在整数 iData1、iData2、iData3 中寻找最大数
FindMaxData(int  iData1，int  iData2，int  iData3)
{
   X=iData1;
   If  iData2＞X
      X=iData2;
   If  iData3＞X
      X=iData3;
   Return X
}
```

1.3 算法分析

阅读一个算法，理解该算法实现的功能和思路，称为算法跟踪。算法跟踪是一个计算机专业人士要掌握的基本功。针对例 1.2.1，可以看出其主要思想是用一个临时变量 X

来记录程序运行过程中找到的最大数。

同一个问题可以用不同的算法来解决，而一个算法的优劣将影响到算法乃至程序的运行效率。算法分析的目的在于选择最合适的算法和不断改进算法。对一个算法的评价主要从时间复杂度和空间复杂度来考虑。

算法的时间复杂度，是指算法需要消耗的时间资源。一般来说，计算机算法的时间复杂度是问题规模 n 的函数 $f(n)$；算法的空间复杂度，是指算法需要消耗的空间资源，其计算和表示方法与时间复杂度类似，一般都用复杂度的渐近性来表示。同时间复杂度相比，空间复杂度的分析要简单得多。

在一般的算法分析中，多以时间复杂度分析为主，但执行一个算法所耗费的时间，从理论上是很难计算出来的，必须上机测试后才能知道。但我们不可能也没有必要对每个算法都进行上机测试，而是只要知道哪个算法花费的时间多，哪个算法花费的时间少即可。一个算法花费的时间与算法中语句的执行次数成正比，算法中语句执行次数多，它花费的时间也就多。

一个算法中的语句执行次数称为语句频度或时间频度，记为 $f(n)$。其中，n 称为问题的规模，当 n 不断变化时，时间频度 $f(n)$也会不断变化。但有时只想知道它变化时呈现什么规律。一般情况下，算法中基本操作重复执行的次数是问题规模 n 的某个函数，用 $f(n)$表示，若有某个辅助函数 $T(n)$，使得当 n 趋近于无穷大时，$T(n)/f(n)$的极限值为不等于零的常数，则称 $T(n)$是 $f(n)$的同数量级函数，记作

$$T(n)=O(f(n))$$

$O(f(n))$称为算法的渐进时间复杂度，简称时间复杂度。

在各种不同的算法中，若算法中语句的执行次数为一个常数，则时间复杂度为 $O(1)$。另外，在时间频度不相同时，时间复杂度有可能相同，如 $f(n)=n^2+3n+4$ 与 $f(n)=4n^2+2n+1$ 的频度不同，但时间复杂度相同，都是 $O(n^2)$。

按数量级递增排列，常见的时间复杂度有：

- 常数阶 $O(1)$。
- 对数阶 $O(\log_2 n)$。
- 线性阶 $O(n)$。
- 线性对数阶 $O(n\log_2 n)$。
- 平方阶 $O(n^2)$。
- 立方阶 $O(n^3)$。
- k 次方阶 $O(n^k)$。
- 指数阶 $O(2^n)$。

随着问题规模 n 的不断增大，时间复杂度不断增大，算法的执行效率降低。

例 1.3.1

```
for i=1 to n
    x=x+1
```

x=x+1 执行次数数量级函数为 $O(n)$。

例 1.3.2

```
for i=1 to n
  for j=1 to n
    x=x+1
```

x=x+1 执行次数数量级函数为 $O(n^2)$。

习　　题

1．写出在 n 个整数中寻找最小整数的算法。

2．写出在 n 个整数中寻找最大整数的算法。

3．写出在 n 个整数中，判定是否有相等的数的算法。

4．写出针对一个字符串，统计出各个字母出现次数的算法。

5．写出判定一个数是奇数还是偶数的算法。

6．给定一个整数数组 $S=s_1, s_2, s_3, \ldots, s_n$ 和一个关键整数 key。写出在 S 中找出 key 的位置的算法。如果 key 在 S 中不存在，则输出 0。

7．判断下列程序中 $x=x+1$ 执行次数的 $O(f(n))$。

（1）
```
for i=1 to 2n
    x=x+1
```

（2）
```
i=1
while (i<=2n)
{
  x=x+1
  i=i+2
}
```

（3）
```
for i=1 to n
  for j=1 to n
    x=x+1
```

（4）for i=1 to n
　　for j=1 to i
　　x=x+1
（5）i=n
　　while (i>=1)
　　{
　　for j=1 to n
　　x=x+1
　　i=[i/2]　//小于 i 除以 2 的最大整数
　　}

上 机 习 题

1．实现在 *n* 个整数中寻找最小整数的算法。
2．实现在 *n* 个整数中寻找最大整数的算法。
3．实现在 *n* 个整数中，判定是否有相等的数，并输出该数。
4．实现在一个字符串的反向输出：
　输入：$N=n_1n_2n_3n_4n_5$
　输出：$N=n_5n_4n_3n_2n_1$
5．实现习题 4 的算法。
6．实现习题 5 的算法。
7．实现习题 6 的算法。

第2章 集合与序列

2.1 定义和运算

2.1.1 集合的定义

集合论是数论的基础，在初等数学里，大家已经有所了解。

集合就是具有共同属性的对象的全体，其中每一个对象称为集合的元素。集合中元素的个数称为集合长度。集合中的元素是无序的、不可重复的。

集合的基本运算有并、交、补、差，它们的定义分别如下。

定义 2.1.1 设 A、B 是两个集合，E 是全集，则 A 与 B 的并集 $A \cup B$、A 与 B 的交集 $A \cap B$、A 的补集 $\overline{A}$ 分别定义如下：

$$A \cup B = \{x | x \in A \text{ 或 } x \in B\}$$

$$A \cap B = \{x | x \in A \text{ 且 } x \in B\}$$

$$\overline{A} = \{x | x \notin A \text{ 且 } x \in E\}$$

由定义可以看出，$A \cup B$ 是由 A 或 B 中的元素构成的；$A \cap B$ 是由 A 和 B 中的公共元素构成的；$\overline{A}$ 是由全集中但不在 A 中的其他元素构成的。

定义 2.1.2 设 A、B 是两个集合，A 与 B 的差 $A-B$ 定义为：

$$A-B = \{x | x \in A \text{ 且 } x \notin B\}$$

由定义可以看出，$A-B$ 是由 A 中但不属于 B 的元素构成的。

2.1.2 集合运算的实现

集合在计算机领域有着广泛的应用，而其在计算机语言中的表示方法根据语言的不同有所差别，一般是以数组的形式存储，但计算机语言中的数组和集合是有一定区别的，在集合中，元素没有顺序差别，但在数组中，元素是有顺序区分的。用数组表示集合时，不需要考虑元素的顺序，但要保证元素没有重复。

一个集合中往往含有大量的元素，集合的运算要用计算机来完成，下面就给出集合基本运算的算法。

算法 2.1.1 求两个集合交集的算法。