计 算 机 科 学 丛 书

计算理论导引

Introduction to the Theory of Computation

(美) Michael Sipser 著
麻 省 理 工 学 院
张立昂 王捍贫 黄 雄 译

机械工业出版社
China Machine Press

计算理论导引
Introduction to the
业出版社

计算机科学丛书

计 算 理 论 导 引

(美) Michael Sipser 著
(麻 省 理 工 学 院)

张立昂 王捍贫 黄雄 译
(北京大学)

机械工业出版社
China Machine Press

本书系统地介绍了计算理论的三个主要内容：自动机与语言、可计算性和计算复杂性。绝大部分内容是基本的，同时对可计算性和计算复杂性理论中的某些高级内容作了重点介绍。作者以清新的笔触、生动的语言给出了宽泛的数学原理，而没有拘泥于某些低层次的细节。本书可作为计算机专业高年级本科生和研究生的教材，也可作为教师和研究人员的参考书。

本书版权登记号：图字：01－1999－2360

图书在版编目（CIP）数据

计算理论导引/（美）西普塞（Sipser, M.）著；张立昂，王捍贫，黄雄译 .—北京：机械工业出版社，2000.2
（计算机科学丛书）
书名原文：Introduction to the Theory of Computation
ISBN 7－111－07574－9

Ⅰ.计… Ⅱ.①西…②张…③王…④黄… Ⅲ.算法理论 Ⅳ.0241

中国版本图书馆 CIP 数据核字（1999）第 56458 号
机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）
责任编辑：陈贤舜
北京牛山世兴印刷厂印刷·新华书店北京发行所发行
2000年2月第1版·2000 年12月第2次印刷
787mm x 1092 mm 1/16·17.75印张
印数：5 001~6 000册
定价：30.00元

译 者 序

M.Sipser 著的《Introduction to the Theory of Computation》是关于计算理论的基础教材，在美国的大学计算机系中得到广泛的使用，我们希望这个译本对我国高等院校的计算机教育能有所帮助。

本书的内容分三个部分：自动机与语言、可计算性和计算复杂性理论。绝大部分内容是基本的，对可计算性和计算复杂性理论中的某些高级内容作了重点介绍（第 7 章和第 11 章）。本书的一个重要写作特点是不仅让读者“知其然”，而且极力使读者“知其所以然”。在叙述概念的形式定义之前，总是先给出问题提出的背景、概念的直观含义、提出这个概念的初衷以及它在实际中的应用。在定理的证明之前，差不多都有“证明思路”，给出定理证明的直观想法和说明在进行严格的形式证明时需要注意的地方或可能遇到的障碍。这些“新鲜的”思想对读者肯定是很有裨益的。

我们通过网络下载了本书作者维护的勘误表，并将其指出的错误在译本中全部改正。另外，在翻译过程中也发现了少量的其他错误，都予以纠正。对原书索引中的主要术语，都以中英文对照形式给出。并为符合中文习惯，按汉字笔画对其重新进行了排序。

第 1 章、第一部分的全部以及第三部分的第 11 章由张立昂翻译，第二部分由王捍贫翻译，第三部分的第 8～第 10 章由黄雄翻译。

由于译者水平所限，译文难免有错误和不妥之处，恳请读者批评指正。

译 者

1999 年 8 月

译者介绍

张立昂，1941 年 2 月出生，1965 年毕业于北京大学数学力学系数学专业。现为北京大学计算机科学与技术系教授、博士生导师。主要研究方向：算法设计与分析、计算复杂性理论。著作有《可计算性与计算复杂性导引》等。

王捍贫，1964 年 7 月出生，1993 年毕业于北京师范大学数学系数理逻辑专业，获博士学位。现为北京大学计算机科学与技术系副教授。主要研究方向为数理逻辑、计算复杂性、程序语义及正确性验证技术。著作有《数理逻辑》等。

黄雄，1969 年 8 月出生，北京大学计算机科学硕士，北京航空航天大学计算机科学博士。现在中国科学院计算所做博士后。研究领域包括：算法设计与分析、计算复杂性和 Web 信息检索。

前　言

写给学生

欢迎使用本书！

你将开始学习的是重要而又引人入胜的一课：计算理论。它包括计算机硬件、软件以及某些应用的基本数学特性。通过学习这一科目，我们试图解决什么是能计算的，什么是不能计算的，有多快，要用多少存储，以及采用什么计算模型等。该科目与工程实践有着明显的联系。同时，与许多学科一样，它也具有纯理论的一面。

