高等学校教材

计算机算法设计与分析

郑丽英　孟昱煜　王海涌　邬开俊　编著

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

高等学校教材

计算机算法设计与分析

郑丽英　孟昱煜　王海涌　邹开俊　编著

中国铁道出版社

2009年·北京

内 容 简 介

计算机算法是计算机科学和计算机应用的核心。无论是计算机系统、系统软件的设计，还是为解决计算机的各种应用课题做的设计都可归结为算法的设计。

本书以计算机算法设计策略为知识单元，围绕算法设计的基本方法，对计算机应用领域中许多常用的非数值算法做了系统的描述，并分析了这些算法所需的时间和空间。全书共分十三章，前七章介绍了递归技术、分治策略、动态规划、贪心法、回溯法及分支限界法等基本设计方法，第八到十三章介绍 NP 完全理论和 NP 难题、近似算法、字符串匹配、随机算法、概率算法的相关知识，并对近年来广泛受到关注的网络路由算法及生物信息算法的基本设计方法作了介绍。书中既涉及传统算法的实例分析，更有算法领域热点研究课题追踪，具有较高的实用价值。

本书可作为高等院校计算机及相关专业本科生及研究生的教学用书，也可作为从事计算机科学、工程和应用的工作人员的自学教材和参考书。

图书在版编目(CIP)数据

计算机算法设计与分析/郑丽英等编著. —北京：中国铁道出版社，2009.6

高等学校教材

ISBN 978-7-113-09629-8

Ⅰ.计… Ⅱ.郑… Ⅲ.①电子计算机-算法设计-高等学校-教材②电子计算机-算法分析-高等学校-教材

Ⅳ.TP301.6

中国版本图书馆 CIP 数据核字(2009)第 017007 号

书　　名：计算机算法设计与分析

作　　者：郑丽英　孟昱煜　王海涌　邬开俊　编著

责任编辑：阚济存　　**电话**：010-51873133　　**电子信箱**：td51873133@163.com

封面设计：崔　欣

责任校对：孙　玫

责任印制：陆　宁

出版发行：中国铁道出版社(100054，北京市宣武区右安门西街 8 号)

网　　址：http://www.tdpress.com

印　　刷：三河市华业印装厂

版　　次：2009 年 6 月第 1 版　　2009 年 6 月第 1 次印刷

开　　本：787 mm×1092 mm　1/16　**印张**：19.5　**字数**：487 千

印　　数：1～2000 册

书　　号：ISBN 978-7-113-09629-8/TP·3150

定　　价：36.00 元

前　言

算法设计与分析是计算机科学的一个主要研究领域。本课程是计算机、管理信息系统、系统工程、应用数学和计算数学等专业高年级学生和研究生的一门重要专业课程。它的主要目的是讲授在计算机应用中常常遇到的实际问题的有效解法，讲授设计和分析各种算法的基本原理、方法和技术，以培养读者在选择或设计一个算法时，思考下列问题：这个算法是否有效？这个算法有多好？是否还有更好的算法？用什么方法和技巧去获得更好的算法？从而使得所设计算法的时空复杂性最优，进而为编写高效的程序、开发优秀软件奠定基础。

本书旨在全面介绍计算机算法设计与分析的内容。力争做到取材先进、内容实用、重点突出、少而精，便于自学；并注意收录一些典型问题的最新研究成果。期望读者通过本课程的学习，接受算法设计基础研究的初步训练，了解算法设计的最新应用领域，培养独立开展科研工作的能力和创新意识。本书可作为计算机科学及相关专业高年级学生和研究生课程的教材；也可供其他从事计算机研究与应用的人员参考。

全书共分十三章。第一章介绍算法分析的基本概念和基本理论，介绍设计算法的基本技术和分析算法复杂性的基本方法；第二章介绍递归技术，重点讲述了递归方程的解法；第三章介绍分治策略，它是设计有效算法最常用的策略，也是必须掌握的方法；第四章介绍动态规划算法，以具体实例详述动态规划算法的设计思想以及算法的设计要点；第五章介绍贪心算法，它是一种重要的算法设计策略，按其设计出的许多算法能导致最优解；第六章和第七章分别介绍了回溯法和分支限界法，这两章所介绍的算法适合于处理难解问题，其解题的思想各有特色，值得学习和掌握；第八章介绍了 NP 完全问题及近似算法，重点剖析一些典型、能启迪人们思维的算法设计思想；第九章介绍了字符串精确匹配和近似匹配技术，也重点剖析一些典型、能启迪人们思维的算法设计思想；第十章介绍了网络路由算法，这是当前计算机网络应用研究中受到广泛关注的研究内容；第十一章介绍了随机算法；

