高等学校试用教材

算法设计与分析

北京理工大学　周培德　编著

机械工业出版社

高等学校试用教材

算法设计与分析

北京理工大学　周培德　编著

机械工业出版社

(京)新登字054号

本书系统地介绍了算法的概念、设计方法及复杂度分析。全书分13章，包括：绪论、算法设计步骤及算法分析的概念、基础数学、算法设计的方法、分类、数据集合上的操作、图和网络的算法、几何问题和代数问题的算法、串匹配算法、NP完全性理论及近似算法、下界理论、概率算法和算法的概率分析简介、并行算法等，内容丰富、由浅入深，可读性较好。

本书是高等院校计算机、自动控制、工程管理等专业本科生、研究生的教材，也可作为相应专业科技人员的参考书。

算法设计与分析

北京理工大学　周培德　编著

责任编辑：卢若薇　　责任校对：刘　茹

封面设计：郭景云　　版式设计：胡金琪

责任印制：　王国光

机械工业出版社出版(北京阜成门外百万庄南街一号)

(北京市书刊出版业营业许可证出字第117号)

机械工业出版社京丰印刷厂印刷

新华书店北京发行所发行·新华书店经售

开本 787×1092 1/16·印张21 1/4　·字数518 千字

1992年5月北京第1版·1992年5月北京第1次印刷

印数　0,001—1,700·定价：　5.90元

ISBN 7-111-03085-0/TP·152(课)

前　言

本书是经全国高等工业学校机电类、兵工类专业教学指导委员会计算机及应用专业教学指导小组招标审定的教材。在编写过程中，参考了IEEE-83教程推荐大纲及笔者自编的讲义。为了适应计算机专业教学发展的需要，对大纲的要求作了某些补充。

算法的设计与分析是计算机科学的核心问题之一，是计算机系各专业（计算机系统结构、系统软件和应用等专业）学生及研究生的一门重要的专业基础课。其内容是研究计算机领域及其它有关领域中的一些常用算法。通过学习，读者可以掌握算法设计的常用方法，以便运用这些方法去独立设计解决各类实际问题的算法；利用已有的算法去解决计算机科学与工程领域中较为复杂的实际问题；对已有的算法进行改进，使之能更快、更有效地解决问题。此外，还要学会分析算法，估计算法的时空复杂性，因为只有正确地分析了一个算法，才能真正理解这一算法并作出科学的评价。学习和掌握这些方面的知识，不仅对计算机专业的技术人员，而且对要使用计算机的其它各专业技术人员也是必不可少的。

按处理问题的方式分类，算法可以分为串行算法和并行算法。这两类算法在处理问题的方式、速度和时空复杂性及其度量标准等方面都有着相当大的差别。当然，可以把串行算法看成是特殊的并行算法。本书主要介绍串行算法。

本书共分十三章。前六章可作为大学生选修课的教材，后七章可供研究生学习使用。当然，也可由教师根据实际情况选用。

第一、二章介绍算法的定义及基本概念，计算模型和复杂度的度量。这是本书的理论基础。

第三章讲述本书涉及的主要数学知识，为分析算法的复杂性作准备。对这一部分内容，有较好数学基础的读者只要翻阅一下就可以了。

第四章系统地讲述设计算法的常用方法。该章一共介绍了十种方法，内容十分丰富。学习这一章时，要掌握这些方法的基本思想，最好能编制出相应的程序上机调试，并分析其复杂性，以加深理解。

第五章讲述分类。分类方法在《数据结构》中大部分已讲过。这里的重点是分析分类方法的复杂性，所以与《数据结构》的内容重复不多。

第六章介绍数据集合上的各种操作，目的是寻找一种合适的数据结构，能对给定的问题较快地执行某些操作。同时给出了时间复杂性的分析。

第七章系统地介绍图论领域中的一些问题的最佳算法。这些算法有广泛的应用。

第八章讲述几何问题与代数问题的算法。其中几何问题的算法虽然只有短暂的历史，但对它的作用却不能低估，这是值得重视的一个新的领域。

第九章介绍串匹配的算法。列举的五种算法是比较巧妙的。

第十章介绍NP完全问题。这个问题对于认识算法的本质和各种问题的难解度具有十分重要的理论意义，是计算机科学工作者及应用计算机和数学方法解决实际问题的科技工作者的必备知识之一。NP完全问题本身还有许多问题需要进一步研究。