我知道你们中有许多人盼望学习这门课程，而有些人可能不是这样想的。你可能想得到计算机科学或者计算机工程的学位，而一门理论课程是必需的——天晓得为什么要这样。难道理论终究不是神秘的、令人厌烦的吗？而且最坏的是，不是与我毫不相干的吗？

读下去你会发现理论既不神秘、也不令人厌烦，而是相当好理解、甚至是有趣的。理论计算机科学有许多迷人而重要的思想，同时它也有许多细小的、有时是枯燥乏味的细节，这些细节可能令人感到厌倦。学习任何一门新的科目都是一件艰苦的工作。但是，如果能把它适当地表述出来，学习就会变得容易和更愉快些。写这本书的基本目标是让你接触到计算理论中真正令人激动的方面，而不要陷入单调乏味之中。当然，使你对理论感兴趣的唯一途径是努力去学习并掌握它。

理论与实践是相关联的，它为实际工作者提供了在计算机工程中使用的理性工具。要为具体的应用设计一个新的程序设计语言吗？你在本课程中学习的关于语法的内容迟早是会有用的。要进行字符串搜索和模式匹配吗？不要忘了有穷自动机和正则表达式。遇到了一个看来需要比你能够提供的计算机时间还要多的问题吗？想一想你学过的有关 NP 完全性的内容。各种应用领域，如现代密码协议，依赖于你在这里将要学习的理论原则。

理论对你也是有意义的，因为它向你展示了计算机新的、简单的、更加优美的一面，而通常我们把计算机看作一台复杂的机器。最好的计算机设计和应用出自完美的构思。一门理论课程可以提高你的审美意识，帮助你建立更加优秀的系统。

最后，理论是个好东西，因为学习理论能够扩展你的思维。计算机技术变化很快，专门的技术知识虽然今天有用，但是仅仅在几年内就会变成过时的东西。应该代之以培养思考的能力、清楚而准确地表达自己的能力、解决问题的能力、以及知道问题什么时候还没有解决的能力。这些能力具有持久的价值。学习理论能使你在这些方面得到训练。

除了实际的考虑，几乎每一位使用计算机的人都想了解这个神奇的创造，它的能力，以及它的局限性。为了解答某些基本问题，在过去的 30 年里，一个全新的数学分支已经确立。这里还有一个重大问题没有解决：如果我给你一个大的自然数，譬如说有 500 位，你能够在合理的时间内把它分解成素数的乘积吗？哪怕是使用一台超级计算机，现今还没有人知道怎样才能在宇宙毁灭之前做完这件事！因子分解问题与现代密码系统中的某些密码有关。去寻找一个快速的因子分解方法吧，这样你就会一举成名！

写给教师

本书打算作为计算机学科高年级本科生教材或研究生的介绍性教材。它涉及本科目的数学论述，包括定理和证明。我努力使本书适应那些缺乏定理证明经验的学生，当然具有较多这种经验的学生将会轻松得多。

在叙述本书的内容时，首要的目的是清楚和生动。因此，本课程对某些低层次的细节强调了直觉和“大的轮廓”。

例如，尽管我在第 1 章介绍了证明的归纳法以及其他的数学预备知识，但在后面部分它并不是重点。关于自动机的各种构造方法的正确性，我一般不叙述常用的归纳证明。只要叙述得清楚，这些构造方法是令人信服的，不需要进一步的论证。归纳证明可能会把人搞胡涂而不是给人以启迪，因为归纳法本身是相当复杂的技术，可能还有些神秘。用归纳法对十分明显的事情作反复的说明有这样的危险：使学生认为数学证明是一种形式手法，而不是教给他们懂得什么是有说服力的证据，什么不是有说服力的证据。

第二个例子在第二和第三部分，在那里我用普通的语言、而不是用伪码描述算法。我没有花很多时间去设计图灵机（或任何其他形式模型）的程序。现在的学生都有程序设计的经历，觉得丘奇－图灵论题是不言自明的。因此我不去用很长的篇幅叙述用一个模型模拟另一个模型来说明它们的等价性。