第十二章介绍了概率算法；第十三章简单介绍了算法设计在当前计算机最新应用领域即生物信息处理中的应用研究。

本书采用C/C++语言为算法描述手段，在保持其优点的同时，尽量使算法的描述简明、清晰。

为了帮助读者加深对知识的理解，各章配有难易适当的习题，以满足不同程度读者练习的需要。

编者自20世纪90年代中后期以来，一直从事算法设计与分析等领域的学习、教学和研究。本书在授课讲义基础上，参考国内外有关论著编写而成。在编写过程中，兰州交通大学计算机系的老师提出了许多宝贵的建议，在此深表感谢。

本书由兰州交通大学郑丽英、孟昱煜、王海涌、邬开俊编著。书中的第二、十、十一、十二章由郑丽英编写，第三、四、五章由孟昱煜编写，第一、八、十三章由王海涌编写，第六、七、九章由邬开俊编写。

在此书稿出版之际，感谢兰州交通大学教务处对此书的支持和资助，同时感谢研究生黄涛、张明昭、梁瑞艳、周红强、杜小刚等帮助校对整理部分文档和绘制部分图表。

由于编者学识有限，难免有缺点和错误。书中如有不足之处，敬请诸位专家和读者批评指正，使本书在使用中不断得到改进，日臻完善。

编　者

2009年6月

目　录

第一章 导 论

计算机算法是计算机科学和计算机应用的核心，无论是计算机系统、系统软件和解决计算机的各种应用课题都可归结为算法的设计。通常，给了一个问题，我们关心三件事：

1. 怎样找到解决此问题的有效算法？
2. 如何比较解决同一问题的不同算法？
3. 如何判断一个算法的优点？

简单地说，解决这三个问题就是应该掌握常规的或经典的算法设计方法，掌握算法分析的基本手段。

第一节 算法与程序

一、算法的概念及特性

对于计算机科学来说，算法(Algorithm)的概念是至关重要的。例如在一个大型软件系统的开发中，设计出有效的算法将起决定性的作用。

通俗地讲，算法是指解决问题的一种方法或一个过程。更严格地讲，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。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，且满足下述几条性质：

1. 输入：有零个或多个由外部提供的量作为算法的输入。
2. 输出：算法产生至少一个量作为输出。
3. 确定性：组成算法的每条指令是清晰的、无歧义的。
4. 可行性：算法中有待实现的运算都相当基本(都是基本运算)，每种运算至少在原理上能由人用纸和笔在有限的时间内完成。整数算术运算是可行性运算的一个例子，而实数算术运算则不是可行的，因为某些实数值只能由无限长的十进制数展开式来表示，像这样的两个数相加就违背可行性这一特性。
5. 有穷性：算法中每条指令的执行次数是有限的，执行每条指令的时间也是有限的。

但是我们要注意以下两点：

(1)如果只满足前四条的只能叫做计算过程，算法则必须在有穷步内终止运行。

(2)由于研究算法的最终目的是为了有效地求出问题的解,那就需要将算法投入到计算机上运行,因此对算法的讨论不能只研究到它能在有穷步内终止就结束,而应对有穷性作进一步的研究,即对算法的效率作出分析。

例如,在象棋比赛中,对任意给定的一种棋局,都可以设计出一种算法来判断这一棋局是否可以导致赢局。这样的一个算法需要从开局起对所有棋子可能进行的移动以及相应的对策作逐一的检查。为了做出应走哪些棋子的决策,其计算步骤虽是有穷的,但实际上即使在最先进的计算机上运算也要千千万万年。由此可知,只应把那些在相当有穷步内就终止的算法投入计算机中运行,而对不能在相当有穷步内终止的算法则不必在计算机上运行,免得耗费计算机的宝贵资源。对这类问题的求解应另辟捷径。

二、算法与程序的区别

求解一个给定的可计算或可解的问题,不同的人可以编写出不同的程序,这里存在两个问题:一是与计算方法密切相关的算法问题;二是程序设计的技术问题。算法和程序之间存在密切的关系。

