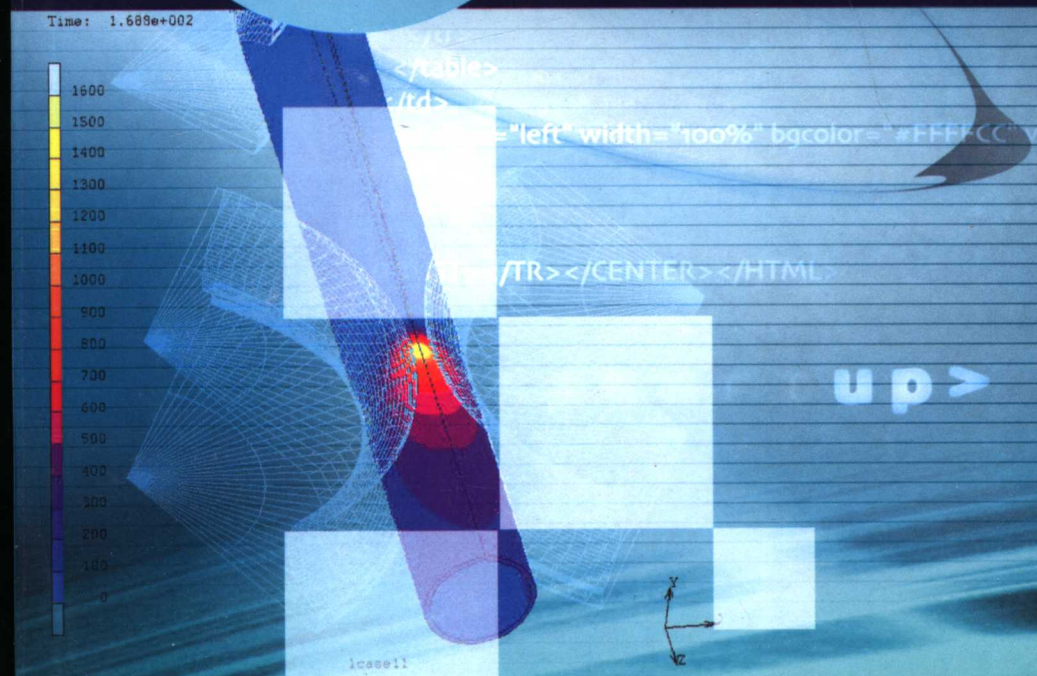
普通高等教育“十一五”规划教材

高等院校计算机科学与技术系列教材

算法分析与设计

邓向阳　万婷婷　编著



冶金工业出版社

普通高等教育“十一五”规划教材
高等院校计算机科学与技术系列教材

算法分析与设计

邓向阳　万婷婷　编著

北　京

冶 金 工 业 出 版 社

内 容 简 介

本书是根据普通高等教育“十一五”国家级规划教材的指导精神而编写的。

算法分析与设计是一门理论性与实践性兼顾的课程，是计算机科学与技术应用的核心，本书主要介绍算法设计的基本方法、基本理论，突出了计算机科学领域中的非数值算法和算法分析的基本知识。本书共分十三章，第一、二章介绍基本概念，第三至十一章介绍分别讨论各类方法。如：树及其操作、集合操作、排序、查找、图、动态规划、贪心法、分治法等。最后讨论了傅氏变换和 NP 完全问题。为进一步研究算法奠定了基础。

本书可作为计算机软件专业本科类和研究生教材，也可供其他从事计算机研究与应用人员参考。

图书在版编目（CIP）数据

算法分析与设计 / 邓向阳，万婷婷编著. —北京：冶金工业出版社，2006.7
ISBN 7-5024-4013-5

I. 算… II. ①邓…②万… III. ①电子计算机—算法分析—高等学校—教材②电子计算机—算法设计—高等学校—教材 IV. TP301.6

中国版本图书馆 CIP 数据核字（2006）第 063578 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
广州锦昌印务有限公司印刷；冶金工业出版社发行；各地新华书店经销
2006 年 7 月第 1 版第 1 次印刷
787mm × 1092mm 1/16； 17 印张； 388 千字； 262 页
30.00 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、本书背景

随着科学技术的进步，软件的理论与开发方法不断涌现。算法分析与设计是指导计算机软件开发的理论科学。算法分析与设计已成为计算机科学与技术中一门重要内容。

目前，全国各地本科院校普遍扩招，本科生人数迅速增长，这给他们的就业带来了巨大的压力。而当前本科生的就业情况还不如专科学生，究其原因，所用教材与实际应用脱轨是一主要因素。针对现有教材质量较差、品种单一、版本陈旧、实用性和可操作性不强等原因，肩负着应用型人才培养的高等本科院校急需一系列符合当前教学改革需要的教材。