第十一章介绍下界理论。下界理论为证实某个算法是最有效的算法提供理论依据，当然是读者最关心的问题之一。本章还介绍了求解某类问题的下界的一些方法。

第十二章介绍概率算法和算法的概率分析。概率算法是70年代中期发展起来的，它把随机性注入算法本身，使某些问题得到了较好的解决。

第十三章介绍并行计算的某些概念、形式模型、技术及算法。

本书根据不同情况采用拟ALGOL-PASCAL语言、框图或自然语言来描述算法。为把这些描述形式变成上机程序，读者要下一番功夫，这样做，与本书的目标并不矛盾。

本书由合肥工业大学张奠成教授主审，北京理工大学吴鹤龄教授仔细阅读了第一至第四章底稿，提出了许多宝贵意见，在此，对他们的帮助表示衷心的感谢。

由于笔者水平有限，加上时间仓促，书中的缺点和错误在所难免，恳请同行与读者批评指正。

编著者

1991年4月

目　录

第一章 绪 言

众所周知，计算机与数值问题的求解（比如方程式求根、插值计算、数值积分和函数逼近等）有关，但本书一般不讨论这些问题。近30多年来，发展了利用计算机来解决非数值问题，比如分类、翻译语言、图形识别、解高等代数和组合分析等方面的数学问题以及定理证明、公式推导乃至对生产和日常生活中的各种过程的模拟等。在这样一些问题中，主要是进行判断、比较，而不是进行算术运算。算法设计与分析的主要研究对象是非数值问题，当然也不排斥对某些数值问题的研究。

一、什么是算法

下面举一个例子，使读者对什么是算法有个初步的了解。

例1 给定两个正整数m和n，求它们的最大公因子，即能够同时整除m和n的最大整数。

算法1-1 欧几里得算法

输入 正整数m、n。

输出 m和n的最大公因子。

第1步 求余数。m除以n，并令r为所得的余数（$0 \leqslant r < n$）。

第2步 判余数r是否为0。若$r=0$，则算法终止（n为答案）；否则转第3步。

第3步 互换。置$m \leftarrow n$，$n \leftarrow r$，返回第1步。

这个算法的框图如图1-1所示。

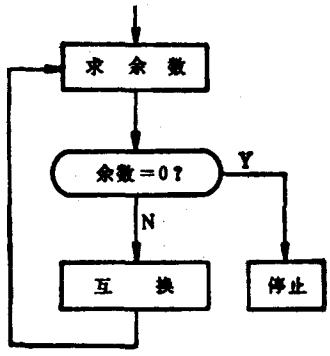

图1-1 算法1-1的框图

下面说明算法的常用符号：

（1）箭头“←”是代替运算（或叫赋值），与“=”运算不同。$m \leftarrow n$意思是用变量n的当前值代替变量m的值。“=”号标记一个可被测试的条件，而“←”号表示一个可被执行的动作。“$n \leftarrow n+1$”表示n增加1的运算。

（2）在算法1-1第3步中，两个代替运算$m \leftarrow n$、$n \leftarrow r$的次序不能颠倒。$n \leftarrow r$、$m \leftarrow n$，表示用r的值代替n的值，用n的值代替m的值，实际上是用r的值代替n、m的值。

（3）两个变量的值相互交换，可以写成“交换$m \rightleftharpoons n$”，或通过一个新的变量来实现，写成“置$t \leftarrow m$，$m \leftarrow n$，$n \leftarrow t$”。这里的操作顺序也不能随意改变。

实现算法的通常方法是先赋给变量一定的值，然后沿着框图所示的步骤顺序执行，直到执行完毕为止。仍以上面的算法为例。

设$m=119$，$n=544$。从第1步开始，119除以544商是0，余数是119，余数不为0，转第3步，置$m \leftarrow 544$，$n \leftarrow 119$（交换$m \rightleftharpoons n$），转第1步。544除以119，商是4，余数68，$r \leftarrow 68$（$r \neq 0$），转第3步。置$m \leftarrow 119$，$n \leftarrow 68$。依次类推。最后，51除以17时，$r \leftarrow 0$，于是算法在第2步终止。119和544的最大公因子是17。