著名计算机科学家、Pascal 语言发明者 N·Wirth 教授提出:程序=算法+数据结构。也就是说,计算机按照程序所描述的算法对某种结构的数据进行加工处理。设计程序首先要了解需要解决的问题,再提出解决此问题的方法和步骤。程序(Program)与算法不同。程序是算法用某种程序设计语言的具体实现,目的是加工数据,算法就是解决问题的方法,算法处理的对象就是数据(即如何加工数据),数据结构是问题的数学模型。

程序可以不满足算法的可行性。例如操作系统,它是一个在无限循环中执行的程序(设计它的目的是为了控制作业的运行,当没有作业运行时,这一计算过程并不终止,而是处于等待状态,一直等到一个新的作业进入),因而不是一个算法。然而我们可把操作系统的各种任务看成是一些单独的问题,每一个问题由操作系统中的一个子程序通过特定的算法来实现。该子程序得到输出结果后便终止。

三、算法的制定过程

算法是一组有穷的规则,它们规定了解决某一特定类型问题的一系列运算,是对解题方案的准确与完整的描述。制定一个算法,一般要经过设计、确认、分析、编码、测试、调试、计时等阶段。因此学习计算机算法必须涉及这些方面的内容。在这些内容中有许多都是现今重要而活跃的研究领域。

对算法的学习主要包括五个方面的内容:

1. 设计算法

算法设计工作是不可能完全自动化的,应学习了解已经被实践证明是有用的一些

基本的算法设计方法，这些基本的设计方法不仅适用于计算机科学，而且适用于电气工程、运筹学等领域。

2. 表示算法

描述算法的方法有多种形式，例如自然语言和算法语言，各自都有适用的环境和特点。

3. 确认算法

算法确认的目的是使人们确信这一算法能够正确无误地工作，即该算法具有可计算性。正确的算法用计算机算法语言描述，构成计算机程序，计算机程序在计算机上运行，得到算法运算的结果。

一旦设计出了算法，就应证明它对所有可能的合法输入都能算出正确的答案，这一工作称为算法确认(algorithm validation)。确认的目的在于使我们确信这一算法将来正确无误地工作，而与写出这一算法所用的程序语言无关。一旦证明了算法的正确性，就可以将其写成程序，再将程序放到机器上，而在运行之前，实际上还应证明程序是正确的，即证明程序对所有可能的合法输入都能得到正确的结果，这一工作称为程序证明(program proving)。这一领域是当前很多计算机科学工作者集中研究的对象[算法在计算机科学中的重要体现在：有超过1/3的图灵奖获得者(18/49)，其成果与算法有关，最著名的是：Edsger W. Dijkstra 求最短路径的 Dijkstra 算法、PV 操作、goto 有害等]，还处于初期阶段。在这一领域的工作还没取得突破性的进展之前，为了增强对所编制程序的置信度，只能用对程序的测试来权益代替。

4. 分析算法

算法分析是对一个算法需要多少计算时间和存储空间作定量的分析。上面提及的只应把能在相当有穷步内终止的算法实际投入计算机运行，细心的读者可以发现“相当”一词用得非常含糊，能否对有穷步给出一个数量界限呢？算法分析(analysis of algorithms)是对一个算法需要多少计算时间和存储空间作定量的分析。分析算法不仅可以预计所设计的算法能在什么样的环境中有效地运行，而且可以知道在最好、最坏和平均情况下执行得怎么样，还可以对解决同一问题不同算法的有效性做出比较判断。

5. 验证算法

用计算机语言描述的算法是否可计算、有效合理，须对程序进行测试，测试程序的工作由调试和作时空分布图组成。调试(debugging)程序是在抽象数据集上执行程序，以确定是否会产生错误的结果，若有，则修改程序。但是，这一工作正如 E. Dijkstra 所说的：“调试只能指出有错误，而不能指出它们不存在错误。”尽管如此，在程序正确性证明没有突破性进展的今天，调试仍是不可缺少且必须认真进行的一项重要工作。作时空分布图是用各种给定的数据执行经过调试认为是正确的程序，并测定计算出结果所花去的时间和空间，以验证以前所作的分析是否正确和指出实现最优化的有效逻辑位置。

四、算法设计的原则

设计算法时我们应当严格考虑以下四点：

1. 正确性(Correctness)