本书是根据普通高等教育“十一五”国家级规划教材的指导精神而编写的，旨在介绍算法分析与设计的方法知识。

算法分析与设计在计算机软件专业的重要性是众所周知的，在计算机科学技术领域占据了无可争议的基础地位。鉴于此我们编写了本书，其目的是向读者提供一本关于算法分析与设计的教科书，以使更多读者受益。

二、本书结构

本书由五部分组成：

第一部分：递归技术。包括绪论，递归技术。

第二部分：树、图。包括树，图及有向图的应用。

第三部分：查找、排序、集合操作。

第四部分：动态规划、贪心法、回溯法。

第五部分：分治与平衡、NP 完全问题。

三、本书特点

本书系统、全面地研究和借鉴了国外相关教材先进的教学方法，结合国内院校教学实际和先进的教学成果，根据教育部“十一五”国家级规划教材应用型本科教育的指导思想编写，具有实用性和可操作性，与时俱进，与当前就业市场结合得更加紧密。

本书的着眼点在于算法分析与设计的介绍，内容丰富，除了传统的排序、字符串匹配和图论问题以外，还介绍了数值计算、组合数学的问题，且每章后面配有相应的练习，供读者学习和实践。

四、编写方法

本书是作者根据近十年来对算法分析与设计的教学与研究，以及作者领导或参与的二十项软件项目开发的实际应用经验，并结合软件开发新技术编写而成。根据过去的教学经验，读者学习一门新技术，教材是非常重要的。因此，在编写之前，在各方面进行了充分的准备。

五、如何使用本书

算法分析与设计分为 5 部分。根据读者的实际情况，教师在教授本书时，可以按照自己的风格和喜好删除章节，也可以根据教学目标灵活调整章节顺序。

第一章　绪论（2 学时）
第二章　递归技术（6 学时）
第三章　树（4 学时）
第四章　图及有向图的应用（6 学时）
第五章　无向图（4 学时）
第六章　查找（6 学时）
第七章　排序（6 学时）
第八章　集合操作（6 学时）
第九章　动态规划（6 学时）
第十章　贪心法（5 学时）
第十一章　回溯法（5 学时）
第十二章　分治与平衡（2 学时）
第十三章　NP 完全问题（2 学时）

六、适用对象

算法分析与设计是计算机软件专业的一门学科。讲授时间为 60 学时。本书可作为计算机软件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。

在选修本课程之前，读者应该具有计算机的基础知识，同时具有 Visual 类语言或者 C/C++的编程经验，会有助于深入理解信息系统开发过程。

由于算法分析与设计的知识面广，在介绍中不能面面具到。加上编者水平有限，编写时间仓促，书中如有疏漏和不足之处，敬请广大读者批评指正。联系方式如下：

电子邮箱：service@cnbook.net

网址：www.cnbook.net

本书的电子教案及习题参考答案可在该网站免费下载。此外，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

2006 年 5 月

目　录

第1章 绪 论

信息的表示是计算机科学的基础。毋庸置疑，人们编写程序是为了解决问题。要用计算机解决一个具体的问题，编写程序的工作就要经历这样一些步骤：首先使问题有一个简明、严格的提法，然后设计求解方法，使这种解法能为计算机实现，还要经过测试和文本编制，最后还要评价解法的优劣。本章将概括地介绍我们在这些步骤中所采取的方法。

1.1 引论

算法（algorithms）的研究是计算机科学的核心课题之一。早在电子计算机问世以前，就有人开始了算法的研究，并创造了许多有效的算法。中国古代数学以算法为主要特征。吴文俊指出：我国传统数学在从问题出发以解决问题为主旨的发展过程中，建立了以构造性与机械化为其特色的算法体系，这与西方数学以欧几里得《几何原本》为代表的所谓公理化演绎体系正好遥遥相对。肇始于我国的这种机械化体系，在经过明代以来几百年的相对消沉后，由于计算机的出现，已越来越为数学家所认识和重视，势将重新登上历史舞台。吴文俊创立的几何定理的机器证明方法（称吴方法），用现代的算法理论，焕发了中国古代数学的传统算法，享有很高的国际声誉，因此于2001年他获得了第一届国家最高科学技术奖。

20世纪上半叶，科学研究方式归结为两种方式：理论+实验。后来由于计算机技术能力的开发，计算成为第三种重要手段。未来的趋势是，“理论+实验+计算”将成为标准的科学研究方法。那么，计算机如何按照人的意愿进行计算呢？这就要靠算法。因此，毫不夸张地说，算法既是数学科学的重要基础，也是计算机科学的核心。