二、算法的改进

从上面的过程可以看出，当m＜n时，第1步中的商总为0，实际上没有做有效的除法。第一次循环，只是把m、n交换，尔后开始第二次循环，然后才能进行有效操作。这样就加长了实现算法所需要的时间。如果在算法中增加第0步：如果m＜n，则交换m⇌n。这样改动后，算法的长度增加了（由3步变成4步），但对于输入集合中的元素对，大约有一半的情况，减少了实现算法所需要的时间。这就用空间换取了时间。改进后的算法如下：

算法1-2 改进的欧几里得算法

输入、输出与算法1-1相同。

第0步 保证m≥n。如果m＜n，交换m⇌n；否则转第1步。

第1步 求余数。m除以n，令r为余数（0≤r＜n）。

第2步 判余数是否为0。若r＝0，算法终止，n为答案；否则转第3步。

第3步 互换。置m←n，n←r，转第1步。

如果再用上述数值代入运行，那么循环的次数比以前减少，实现算法所需要的时间也减少，因此算法1-1得到了改善。

注意，这里假设循环一次耗费一个单位时间，所以循环次数越多，耗时也越多。

三、算法的第一个定义

定义1-1 算法是一个有穷规则的集合。

其中“规则”规定了一个解决某一特定问题类的运算序列。下一章还要给出算法的另外两个定义。值得指出的是，算法不等于程序，算法设计不等于程序设计。

四、算法的五个特征

进行算法设计和分析，一定要注意算法的以下五个特征。

1．有穷性 一个算法必须在执行有穷步之后结束。算法1-1就满足这个特性，因为它的执行导致一个递减的正整数序列，必然要终止。

2．确定性 算法的每个步骤必须有确定的意义。对于每种情况，有待执行的动作必须严格地规定，不能出现二义性。

如果算法是用自然语言描述的，那就有可能产生不同的理解。为了克服这一困难，一般采用拟ALGOL语言等来描述算法。

对于算法1-1而言，确定性意味着：m和n都是正整数，在正整数范围内，除法所得的余数仍然是正整数，不会是小数和无理数。这是设计算法1-1时的约定。如果m和n不是正整数，那么除和余数就没有一般的约定。如π除以5的余数是什么？0除$\frac{3}{2}$的余数是什么？在这种情况下，算法1-1就是不确定的。

3．输入 一个算法可以有0个或多个输入。所谓算法的输入就是指在算法开始执行之前，对算法中的输入变量所给的初始值。算法的输入取自特定的对象集合。如算法1-1，有两个输入，即m和n，它们取自正整数集合。

4．输出 一个算法可以有一个或多个输出。算法的输出是同输入有特定关系的量。如算法1-1有1个输出，即第2步中的n，它是两个输入量的最大公因子。

5．能行性（可行性） 能行性指的是算法中的所有运算都是基本的，原则上它们都能够精确地进行，而且进行有穷次即可完成。

例如算法1-1中用了除法测试一个整数是否为0、置一个变量的值等于另一个变量的值这样一些运算。这些运算都是可行的，因为整数可以用有穷的方式表示，而且至少有一个方法（即除法）可用来完成一个整数除以另一个整数的运算。但若两数是由无穷的十进展开式所确定的任意实数，则除法运算就不是可行的。

五、评价算法的标准

一个算法应该在有穷步内终止，这是对算法的最起码的要求。而作为一个有效的算法，则不仅要求有穷的步骤，而且要求是很有限的步骤。也就是说，我们不仅需要算法，而且更需要好的算法。什么是好的算法？这并没有严格的定义，但通常指两个方面：执行算法所需要的时间短；执行算法所需要的计算机内存容量（即空间）小，即指算法的时间复杂性和空间复杂性。这两方面的内容以后还要讨论，这是本书的重点。此外如算法的正确性、简单性、最佳性和精确性等，也是评价算法的标准，在下面的章节中将分别给予介绍。

六、算法分析

对于同一个问题可能有若干个不同的算法，这就产生了要判断哪一个算法是最好的问题。这是算法分析领域的问题，即确定算法的性能特征。

在进行算法分析时，要考虑最坏情况下的复杂性和平均特性。对于算法1-1，假定n的值已知，而m可以取任意正整数，那么，执行算法1-1的第1步（即除法）的平均次数T_n是多少呢？这个问题可以归结为，对m＝1，2，…，n逐一试验算法，计算出执行第1步的总次数，然后除以n。可以证明，对于充分大的n值，T_n近似等于$\left(\frac{12\ln 2}{\pi^2}\right)\ln n$，即同n的自然对数成比例。另外，还可以证明，算法1-1所需除法的次数不超过$1.441\log_2 n+1$，即算法1-1的最坏情况下的复杂性为$1.441\log_2 n+1$。