首先，算法应当满足以特定的“规格说明”方式给出的需求。对算法是否“正确”的理解可以有以下四个层次：

(1)算法中不含语法错误。

(2)算法对于几组输入数据能够得出满足要求的输出结果。

(3)算法对于精心选择的、典型的、苛刻的几组输入数据能够得出满足要求的结果。

(4)算法对于一切合法的输入数据都能得出满足要求的结果。

通常以第(3)层意义的正确性作为衡量一个算法是否合格的标准。因为作为输入，我们有时候不可能提前做出所有的预期。

2. 可读性(Readability)

算法主要是为了人的阅读与交流，其次才是为计算机执行。因此算法应该易于人的理解；另一方面，晦涩难读的算法易于隐藏较多错误而难以调试；有些程序设计者总是把自己设计的算法写的只有自己才能看懂，这样的算法反而没有太大的价值。

3. 健壮性(Rubustness)

当输入的数据非法时，算法应当恰当地作出反映或进行相应处理，而不是产生莫名其妙的输出结果。这就需要我们一定要充分地考虑异常情况，并且处理出错的方法不应是中断程序的执行，而应是返回一个表示错误或错误性质的值，以便在更高的抽象层次上进行处理。

4. 高效率与低存储量需求

通常，效率指的是算法执行时间，而存储量指的是算法执行过程中所需的最大存储空间。两者都与问题的规模有关。

第二节　算法的描述

算法的描述方法可以归纳为以下几种：

(1)自然语言。

(2)图形，如 N-S 图、流程图，图的描述与算法语言的描述对应。

(3)算法语言，即计算机语言、伪代码。

(4)形式语言，用数学的方法，可以避免自然语言的二义性。

用各种算法描述方法所描述的同一算法，该算法的功能是一样的，允许在算法的描述和实现方法上有所不同。

人们的生产活动和日常生活离不开算法，都在自觉不自觉地使用算法，例如人们到

商店购买物品，会首先确定购买哪些物品，准备好所需的钱，然后确定到哪些商场选购、怎样去商场、行走的路线。若物品的质量好如何处理，对物品不满意又怎样处理，购买物品后做什么等。以上购物的算法是用自然语言描述的，也可以用其他描述方法描述该算法。

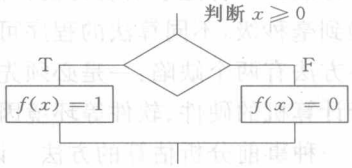

图 1-1 用流程图图形描述算法

图 1-1 是用流程图描述算法的例子，其函数为

$$f(x)=\begin{cases}1 & (x\geqslant 0)\\ 0 & (x<0)\end{cases}$$

描述算法可以有多种方式，如自然语言方式、表格方式等。在本书中，我们采用 C/C++语言来描述算法。C/C++语言的优点是类型丰富、语句精练，具有面向过程和面向对象的双重特点。用 C/C++来描述算法可使整个算法结构紧凑，可读性强。在本书中，有时为了更好地阐明算法的思路，我们还采用 C/C++与自然语言相结合的方式来描述算法。

第三节 算法的评价与优化

一、算法评价

对于解决同一个问题，往往能够编写出许多不同的算法。例如，对于数据的排序问题，常见的有：枚举排序、冒泡排序、插入排序、快速排序、希尔排序等多种方法，对于这些排序算法，它们各有优缺点，其算法如何有待用户的评价。因此，对问题求解的算法优劣的评定称为“算法评价”。算法评价的目的，在于从解决同一问题的不同算法中选择出较为合适的一种算法，或者是对原有的算法进行改造、加工，使其算法更优、更好。

一般对算法进行评价主要有四个方面：

1. 算法的正确性

正确性是设计和评价一个算法的首要条件，如果一个算法不正确，其他方面就无从谈起。一个正确的算法是指在合理的数据输入下，能在有限的运行时间内得到正确的结果。通过对数据输入的所有可能情况的分析和上机调试，以证明算法是否正确。

2. 算法的简单性

算法简单有利于阅读，也使得证明算法正确性比较容易，同时有利于程序的编写、修改和调试。但是算法简单往往并不是最有效。因此，对于问题的求解，我们往往更注意有效性。有效性比简单性更重要。

3. 算法的运行时间

算法的运行时间是指一个算法在计算机上运算所花费的时间。它大致等于计算机执行简单操作（如赋值操作、比较操作等）所需要的时间与算法中进行简单操作次数的乘积。通常把算法中包含简单操作次数的多少叫做算法的时间复杂性。它是一个算法