除增加直观性和压缩某些细节外，本书可以称得上是对本科目内容的经典陈述。多数理论工作者将发现，素材的选取、术语以及内容的前后顺序都与其他广泛使用的教材一致。只在少数地方，当我发现标准的术语十分模糊不清或会引起混淆时，才引进了新的术语。例如，引进名词映射可归约性代替多一可归约性。学习任何数学科目，做题是必不可少的。本书把习题分成两大类，分别叫做练习和问题。练习用来复习定义和概念。问题需要多动些脑筋。带星号的问题更难一些。我努力使练习和问题令人感兴趣，并有挑战性。

反馈给作者

互联网为作者与读者之间的交流提供了新的机会。我收到很多电子邮件，对本书的初版提出了建议、赞许和批评，或者指出错误。请继续来函。只要有时间，我尽量亲自给每一个人回信。与本书有关的电子信箱是

sipserbook@math.mit.edu

另外，还有一个 Web 站点，包括一张勘误表。可能还有一些其他材料也要加入这个站点用来帮助教师和学生。请告诉我你希望在这里看到什么。这个站点的地址是

http://www-math.mit.edu/~sipser/book.html

Michael Sipser
Cambridge, Massachusetts
1996 年 10 月

目　录

第三部分 复杂性理论

第1章 导 引

首先，让我们概述一下在本课程中讲述的计算理论的那些领域，然后，你将有机会学习或者复习一些后面需要的数学概念。

1.1 自动机、可计算性与复杂性

本书集中在计算理论的三个传统的核心领域：自动机、可计算性和复杂性。下述问题把它们联系在一起：

计算机的基本能力和限制是什么？

这个问题要回溯到20世纪30年代，那时数理逻辑学家们首先开始探究计算的含义。自那时以来，技术进步极大地增强了我们的计算能力，并且把这个问题从理论王国带进现实世界。

在这三个领域（自动机、可计算性和复杂性）中的每一个领域内，对这个问题作了不同的解释，并且答案随解释的不同而各不相同。在本章之后，我们将把本书分成三部分，每一部分研究一个领域。在此，我们用相反的顺序介绍这几部分，因为从后面开始介绍能够更好地理解本书为什么这样安排顺序。

1.1.1 计算复杂性理论

计算的问题是各种各样的，有的容易，有的困难。例如，排序是一个容易的问题。譬如说需要按升序排列一张数表，即使一台小型计算机都能相当快地处理100万个数。与时间表问题比较一下。譬如说要制定一所大学的课程表，课程表必须满足某些合理的限制，如不能有两个班在同一时间使用同一教室。时间表问题看来比排序问题困难得多。如果有1000个班，即使是使用一台超级计算机，制定一份最好的课程表也可能需要若干世纪。

是什么使某些问题很难计算，又使另一些问题容易计算？

这是复杂性理论的核心问题。值得注意的是，虽然在过去的25年里对它进行了深入细致的研究，但是我们仍然不知道它的答案。以后我们将探究这个迷人的问题和它的一些分支。迄今为止，复杂性理论的一个重要成果是，发现了一个按照计算难度给问题分类的完美体系。它类似按照化学性质给元素分类的周期表。运用这个体系，我们能够提出一种方法给出某些问题是难计算的证据，尽管还不能证明它们是难计算的。

当你面对一个看来很难计算的问题时，有几种选择。首先，搞清问题困难的根源，你可能会做某些变动，使问题变得容易解决。其次，你可能会求出问题的不那么完美的解。在某些情况下，寻找问题的近似解相对容易一些。第三，有些问题仅仅在最坏的情况下是困难的，而在绝大多数的时候是容易的。就应用而言，一个偶尔运行得很慢、而通常运行得很快的过程是能够令人满意的。最后，你可以考虑其他类型的计算，如随机计算，它能加速某些工作。

受到复杂性理论直接影响的一个应用领域是密码技术，这是一个古老的研究领域。在绝大多数的情况下，容易计算的问题比难计算的问题更可取，因为求解容易问题的代价要小。密码技术与众不同，它特别需要难计算的问题，而不是容易计算的问题。在不知道密钥或口令时，密码应该是很难破译的。复杂性理论给密码研究人员指出了寻找难计算问题的方向，围绕这些问题已经设计出新的革命性的编码。