算法论是另外的一个课题，它主要讨论算法的存在性或不存在性。

七、算法的形式化表示

可以借助于数学的集合论来建立算法的形式化描述。

定义算法是一个四元组（Q、I、Ω、f），其中Q是一个集合，I和Ω是Q的子集，f是由Q到它自身的一个映射，这个映射保留Ω的每个元素不动，即对于$q\in\Omega$，有

$$f(q)=q$$

四元组中的四个元素Q、I、Ω、f分别用来表示计算的状态、输入、输出和计算的规则。对于集合I中的每个输入x，通过f可以定义一个计算序列y_0、y_1、y_2、…如下

$$\begin{cases} y_0=x \\ y_{k+1}=f(y_k) \qquad k\geqslant 0 \end{cases}$$

若此计算序列在第k步终止，且k是使y_k在Ω中的最小整数，则称输出y_k是由x产生的：

$$x=y_0\xrightarrow{f}y_1\xrightarrow{f}y_2\xrightarrow{f}\cdots\xrightarrow{f}y_k\xrightarrow{f}y_{k+1}$$

如果y_k在Ω中，则y_{k+1}也在Ω中，此时$y_{k+1}=y_k$。某些计算序列可能永不终止。一个算法是：对于I中的所有x，都在有穷多步内终止的一个计算方法。

例2 算法1-1的形式化。

设Q是所有单个正整数（n），所有有序偶（m、n）和所有有序的四元组（m、n、r、1）、（m、n、r、2）、（m、n、p、3）的集合，其中m、n、p是正整数，r是

非负整数。设 I 是所有有序偶（m、n）组成的子集，Ω是所有单个的数（n）组成的子集。设 f 定义如下：

$$f(m、n)=(m、n、0、1)$$

$$f(n)=(n)$$

$$f(m、n、r、1)=(m、n、m除以n的余数，2)$$

$$f(m、n、r、2)=\begin{cases}(n)，如果\ r=0\\(m、n、r、3)，否则\end{cases}$$

$$f(m、n、p、3)=(n、p、p、1)$$

这些符号和表示式与算法1-1是对应的，它是算法1-1的形式化表示。

本书将介绍算法的基本概念，算法设计与分析的基本方法、分类方法及复杂性，数据集合上的操作等基本内容。另外，还将介绍图的算法，几何代数算法，串匹配算法和NP完全性理论、下界理论、概率算法及并行算法。希望读者通过对本书的学习，能掌握算法设计的基本方法及分析算法的方法，以便能针对具体问题设计出有效的好算法。

习　题

1-1　利用算法1-1和1-2，求1986和5976的最大公因子。

1-2　分析算法1-2在什么情况下比算法1-1好。

1-3　改进算法1-2，使循环次数有所减少。

1-4　在算法的形式化(四元组)表示中，算法的有穷性如何处理？

第二章　算法设计的步骤及算法分析的基本概念

§2-1　算法的定义

作为一个解决现代工程问题的设计者，都希望利用计算机来解决一些自己所关心的实际问题，这些问题包括科学计算、过程控制和信息处理等。为了解决这些问题，就需要找出解决问题的算法。因此，算法是解决问题过程中的一个重要环节。

现代工程中提出的问题越来越复杂，现代的计算机系统也越来越复杂。因此，利用计算机来解决一个新问题，或对前人已研究过的问题重新进行深入研究，建立新的算法，必须要有丰富的算法设计经验，要有较强的抽象思维能力，要善于发现已有算法和程序的弱点并加以改进。

在第一章中，已给出算法的一个定义，这里再给出算法的另一定义。

定义2-1　算法是求解一个问题类的无二义性的有穷过程。

该定义中的过程是求解问题而执行的一步一步动作，每一步动作只需要有限存储单元和有限的操作时间。对符合条件的任何输入，任一算法必须在可以接受的有限时间内终止。

这个定义的缺陷是“无二义性”这个术语有些模糊，它本身就隐含着“二义性”。所谓二义性就是对同一事物可以这样理解也可以那样理解。这是由于对某一概念给以不同的定义或给出一个模糊的定义所造成的。此外，二义性也与有关人员掌握的知识量不同有关。