运行时间的相对量度，一般用数量级的形式给出。度量一个程序的执行时间通常有两种方法。一种是事后统计的方法。因为很多计算机内部都有计时功能，有的甚至可精确到毫秒级，不同算法的程序可通过一组或若干组相同的统计数据以分辨优劣。但这种方法有两个缺陷：一是必须先运行依据算法编制的程序；二是所得时间的统计量依赖于计算机的硬件、软件等环境因素，有时容易掩盖算法本身的优劣。因此人们常常采用另一种事前分析估算的方法。该方法求出算法的一个时间界限函数（是一些有关参数的函数）。一个用高级程序语言编写的程序在计算机上运行时所消耗的时间取决于下列因素：

(1)依据的算法选用何种策略。

(2)问题的规模。例如求100以内还是1 000以内的素数。

(3)书写程序的语言。对于同一个算法，实现语言的级别越高，执行效率就越低。

(4)编译程序所产生的机器代码的质量。

(5)机器执行指令的速度。

显然，同一个算法用不同的语言实现，或者用不同的编译程序进行编译，或者在不同的计算机上运行时，效率均不相同。这表明使用绝对的时间单位衡量算法的效率是不合适的。撇开这些与计算机硬件、软件有关的因素，可以认为一个特定算法“运行工作量”的大小，只依赖于问题的规模（通常用整数量 n 表示），因此，事前估算方法求出的算法的一个时间限定函数，通常是问题规模的一个函数，称为时间复杂度。

一个算法是由控制结构（顺序、分支和循环三种）和原操作（指固有数据类型的操作）构成的，则算法时间取决于两者的综合效果。为了便于比较同一问题的不同算法，通常的做法是，从算法中选取一种对于所研究的问题（或算法类型）来说是基本运算的原操作，以该基本操作重复执行的次数作为算法的时间度量。

一般情况下，对一个问题（或一类算法）只需选择一种基本操作来讨论算法的时间复杂度即可，有时也需要同时考虑几种基本操作，甚至可以对不同的操作赋以不同权值，以反映执行不同操作所需的相对时间，这种做法便于综合比较解决同一问题的两种完全不同的算法。

由于算法的时间复杂度考虑的只是对于问题规模 n 的增长率，则在难以计算基本操作执行次数（或语句频度）的情况下，只需求出它关于 n 的增长率或阶即可。

4. 算法所占用的存储空间

算法在运行过程中临时占用的存储空间的大小被定义为算法的空间复杂性。空间复杂性包括程序中的变量、过程或函数中的局部变量等所占用的存储空间以及系统为了实现递归所使用的堆栈两部分。算法的空间复杂性一般也以数量级的形式给出。

类似于算法的时间复杂度，以空间复杂度作为算法所需存储空间的量度，记作：

$$S(n)=O(f(n))$$

其中 n 为问题的规模（或大小）。一个上机执行的程序除了需要存储空间来寄存本

身所用指令、常数、变量和输入数据外，也需要一些对数据进行操作的工作单元和存储一些为实现计算所需信息的辅助空间。若输入数据所占空间只取决于问题本身，和算法无关，则只需要分析除输入和程序之外的额外空间，否则应同时考虑输入本身所需空间(和输入数据的表示形式有关)。若额外空间相对于输入数据量来说是常数，则称此算法为原地工作。又如果所占空间量依赖于特定的输入，则除特别指明外，均按最坏情况来分析，即以所占空间可能达到的最大值作为其空间复杂度。

二、算法优化

算法优化的几种常用方法。

(1)空间换时间算法中的时间和空间往往是矛盾的，时间复杂性和空间复杂性在一定条件下也是可以相互转化的，有时候为了提高程序运行的速度，在算法的空间要求不苛刻的前提下，设计算法时可考虑充分利用有限的剩余空间来存储程序中反复要计算的数据，这就是“用空间换时间”策略，是优化程序的一种常用方法。相应的，在空间要求十分苛刻时，程序所能支配的自由空间不够用时，也可以以牺牲时间为代价来换取空间，由于当今计算机硬件技术发展很快，程序所能支配的自由空间一般比较充分，这一方法在程序设计中不常用到。