1.1.2　可计算性理论

在20世纪前半叶，数学家们，如哥德尔（Kurt Gödel）、图灵（Alan Turing）及丘奇（Alonzo Church），发现一些基本问题是不能用计算机解决的。确定一个数学命题是真是假就是这样一个例子。这项工作是数学家的生计。用计算机来解决似乎是天经地义的，因为这是数学王国里的事。但是，没有计算机算法能够完成这项工作。

关于计算机理论模型思想的发展是这一意义深远的结论的成果之一，它最终促使建造出实际的计算机。

可计算性理论与复杂性理论是密切相关的。在复杂性理论中，目标是把问题分成容易的和困难的；而在可计算性理论中，是把问题分成可解的和不可解的。可计算性理论介绍了若干在复杂性理论中使用的概念。

1.1.3　自动机理论

自动机理论论述计算的数学模型的定义和性质。这些模型在计算机科学的若干应用领域中起着作用。一个叫做*有穷自动机*的模型在文本处理、编译程序以及硬件设计中有用。另一个叫做*上下文无关文法*的模型在程序设计语言和人工智能中有用。

自动机理论是学习计算理论的极好起点。可计算性理论和复杂性理论需要给计算机下一个精确的定义。自动机理论允许在引入与计算机科学的其他非理论领域有关的概念时使用计算的形式定义。

1.2　数学概念和术语

和任何数学科目一样，开始时先讨论一下预计要用到的基本的数学对象、工具和概念。

1.2.1　集合

集合是一组对象，把它作为一个整体。集合能够包含任何类型的对象，包括数、符号、甚至其他集合。集合中的对象称作它的**元素**或**成员**。集合可以用若干种方式形式地描述。一种方式是在大括号内列出它的元素。例如，集合

$$\{7,\ 21,\ 57\}$$

包含元素7，21和57。符号$\in$和$\notin$分别表示集合成员和非集合成员。如，$7\in\{7,\ 21,\ 51\}$，$8\notin\{7,21,57\}$。对于两个集合A和B，如果A的每一个成员也是B的成员，则称A是B的**子集**，记作$A\subseteq B$。如果A是B的子集且不等于B，则称A是B的**真子集**，记作$A\subsetneq B$。

集合与描述它时元素的排列顺序无关，也不考虑元素的重复。$\{57,\ 7,\ 7,\ 7,\ 21\}$和$\{7,\ 21,\ 57\}$表示同一个集合。如果要考虑元素出现的次数，则把它称作**多重集合**。例如，$\{7\}$和$\{7,\ 7\}$作为多重集合是不相同的，而作为集合是相同的。**无穷集合**包含无穷多个元

素。不可能列出无穷集合的所有元素，所以有时用记号“…”表示“序列永远继续下去”。于是，我们把**自然数集**$\mathcal{N}$写成

$$\{1, 2, 3, \cdots\}$$

把**整数集**$\mathcal{Z}$写成

$$\{\cdots, -2, -1, 0, 1, 2, \cdots\}$$

不含任何元素的集合称作**空集**，记作 $\emptyset$。

当想要描述由服从某种规则的元素组成的集合时，写成 $\{n \mid$关于 n 的规则$\}$。例如，$\{n \mid$对某个 $m \in \mathcal{N}$，$n = m^2\}$ 表示完全平方数集合。

给定两个集合 A 和 B，把 A 和 B 中的所有元素合并成一个集合，这样得到的集合称作 A 和 B 的**并集**，记作 $A \cup B$。既在 A 中又在 B 中的所有元素组成的集合称作 A 和 B 的**交集**，记作 $A \cap B$。所有需要考虑的、不在 A 中的元素组成的集合称作 A 的**补集**，记作 $\overline{A}$。

正象数学中经常做的那样，用形象化的图形帮助阐明概念。对于集合，采用所谓的**文氏图**。把集合表示成圆圈围成的区域。例如，在图 1-1 中，圆表示 t 开头的集合，该集合的几个元素表示成圆内的几个点。