学会设计数据结构与算法，可以让我们编写出高效率的程序。也许有人要问，在计算机速度日新月异的今天，为什么还需要高效率的程序？计算机的功能越强大，人们就越想去尝试更复杂的问题，而更复杂的问题就需要更大的计算量，计算速度和存储容量上的革新仅仅提供了处理更复杂问题的有效工具，所以高效率的程序永远不会过时。

程序的高效性的要求不会与合理的设计和清晰的编码相矛盾。设计高效率的程序是基于良好的数据结构与算法，而不是基于“编程小技巧”。因此大多数计算机科学系在设置课程时，都重视学习基本的软件工程原理，以及数据结构与算法设计。

那到底究竟什么是算法呢？关于算法的严格定义，如果建立了图灵（A.M Turing）可计算的概念之后，是可以用图灵机形式地来描述的。在此，只能给出“算法”概念的非形式的描述。

简而言之，算法是解决某一特定类型问题的有限运算序列。对于某一类问题的任何初始输入，它能一步一步地计算，通过有限步之后，计算终止，并产生一个输出。从数据表示的观点来看，算法是指对数据结构施加的一些操作，例如对一个线性表进行检索、插入、删除等操作。

【例1-1】算法E（欧几里得算法）。约定两个正整数m和n，求它们的最大公因子，即能够同时整除m和n的最大的正整数。

E1：[求余数]以 n 除 m 并令 r 为所得余数（特设 $0 \leqslant r < n$）。

E2：[余数是否为 0]若 r = 0，算法结束；n 即为答案。

E3：[互换]置 m←n，n←r，并返回步骤 E1。

一个算法的每一步（例如上边的步骤 E1）以一个方括号中的一个短句开始，它尽可能简短地概括这一步的主要内容。这一短句通常也出现在一个与这个算法相对应的框图（例如图 1-1）中。借助于框图，能够直观地看出这个算法的流程。

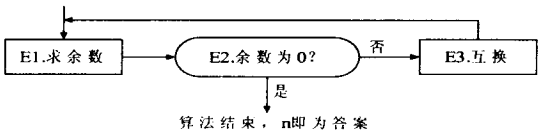


图 1-1 欧几里得算法框图

上述计算过程规定了三个计算步骤，m 和 n 都是给定的有限数，每次相除后得到的余数 r 若不为 0，总有 r<min(m, n)。这就保证了经过有限次循环之后，总有 r=0 的情况发生。这时循环就会终止，而且产生一个输出，这就是一个算法。

由上述算法的例子可知，算法具有以下五大特性：

（1）确定性。一个算法中给出的每一个计算步骤，必须是精确的定义、无二义性的。

（2）有穷性。一个算法在执行有穷个计算步骤后必须停止。

（3）可行性。算法中要执行的每一个计算步骤都是可以在有限时间内做完的。可行性、有穷性和确定性是相容的。

（4）输入。一个算法一般都要求一个或多个输入信息。

（5）输出。一个算法一般有一个或多个输入信息。它们通常可以被解释成为“对输入的计算结果”。

一个算法很类似于过程、方法、规程、程序等，但它并不等同于一个计算过程后的一个过程。算法与过程的区别在于：对任何合法的输入，算法要在有限时间内通过有穷步计算后终止；而过程则可以是有穷的，也可以是无穷的，无穷过程永远不会终止。而算法与程序的区别在于：一个计算机程序往往是指用某种机器语言所书写的一个计算过程，但一个特定的算法不一定表现为一个计算机程序，它可以采用多种方式描述，如图解描述、语言描述等，它和计算机程序不一定有某种必然的联系。

人们对常用的数据结构与算法的研究已经相当透彻，可以归纳出一些设计原则：

（1）每一种数据结构与算法都有其时间、空间的开销和收益。当面临一个新问题时，设计者应掌握怎样权衡时空开销和算法有效性的方法。这就要对算法分析的原理有深刻的理解，并且还需要了解所使用的物理介质的特性。