有些问题明显地存在着算法，但要按给定的格式来描述它却可能很困难。比如系鞋带问题，明显地存在有效算法，连只有几岁的小朋友都能自己解决这个问题，即能自己系鞋带。但是要给出系鞋带问题一个纯文字的、不加图形解释的算法却是很困难的。

上述定义的优点是灵活、直观。为了保持这些优点，克服定义中某些二义性的成分，需要详细说明一个典型的现代计算机以及与这种计算机通信的语言，这时凡用这种语言编写的、可以在这种给定的计算机上执行的过程就叫做算法。

现代计算机中最基本的模型是图灵（Turing）机，可以理解算法是对所有有效输入都停机的图灵机。对图灵机的描述属自动机理论，内容将在第十章中给以介绍。

§2-2　算法设计的步骤

本节简单介绍算法设计的完整过程，弄清各步的要求及各步间的关系，反复修改模型和算法，以建立一个较好的算法，求得问题的解。

一、问题的陈述

准确地陈述问题是解决问题的第一步。为了对问题作出准确的陈述，必须注意以下要点：

在未经加工的原始表达中所用的行话大家都懂吗？有二义性吗？

用户给出了什么信息？它们有什么用？

用户希望得到什么结果？

是否遗漏了什么信息？这些信息有什么用处？

怎样判定求得的解正是所需要的？

陈述中作出了什么假设（比如与算法的应用范围有关的约定）？等等。

总之，必须认真审查表达问题的有关陈述，步步深入地追问，以加深对问题的分析。

例1　售货员路线问题（即货郎担问题）。

设售货员在一天内要到五个城市去推销货物，已知从一个城市到其它城市的费用，求总费用最少的路线。

给出的信息主要有五个城市的关系图及相应的费用矩阵（或称耗费矩阵、花费矩阵）。费用矩阵的元素c_{ij}表示从城市 i 到城市 j 的费用（或称耗费、花费）。此时的费用矩阵是一个五阶的正方阵。

希望得到的结果是尽量减少来往于五城市之间的总费用。为此还要弄清楚一些附加问题。如能不能在一天内走遍五个城市？货物推销不出去或供不应求怎么办？五个城市中除作为基地的城市外，其它 4 个城市中有没有必须优先访问的城市？一天之内能否两次经过某城市？

对于这些问题，必须向用户了解清楚。比如用户可能说，一天访问五个城市在时间上是不成问题的；货物推销不出去也不停留，一天还是走遍五个城市；货物供不应求也要走一遍，不能中途返回；五个城市中没有一个城市有优先权；一天之内不能两次通过某城市。把这些弄清楚之后，问题也就清楚了：这就是要得到一张城市表，除基地城市是表的起点和终点外，其它每个城市只能在表中出现一次。表中城市的次序，代表售货员到达城市的次序。表中相邻城市之间标上费用，所有费用之和就是总费用。如果能给用户一张总费用为最小的表，那么用户提出的问题就解决了。

二、模型的拟制

这里所说的模型是数学模型。模型是否恰当，对解决问题的其它阶段影响很大。用计算机解决实际问题时必须有恰当的数学模型，否则，在庞大而复杂的问题面前，计算机将无能为力。对一个问题拟制模型固然要靠经验，但还是有一些原则可以遵循的。在建模阶段至少要考虑以下两个基本问题，即最适合于这个问题的数学结构是什么？有没有已经解决了的类似问题可供借鉴？

对上述第二个问题如果有肯定的答复，那么通过对类似问题的比较和联想，可以加快问题的解决。但毕竟第一个问题是更重要的。如何选择适当的数学工具来表达已知的和要求的量，受下列因素的影响：设计人员掌握的数学知识；数学结构是否表达方便；计算是否简单；与这种结构有关的或对这种结构本身所要进行的操作种类的多少与功能的强弱。