类似地，在图 1-2 中表示以“z”结尾的英语单词集合。

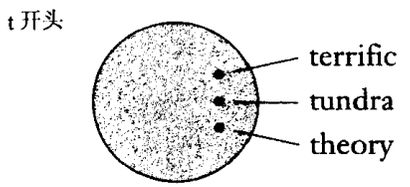


图 1-1 “t”打头的英语单词集合的文氏图

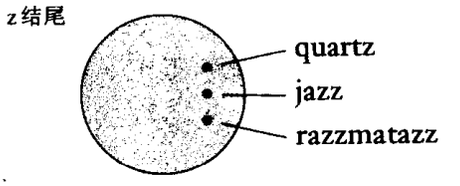


图 1-2 以“z”结尾的英语单词集合的文氏图

在同一个文氏图中表示两个集合，为了表明它们共有某些元素，应该把它们画成部分重叠在一起，如图 1-3 所示。例如，单词 topaz 在两个集合中。图中还有一个表示集合 j 开头的圆，它和表示 t 开头的圆不重叠，因为没有一个单词同时在这两个集合中。

图 1-4 中的两个文氏图描述了集合 A 与 B 的并集和交集。

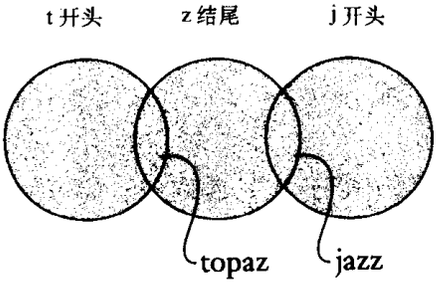


图 1-3 圆部分重叠表明有共同元素

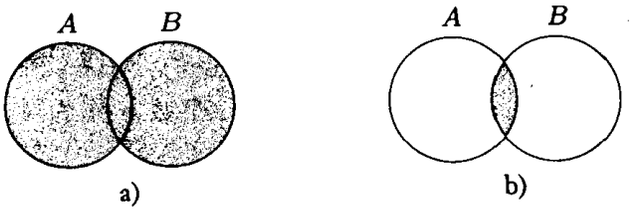


图 1-4 a) $A \cup B$ 和 b) $A \cap B$ 的文氏图

1.2.2 序列和多元组

对象的**序列**是这些对象按某种顺序排列成一列。通常把它写在一对圆括号内指明这是一个序列。例如，序列 7，21，57 写成

$$(7, 21, 57)$$

在集合中不考虑元素的顺序，而在序列中要考虑。因此，(7，21，57) 和 (57，7，21) 是

不相同的。在集合中不允许重复，而在序列中允许重复，从而（7，7，21，57）与前两个都不相同，而集合 {7，21，57} 与 {7，7，21，57} 相同。

与集合一样，序列可以是有穷的或者无穷的。常把有穷序列称作**多元组**。k 个元素的序列称作**k 元组**。例如，（7，21，57）是一个 3 元组。2 元组也叫做**有序对**。

集合与序列可以作为其他集合或序列的元素出现。例如，A 的**幂集**是 A 的所有子集的集合。设 A 是集合 {0，1}，则 A 的幂集是集合 {∅，{0}，{1}，{0，1}}。元素为 0 和 1 的所有有序对的集合是{(0，0)，(0，1)，(1，0)，(1，1)}。

设 A 和 B 是两个集合，A 和 B 的**笛卡儿积**或**叉积**是第一个元素为 A 的成员、第二个元素为 B 的成员的所有有序对组成的集合，记作 $A\times B$。

例 1.1　设 $A=\{1，2\}$ 和 $B=\{x，y，z\}$，则

$$A\times B=\{(1，x)，(1，y)，(1，z)，(2，x)，(2，y)，(2，z)\}$$

还可以有 k 个集合 A_1，A_2，…、A_k 的笛卡儿积，记作 $A_1\times A_2\times\cdots\times A_k$。它是由所有 k 元组（a_1，a_2，…，a_k）组成的集合，其中 $a_i\in A_i$。

例 1.2　设 A 和 B 与例 1.1 中的相同，则

$$\begin{aligned}A\times B\times A=\{&(1，x，1)，(1，x，2)，(1，y，1)，(1，y，2)，\\&(1，z，1)，(1，z，2)，(2，x，1)，(2，x，2)，\\&(2，y，1)，(2，y，2)，(2，z，1)，(2，z，2)\}\end{aligned}$$

集合自身的笛卡儿积采用下述缩写

$$\overbrace{A\times A\times\cdots\times A}^{k}=A^k$$

例 1.3　集合 $\mathcal{N}^2$ 等于 $\mathcal{N}\times\mathcal{N}$。它由所有自然数的有序对组成，也可以写成 $\{(i，j)\mid i，j\geqslant 1\}$。