（2）与开销和收益有关的是时间与空间的权衡。通常可以用更大的空间开销来换取时间的收益，反之亦然。时间与空间的权衡普遍地存在于软件开发的各个阶段中。

（3）数据结构与算法为应用服务。必须先了解应用的需求，再寻找或设计与实际应用相匹配的数据结构。

（4）程序员应该充分地了解一些常用的数据结构与算法，避免不必要的重复设计工作。

具体的细节将在下面几节中详细介绍。

1.2 从问题到程序

从直觉上来讲，问题无非是一个需要完成的任务，即对应一组输入且有一组相应的输出。许多问题的最初提法常常既不精确又不简练。还有一些问题，譬如说，如何保护濒危动物？如何做一道人人都喜欢吃的绝佳美食？这些问题也许不可能简单而精确地表述成能为计算机解决的形式。即使是认为能精确地表述的参数，而那些合理的参数值只有通过实验才能确定。因此，要用计算机解决问题，必须首先用简明、严格的方式将问题表述清楚。换句话说，成功的关键在于明确要解决的问题。

如果能用一个形式模型来刻画问题的某些方面，那么这样做是很有益的。一旦将问题形式化，就可以依据这个严格的模型对问题求解，容易知道是否已有现成的程序可以利用。即使没有现成的程序可用，至少可以利用这个形式模型所具有的种种性质来构造好的解法。

数学或其他科学中的几乎所有分支都可以作为某一类具体问题的抽象模型。例如，在数值计算问题中常用的数学模型为线性方程组（用于求解电源的额定功率，或求解桥梁所能承受的压力等）；符号与文本处理问题中常用字符串及形式语法作为模型，这类问题包括编译（从程序设计语言到机器语言的程序翻译），信息检索（譬如从火车时刻表中查找出所需车次的详细资料）等。

对问题建立了适当的数学模型以后，就可以依据这一模型求解。最初的目标是要给出一个算法形式的解法。例如，上面那个例子中的 r←m%n 就可以作为一个指令，它会在进行有限次求余动作后执行完这条指令。在一个算法中，有些指令可能需要重复执行多次，因此指令的执行次数可能远远超过指令的条数。但无论对于什么样的输入，算法中任何一条指令都不能被无穷多次地执行。

和大多数的教科书相似，本书采用了一种介于高级程序设计语言和伪语言之间的语言形式——类 C 语言。所谓伪语言是将某种程序设计语言的基本结构与非形式的英文或中文叙述语句相结合的产物。类 C 语言采用了 C 语言的基本结构，它的程序描述和数据描述能力强，并且有较好的结构性和可读性。下面举例说明在编写计算机程序的过程中所采取的基本步骤和一般方法。

【例 1-2】考虑交叉路口交通信号灯的设计问题（图 1-2）。需要一个这样的程序：它以路口上可通过的所有转弯路线（直行通过的路线也看作一种转弯）作为输入，程序将这些转弯分组输出，使得同一组内的转弯互不冲突。为了使路口的信号管理简单有效，还要求使转弯分组的组数尽可能少，这样，如果用信号控制器的每一相位指挥一组内的转弯，那么就能以最少的相位数有效地指挥路口的交通。

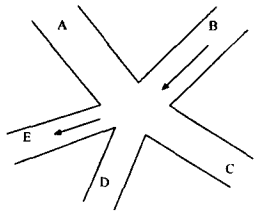


图 1-2 一个交叉路口

图 1-2 是一个相当复杂的交叉路口的图示。图中的 B 和 E 是两条单行路线，其他路线

都可以双向行驶，因此这里共有 13 种不同方向的转变路线。其中有些转变是互不冲突的，例如 CD（从 C 到 D）和 BC（从 B 到 C）就是这样。另外还有些转弯是互相冲突的，例如 CA 和 DC，因此不允许它们同时通行。这个路口的交通信号灯必须禁止像 CA 与 DC 这样的冲突转变同时通行，但是应允许像 CD 与 BC 这样的无冲突转变同时通行。

为了解决这一问题，首先要找一个适当的抽象模型将问题形式化。可以用一个称为图的数学结构作为这个问题的模型。一个图由一些点与一些连接点的线组成，通常称这些点为顶点，称这些线为边。可以用一个具有 13 个顶点的图来刻画这个路口的交通情况：其中的每一个顶点代表一种转弯；如果某两个顶点所代表的转变是相互冲突的，则在这两个顶点之间连接一条边。这样得到的图如图 1-3 所示。表 1-1 是这个图的另一种表示方法。如果表中第 i 行与第 j 行列所对应的两个顶点之间有边相连，则在表中第 i 行，第 j 列的位置填写 1。