对同一问题可以用不同的数学工具建立不同的模型。因此对各种模型要进行比较，选择最有效的模型。模型初步选定后，应该依据所选定的模型对问题重新陈述，并考虑下列问题：

模型是否清楚地表达了与问题有关的所有重要的信息？

模型中是否存在与要求的结果相关的数学量？

模型是否正确反映了输入、输出的关系？

对这个模型处理起来困难吗？

如果能回答这些问题，那么这个模型就可以作为一个候选模型。

仍以售货员路线问题为例，我们先考虑有没有可供借鉴的类似问题。不难想到道路交通

图是可以借鉴的。首先把五个城市画成五个小圆圈或点，标上编号1、2、3、4、5。其次在任意两城市i、j之间的联线上写上从i到j（或从j到i）的费用，这个费用称为"权"。这样建立起来的对象在数学上叫做网络或图，它就是解决售货员路线问题的模型。图2-1表达了这个模型。同这个模型密切相关的费用矩阵为：

$$C=\begin{pmatrix} \infty & 1 & 2 & 7 & 5 \\ 1 & \infty & 4 & 4 & 3 \\ 2 & 4 & \infty & 1 & 2 \\ 7 & 4 & 1 & \infty & 3 \\ 5 & 3 & 2 & 3 & \infty \end{pmatrix}$$

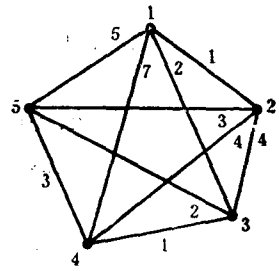

图2-1 五城市售货员路线问题

路线的总费用是经过各边时的费用之和。如能找到最小费用的路线，这个问题就解决了。

图中可以找到这样一条路线，即

$$1 \xrightarrow{1} 2 \xrightarrow{3} 5 \xrightarrow{3} 4 \xrightarrow{1} 3 \xrightarrow{2} 1$$

总费用为1＋3＋3＋1＋2＝10。

三、算法的详细设计

数学模型建立之后，就可以进行算法的详细设计了。算法的选择与模型的选择是密切相关的；但是对同一模型仍然可以有不同的算法，这些算法的有效性可能相差很大。

算法的详细设计是指设计求解某个具体问题类的一系列步骤，并且这些步骤可以通过计算机的各种操作来实现。

仍以售货员路线问题为例。根据问题的陈述和模型的特征，可以采用下述算法。

对任意数目的n个城市，分别用1到n的数字编号，基地城市用n表示。不难发现，每一条旅行路线和整数编号1，2，…，n－1，n的一个排列之间存在着一一对应的关系。因此，对一种给定的排列，可以求出与此对应的旅行路线，并计算出这条路线的费用。对所有的排列（共(n－1)!种）逐一计算，并把到目前为止的最小费用路线存起来。如果找到一条费用更小的路线，就把这条路线作为下次比较的标准。因为n是有穷的，所以此算法是可以终止的。

算法2-1 求n个城市的售货员路线问题的解

输入 城市数目n；费用矩阵$C=(c_{ij})_{n\times n}$。

输出 旅行路线TOUR；最小费用MIN。

方法 费用矩阵C中行列次序和n个城市的编号相一致，基地城市编号为n，前n－1个城市编号的一个排列P对应一条路线T(P)，同时计算其费用COST(T(P))；若当前路线的费用小于当前的最小费用MIN，则MIN←COST(T(P))，否则继续选下一条路线。最后打印输出MIN及对应的TOUR。

```
begin
    I ← 1；TOUR ← 0；MIN ← ∞/初始化/
  while  I ≤ (n－1)!   do
      begin
```

```
        P←PHRMUTI(n－1，I)/PHRMUTI(n－1，I)是生成1到n－1第I个排
                                列的子算法/
        COST(T(P))←EFP(C，T)/EFP(C，T)是由费用矩阵C及路线T(P)所算得的总
                                费用子算法/
        if COST(T(P))<MIN then
            begin
                TOUR←T(P);
                MIN←COST(T(P))
            end
            I←I＋1
        end
        PRINT MIN; TOUR
end
```

其中有两个子算法还需要加工。这是一个精确算法，可以求得最佳解。

四、算法的正确性

对于算法正确性的验证，可以通过验证表达该算法的程序的正确性来实现，这样就把问题转化了。但是这种转化并不能令人完全放心，因为程序正确与算法正确两者不完全一致。由于证明算法的正确性目前仍很困难，所以还是采用验证程序的正确性来说明相应的算法是正确的。