1.2.3　函数和关系

函数是数学的中心。**函数**建立一个输入－输出的关系。它取得一个输入和产生一个输出。对于每一个函数，同样的输入总是产生同样的输出。设 f 是一个函数，当输入值为 a 时它的输出值为 b，则记作

$$f(a)=b$$

函数又称作**映射**，并且若 $f(a)=b$，则说 f 把 a 映射到 b。

例如，绝对值函数 *abs* 取一个数 x 作为输入，当 x 大于等于 0 时返回 x；当 x 小于 0 时返回 $-x$。因此，$abs(2)=abs(-2)=2$。加法是函数的另一个例子，记作 *add*。加法函数的输入是一对数，输出是这两个数的和。

函数的所有可能的输入组成的集合称作它的**定义域**。函数的输出取自于一个集合，这个集合称作它的**值域**。记号

$$f：D\to R$$

表示 f 是定义域为 D、值域为 R 的函数。如果限制在整数范围内，函数 *abs* 的定义域和值域都是 $\mathcal{Z}$，因而写成 $abs：\mathcal{Z}\to\mathcal{Z}$。对于整数的加法函数，定义域是整数有序对集合 $\mathcal{Z}\times\mathcal{Z}$、值域是 $\mathcal{Z}$，因而写成 $add：\mathcal{Z}\times\mathcal{Z}\to\mathcal{Z}$。注意：函数不必取到指定的值域的所有元素。虽然 $-1\in\mathcal{Z}$，但是函数 *abs* 决不会取到值 -1。如果函数取到值域的所有元素，则称它**映上**到这个值域。

可以用若干种方法描述一个具体的函数。一种方法是采用从指定的输入到输出的计算过程。另一种方法是用一张表列出所有可能的输入和对应的输出。

例 1.4 考虑函数 $f:\{0, 1, 2, 3, 4\} \to \{0, 1, 2, 3, 4\}$。

n	0	1	2	3	4
$f(n)$	1	2	3	4	0

该函数把它的输入加 1，然后输出模 5 的结果。一个数的模 m 等于该数除以 m 后的余数。例如，钟表表盘上的分针按模 60 计数。当做模算术时，记 $Z_m = \{0, 1, 2, \cdots, m-1\}$。用这个记号，前面提到的函数 f 可写成 $f: Z_5 \to Z_5$。 ▨

例 1.5 如果函数的定义域是两个集合的笛卡儿积，这时要使用 2 维表格。这里有另一个函数 $g: Z_4 \times Z_4 \to Z_4$。表中标号为 i 的行和标号为 j 的列的内容是 $g(i,j)$ 的值。

g	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

g 是模 4 的加法函数。 ▨

当函数 f 的定义域为 $A_1 \times A_2 \times \cdots \times A_k$ 时，f 的输入是 k 元组 $(a_1, a_2, \cdots, a_k)$，称 $a_1, a_2, \cdots, a_k$ 是 f 的自变量。k 个自变量的函数称作 **k 元函数**，k 称作函数的**元数**。当 k 等于 1 时，f 有一个自变量，称作**一元函数**。当 k 等于 2 时，f 是**二元函数**。某些大家熟悉的二元函数写成特殊的**中缀表示法**的形式，把函数符号放在它的两个自变量之间，而不是采用**前缀表示法**把函数符号放在最前面。例如，加法函数 *add* 通常采用中缀表示法把符号 + 放在两个自变量之间，写成 $a + b$，代替采用前缀表示法的 $add(a,b)$。

谓词或**性质**是值域为{TRUE，FALSE}的函数。例如，设 *even* 是一个谓词，当输入是偶数时为 TRUE；当输入是奇数时为 FALSE。例如，*even* (4) = TRUE，*even* (5) = FALSE。

定义域为 A^k 的谓词称作**关系**、**k 元关系**或**A 上的 k 元关系**。通常用的是**二元关系**。当写包含一个二元关系的表达式时，习惯上采用中缀表示法。例如，“小于”是一个关系，通常写成带中缀符号 < 的形式。“等于”是另一个大家熟悉的关系，写成带中缀符号 = 的形式。设 R 是一个二元关系，命题 aRb 表示 aRb = TRUE。类似地，设 R 是一个 k 元关系，命题 $R(a_1, \cdots, a_k)$ 表示 $R(a_1, \cdots, a_k)$ = TRUE。