(2)尽可能利用前面已有的结论：比如递推法、构造法和动态规划就是这一策略的典型应用，利用以前计算的结果在后面的计算中不需要重复。

(3)寻找问题的本质特征，以减少重复操作：算法的复杂度分析不仅可以对算法的好坏作出客观的评估，同时对算法设计本身也有着指导性作用，因为在解决实际问题时，算法设计者在判断所想出的算法是否可行时，通过对算法作事先评估能够大致得知这个算法的优劣，进而作出决定是否采纳该算法。这样就能避免把大量的精力投入到低效算法的实现中去，特别是现在举行的各级信息学奥林匹克竞赛对程序的运行时间都有着相当严格的限制，掌握好算法复杂度的大致评估方法就显得尤为重要。

第四节 算法的复杂度

同一个问题往往可以由不同的算法解决，但是算法不同，效率也不同。我们怎么知道同一个问题的不同算法哪一个效率高？通常在评价算法性能时，复杂性是一个重要的依据。算法的复杂性的程度与运行该算法所需要的计算机资源的多少有关，所需要的资源越多，表明该算法的复杂性越高；所需要的资源越少，表明该算法的复杂性越低。算法在计算机上执行运算，需要一定的存储空间存放描述算法的程序和算法所需的数据，计算机完成运算任务需要一定的时间。根据不同的算法写出的程序放在计算机上运算时，所需要的时间和空间是不同的，算法的复杂性是对算法运算所需时间和空间的一种度量。不同的计算机其运算速度相差很大，在衡量一个算法的复杂性要注意到这

一点。计算机的资源，最重要的是运算所需的时间和存储程序和数据所需的空间资源，算法的复杂性有时间复杂性和空间复杂性之分。

对于任意给定的问题，设计出复杂性尽可能低的算法是在设计算法时考虑的一个重要目标。另外，当给定的问题已有多种算法时，选择其中复杂性最低者，是在选用算法时应遵循的一个重要准则。因此，算法的复杂性分析对算法的设计或选用有着重要的指导意义和实用价值。

让我们从比较两对具体算法的效率开始：

【例 1-1】 已知不重复且已经按从小到大排好的 m 个整数的数组 $A[1,m]$（为简单起见，还设 $m=2^k$，k 是一个确定的非负整数）。对于给定的整数 c，要求寻找一个下标 i，使得 $A[i]=c$；若找不到，则返回一个 0。

一个简单的算法是：从头到尾扫描数组 A。照此，或者扫到 A 的第 i 个分量，经检测满足 $A[i]=c$；或者扫到 A 的最后一个分量，经检测仍不满足 $A[i]=c$。我们用一个函数 Search 来表达这个算法：

```
int Search (int c,int A[] )
  {
    int i,j=1; //初始化
    //在还没有到达 A 的最后一个分量且等于 c 的分量还没有找到时,
    查找下一个分量并且进行检测
    While ((A[i++]!=c)&&(j<=m))do
          j=j+1;
    if (j>m)   return (0 );//在数组 A 中找不到等于 c 的分量
          else return (j ); //在数组中找到等于 c 的分量
    }
```

容易看出，在最坏的情况下，这个算法要检测 A 的所有 m 个分量才能判断在 A 中找不到等于 c 的分量。

另一个算法利用到已知条件中 A 已排好序的性质。它首先拿 A 的中间分量 $A[m/2]$ 与 c 比较，如果 $A[m/2]=c$ 则解已找到，如果 $A[m/2]>c$，则 c 只可能在 $A[1]$，$A[2]$，…，$A[m/2-1]$ 之中，因而下一步只要在 $A[1]$，$A[2]$，…，$A[m/2-1]$ 中继续查找；如果 $A[m/2]<c$，则 c 只可能在 $A[m/2+1]$，$A[m/2+2]$，…，$A[m]$ 之中，因而下一步只要在 $A[m/2+1]$，$A[m/2+2]$，…，$A[m]$ 中继续查找。不管哪一种情形，都把下一步需要继续查找的范围缩小了一半，再拿这一半的子数组的中间分量与 c 比较，重复上述步骤。照此重复下去，总有一个时候，或者找到一个 i 使得 $A[i]=c$，或者子数组为空（即子数组下界大于上界）。前一种情况找到了等于 c 的分量，后一种情况则找不到。

这个新算法因为有反复把供查找的数组分成两半，然后在其中一半继续查找的特征，我们称为二分查找算法。它可以用函数 B_Search 来表达：