给程序以各种有代表性的典型输入，通过计算获得结果，如果与事先知道的正确结果一致，则说明此程序是可用的，但还不能说已证明了该程序的正确性。

程序正确性的证明类似数学中的定理证明。对于一个具体问题的有关要求可以看作"做什么"的说明，而一个程序就相当于"如何做"的说明。显然，"做什么"是一种"静态"描述，它只描述程序要完成的任务，并没有表明如何去做；而"如何做"是一种"动态"描述，它明确指出解决该问题的具体步骤。这样，程序正确性的证明就是要论证"如何做"才能实现所给定的"做什么"。"做什么"的说明一般包括"输入说明"("输入断言"或"输入谓词")和"输出说明"("输出断言"或"输出谓词")。"输入谓词" $\phi(x)$ 指出输入变元 x 的定义域应满足什么条件，才能使程序的执行有意义。"输出谓词" $\psi(x, z)$ 指出输出变元 z 和输入变元 x 之间应满足的关系。

我们的任务是要证明程序对于满足输入谓词的所有输入，必须保证程序的执行可以终止，而且执行完毕时满足输出谓词。

证明程序正确性时程序的终止性是必须考虑的一个问题，即是考虑程序的循环部分，循环次数一定是有限次而不能是无限次。程序正确性一般分为部分正确性、终止性和完全正确性三种，其中完全正确性是程序正确性证明的最终目标。一般的做法是分别论证程序的部分正确性（这时假设终止性成立）和程序的终止性，也可直接论证其完全正确性。

设程序 P 的变量为：输入变量 $\overline{x}$，程序变量 $\overline{y}$（计算中的工作单元），输出变量 $\overline{z}$。另设 P 由四种语句组成：start语句，assignment语句〔$\overline{y}\leftarrow g(\overline{x}, \overline{y})$，以当前值 $\overline{x}$、$\overline{y}$ 计算 $g(\overline{x}, \overline{y})$ 的值来代替 $\overline{y}$ 的值〕，test 语句，halt 语句。程序 P 由 start 语句开始，在 assignment语句和test语句上循环有限次。如果执行终止，即达到halt语句，并产生输出值 $\overline{z}$。

下面给出程序P是终止的、部分正确的和完全正确的形式定义。

（1）如果对于使$\phi(\bar{\xi})$为真的每一输入$\bar{\xi}$，P的计算终止，则称P关于ϕ是终止的。

（2）如果对于使$\phi(\bar{\xi})$为真而且P的计算终止的每一个$\bar{\xi}$，$\psi(\bar{\xi}、P(\bar{\xi}))$为真，则称P关于$\phi$和$\psi$是部分正确的，其中$P(\bar{\xi})=\bar{z}$。

（3）如果对于使$\phi(\bar{\xi})$为真的每一$\bar{\xi}$，P的计算终止并且$\psi(\bar{\xi}, P(\bar{\xi}))$为真，则称P关于$\phi$和$\psi$是完全正确的。

程序P的部分正确性的证明方法如下：设程序P的流程如图2-2所示。为了证明P关于ϕ和ψ的部分正确性，把输入谓词$\phi(\bar{\xi})$附在A点上，输出谓词ψ附在C点上。然后在B点切割P，即把P的流程分成三条路径：①从A到B，包含弧1和2。②从B经循环回到B，包含弧3、4和5。③从B经弧3、6和7到C。这三条路径分别记为α、β、γ。对任意输入$\bar{\xi}$，程序P终止。执行P时，首先必须通过路径α，其次沿路径β循环若干次，最后经过γ结束P的执行。这样，所有的执行都被这三条路径所覆盖。要验证程序P的部分正确性，就要验证三条路径α、β、γ的部分正确性。对每条路径的起点和终点都找出相应的描述程序变量之间关系的谓词$P(\bar{x}、\bar{y})$。要验证一条路径是部分正确的，只要验证：对某些值$\bar{x}$和$\bar{y}$，该路径的初始谓词为真，经该路径后，它的结尾谓词对新值$\bar{x}$和$\bar{y}$也为真。

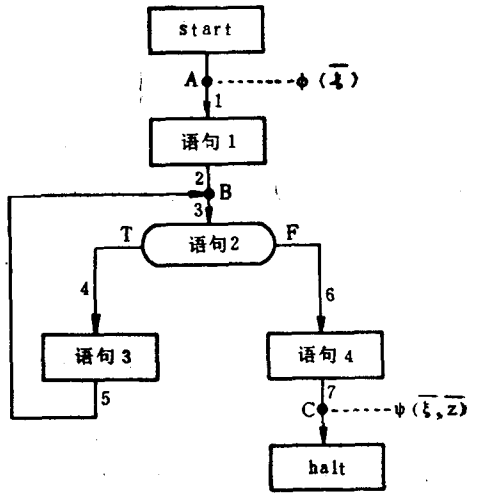

图2-2 程序P的流程

对于程序P的终止性，只要证明在点B处的谓词$P(\bar{x}, \bar{y})$中的程序变量$\bar{y}$的值是严格递减的就可以了。因为不存在无限下降的自然数列，所以循环（即路径β）不可能无限制地进行下去，因此程序P必然终止。