例 1.6 在一个叫做剪刀 - 纸 - 石头的儿童游戏中，两个人同时选择集合{剪刀，纸，石头}中的一个成员，并且用手势表示出来。如果两个人的选择是相同的，则游戏重来。如果选择不相同，则按照下述打败关系，有一个人获胜。

打败	剪刀	纸	石头
剪刀	FALSE	TRUE	FALSE
纸	FALSE	FALSE	TRUE
石头	TRUE	FALSE	FALSE

根据这张表，剪刀打败纸为 TRUE，而纸打败剪刀为 FALSE。

有时用集合代替函数来描述谓词更方便些。谓词$P:D\to\{\text{TRUE},\text{FALSE}\}$可以写成 (D, S)，其中 $S=\{a\in D\mid P(a)=\text{TRUE}\}$，或当根据上下文定义域 D 是明显的时，简单地写成 S。于是，打败关系可以写成

$$\{(\text{剪刀},\ \text{纸}),\ (\text{纸},\ \text{石头}),\ (\text{石头},\ \text{剪刀})\}$$

等价关系是一种特殊类型的二元关系，它抓住了用某种特征把两个对象等同起来的想法。如果二元关系 R 满足下述 3 个条件：

1）R 是**自反的**，即对每一个 x，xRx。

2）R 是**对称的**，即对每一个 x 和 y，xRy 当且仅当 yRx。

3）R 是**传递的**，即对每一个 x，y 和 z，xRy 和 yRz 蕴涵 xRz。

则 R 是一个等价关系。

例 1.7　定义一个自然数集上的等价关系$\equiv_7$。对于 i，$j\in\mathcal{N}$，如果 $i-j$ 是 7 的倍数，则说 $i\equiv_7 j$。这是一个等价关系，因为它满足上面的 3 个条件。首先，它是自反的，因为 $i-i=0$ 是 7 的倍数。其次，它是对称的，因为若 $j-i$ 是 7 的倍数，则 $i-j$ 也是 7 的倍数。第三，它是传递的，因为只要 $i-j$ 是 7 的倍数且 $j-k$ 是 7 的倍数，那么 $i-k=(i-j)+(j-k)$ 是两个 7 的倍数之和，从而也是 7 的倍数。 ▨

1.2.4　图

无向图又简称作**图**，是一个点的集合及连接其中某些点的线段。这些点称作**顶点**，线段称作**边**，如图 1-5 所示。

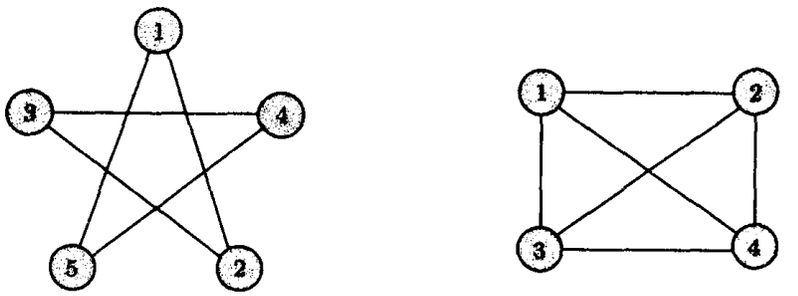


图 1-5　图的例子

顶点的**度数**是以这个顶点为端点的边的数目。在图 1-5a 中所有顶点的度数为 2。在图 1-5b 中所有顶点的度数为 3。任何两个顶点之间至多有一条边。

设图 G 包含顶点 i 和 j，有序对 (i, j) 表示连接 i 和 j 的边。在无向图中不考虑 i 和 j 的顺序，所以有序对 (i, j) 和 (j, i) 表示同一条边。由于顶点的顺序是无关紧要的，所以有时用集合，而不用有序对表示边，写成 $\{i, j\}$。如果 G 的顶点集为 V、边集为 E，则记作 $G=(V, E)$。可以用一个图形或更形式地用指定 V 和 E 来描述一个图。例如，图 1-5a 中图的形式描述为

$$(\{1, 2, 3, 4, 5\},\ \{(1, 2), (2, 3), (3, 4), (4, 5), (5, 1)\})$$

b 中图的形式描述为

$$(\{1, 2, 3, 4\},\ \{(1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4)\})$$

经常用图表示数据。顶点是城市、边是连接城市的高速公路，或者顶点是电子元件、边是连接它们的导线。有时为了方便，给图的顶点或边做标记，这样的图称作**标定图**。图 1-6 画出一张图，它的顶点是城市。如果两个城市之间有直达航班，则它们之间有一条边，并且标上直达飞行的最低票价（美元）。

如果图 G 的顶点集是图 H 的顶点集的子集，则称 G 是 H 的**子图**。如图 1-7 所示，G 的边均是 H 在对应顶点上的边。

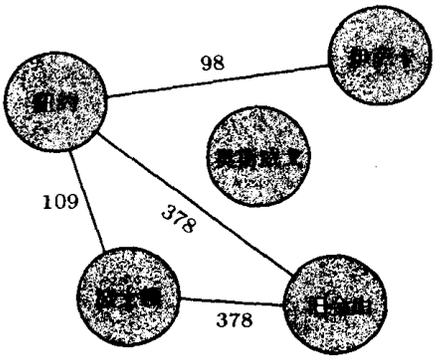


图 1-6 各城市之间直达飞行的最低票价

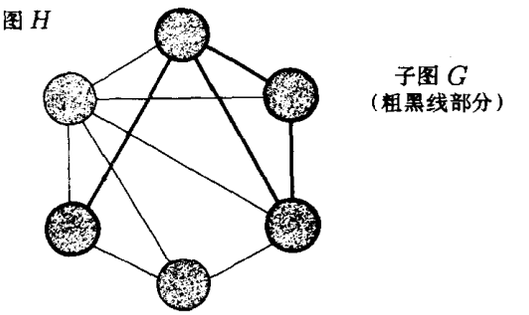
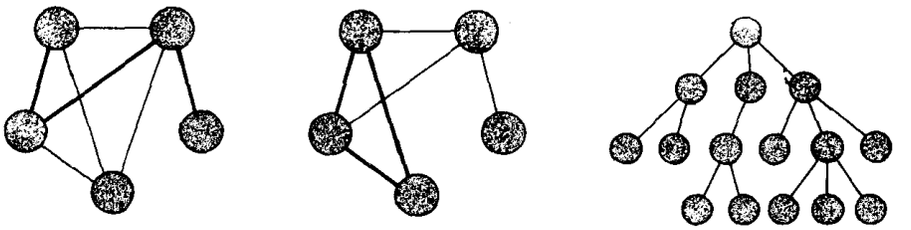


图 1-7 图 G（粗黑线部分）是 H 的子图

图中一条**路径**是由边连接的顶点序列。**简单路径**是没有顶点重复的路径。如果每一对顶点之间都有一条路径，则称这个图是**连通的**。如果一条路径的起点和终点相同，则称它是一个**圈**。如果一个圈除第一个和最后一个顶点之外没有顶点重复，则称它是一个**简单圈**。连通且没有简单圈的图是**树**，如图 1-8 所示。一棵树中度数为 1 的顶点叫做这棵树的**树叶**。有时专门指定树的一个顶点，把它叫做**根**。



a) b) c)

图 1-8 a）图中的一条路径，b）图中的一个圈，c）一棵树

如果在图中用箭头代替线段，则得到**有向图**，如图 1-9 所示。从一个顶点引出的箭头数是这个顶点的**出度**，指向一个顶点的箭头数是这个顶点的**入度**。

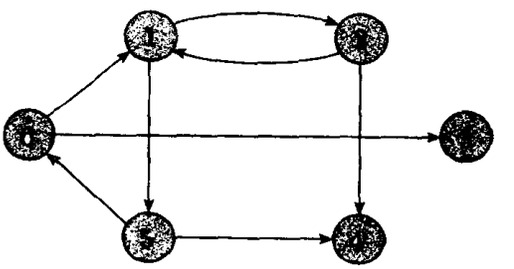


图 1-9 一个有向图

在有向图中，用有序对 (i, j) 表示从 i 到 j 的边。一个有向图 G 的形式描述是 (V, E)，其中 V 是顶点集，E 是边集。图 1-9 中有向图的形式描述是

$$(\{1, 2, 3, 4, 5, 6\}, \{(1, 2), (1, 5), (2, 1), (2, 4), (5, 4), (5, 6), (6, 1), (6, 3)\})$$

所有箭头的方向都与其前进的方向一致的路径叫做**有向路径**。如果从每一个顶点到另