以上简要介绍了程序P的部分正确性和终止性的证明方法。在第三章中，还要介绍用数学归纳法证明程序的正确性和用良序原则证明程序的终止性。这说明可以严格地证明程序的正确性。但是这种证明十分麻烦。因篇幅所限，本书以下各章在证明程序正确性时均不采用这种严格证明的方法。

至于算法正确性问题，可以分两步来考虑，即考虑算法的终止性和算法的每一步是否都正确。

售货员路线问题的算法的正确性是可以证明的，因为调查了每一条旅行路线后，当然也就调查了最小费用路线。按照算法就把这条最小费用路线保存起来。如果没有更小费用的路线来代替它，它就总是被保存着。这个算法是可终止的，因为旅行路线是有限的（城市个数是有限的）。这种算法叫穷举法，是最拙劣的算法。

算法的正确性不蕴含算法的有效性。一个算法是正确的，但可能是无效的。如上例中当城市的数目是20时，若采用穷举法，在一台中型计算机上求最佳路线，则需要70个世纪！

许多计算机科学工作者正采用不同方法来研究程序正确性证明问题。美国的博耶(Boyer)

和穆尔(Moore)博士在这方面做出了卓越贡献，从而获得了1983年首次约翰·麦卡锡（John McCarthy）奖。

博耶和穆尔于1979年提出了新的数学归纳法。这种方法把递归函数论、数学归纳法和定理机器证明巧妙地结合起来，开辟了程序正确性证明的新途径。博耶和穆尔依据这种方法研制的程序证明系统取得了很大的成功，已经证明了密码算法、快速串搜索算法所对应的FORTRAN程序的正确性。

博耶-穆尔的程序证明系统主要由两个程序组成，一个是验证条件生成器，一个是定理证明器。使用该系统分两步：①验证条件生成器把要验证的程序、该程序的输入/输出规范和一些归纳不变式作为输入；验证条件生成器产生一些称为“验证条件”的公式，这些公式蕴涵着源程序按规范执行。②把验证条件转交给定理证明器，若定理证明器能确定验证条件均为定理，则源程序就按规范执行。即源程序是正确的。

五、算法的实现

求解问题的算法被设计出来并证明其正确性之后，就应该着手实现这个算法，即根据算法编制计算机程序。

在算法的描述中，有些步骤不能直接转换成代码形式，而往往需要一些完整的子程序。这需要另下功夫来编制。另外，在编写代码之前，还要设计数据结构的完整体系，用来表达所用模型的各个方面。为此必须注意解决以下问题：

有哪些变量？各是什么类型？

需要多少数组？它们的规模多大？

用什么样的结构组织数据比较合适？

需要什么样的子程序？有没有现成的？

用什么语言比较合适？

算法的实现方式，对解题速度和所需要的内存容量都有很大影响。所以应该予以充分注意。

六、算法的分析

进行算法分析的理由有两个。其一是为了对算法的某些特定的输入，估计或限界该算法所需要的内存和运行时间。因为CPU时间和内存是很珍贵的资源，这些资源往往是许多用户共享的。其二是为了建立衡量算法优劣的标准，用以比较同一问题的不同算法，我们将在下一节介绍有关的基础知识。但深入的研究属于计算机科学的另一分支——计算复杂性。本书第十、第十一章对这方面的内容也将给予适当介绍。

七、程序的测试

编制完实现算法的程序后，应该对程序进行测试。在这个阶段要校正输入中的差错和进行语法检查，最后把程序放在多种情况下运行。

计算机科学中的程序测试类似于其它自然科学中的实验。程序测试可以被认为是对程序应完成的任务的实验证实，同时确定程序的使用范围。测试方法一般有两种：一种是对程序的各个分支（每条路径）进行测试（即白盒测试）。另一种是检查对给定的输入是否有指定的输出（即黑盒测试）。

怎样选择程序测试中的输入？这个问题没有一个一般的答案。通常情况下，输入的集合是十分庞大的，一个完全的测试既不可能也无必要。实用中，常常对输入数据作有代表性的