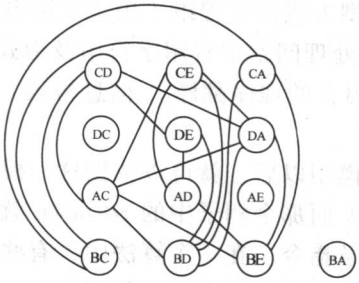


图 1-3　转弯冲突图

表 1-1　转弯冲突表

	CD	CE	CA	DC	DE	DA	AC	AD	AE	BC	BD	BE	DC
CD					1	1	1			1			
CE						1	1	1		1	1		
CA										1	1	1	
DC													
DE	1							1			1		
DA	1	1					1				1	1	
AC	1	1				1					1	1	
AD		1			1							1	
AE													
BC	1	1	1										
BD		1	1		1	1	1						
BE			1			1	1	1					
DC													

利用这样的图，可以将交通信号灯的设计问题形式化为图和着色问题。所谓着色就是要求对图中的每个顶点着染一种颜色，并使得任意两个有边相连的顶点具有不同的颜色，要求不用颜色的数目最小。例如，考虑图 1-4 中的 5 个结点时。将结点任意编号 1 至 5，然后按编号给结点着色，从贪心法的观点考虑，每次给尽可能多的结点着色。首先用颜色 1 给结点 1、2 着色，然后用颜色 2 给 3 和 4 结点着色，最后用颜色 3 给结点 5 着色。不难

看出，上面的问题就是要用尽可能少的几种颜色对上面表示冲突转弯的图进行着色。

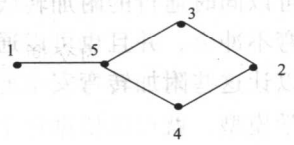


图 1-4 着色问题例图

由于图着色问题的计算量太大，因此产生了寻求最优解的可靠性问题。对此有三种解决方法。如果图很小，可以用穷试法寻求最优解，这样可使问题得到彻底解决，但是对于较大的图，无论怎样提高程序的效率，穷试法的计算量都太大以致根本无法实现。第二种方法是对需要着色的图进行具体分析，找出它的一些特点，这些特点也许能使图具有某些特殊性质， 使我们不必穷试所有可能，即可求得最优解。第三种方法是将问题稍加修改，把要求放宽一点，只要求令人满意的解，而不一定要求最优解。这样也许能有一个快速算法，使得它对较小的图的结果很接近最优解。由于一般交叉路口的情况不会比图 1-2 所示的情况更复杂， 所以采用第三种方法求解。所谓的“启发式方法”就是可以快速求得满意解（不一定是最优解）的一种方法。

以下的贪心算法就是解决图着色问题的一种合理的启发式方法：首先用第一种颜色给尽可能多的顶点着色，然后用第二种颜色给尽可能多的尚未着色的顶点着色，如此做下去。对于每一种新颜色，按下述步骤进行着色：

（1）选一个尚未着色的顶点，给它以新色。

（2）逐个检查所有尚未着色的顶点，对每一个这样的顶点，查看是否有边将它与某个已着新色的顶点相连。如果没有这样的边，则给该顶点着以新色。

至于贪心算法，本书会在第八章给出详细的介绍。

以图 1-3 为例说明贪心算法。假设从 CD 开始执行，首先给 CD 着红色，然后再给 CE、CA 和 DC 着红色，因为这四个顶点中任意两个顶点之间没有边相连；但是 DE 不能着红色，因为 CD 与 DE 之间有边，同理，DA、AC 及 AD 也不能着红色，因为它们都至少与一个已着红色的顶点相连；然而 AE 可着红色，后面还有 BC、BD 及 BE 都不能着红色，而 BA 可以着红色。

再开始用第二种颜色，譬如给 DE 着橙色，DA 也可着橙色，但是 AC 不行，因为 DA 与 AC 之间有边；同理 AD 也不能着橙色；AE 已着过红色；BC 可着橙色； 剩下的其他未着色顶点都有边与某橙色顶点相连，所以它们都不能再着橙色。

这时还有 AC、AD、BD 及 BE 是尚未着色的。如果将 AC 着为黑色，那么 AD 也可着黑色；而 BD 及 BE 则不能着黑色，这两个顶点可以用第四种颜色，譬如用蓝色。最后，着色的情况总结为表 1-2。

表 1-2 图 1-2 中的着色结果

颜色	转弯路线	附加转弯路线
红色	CD、CE、CA、DC、AE、BA	-----
橙色	DE、DA、BC	DC、AE、BA
黑色	AC、AD	CA、DC、AE、BA
蓝色	BD、BE	DC、AE、BC、BA

表的中间一栏是贪心算法对转弯冲突图的着色结果。同一颜色的那些转变是可以无冲突地同时通行的，此外还有一些可以同时通行的附加转线路线。这些转变在表中右边一栏里列出，它们与左边一栏中的转弯不冲突，并且也可以通过贪心算法求出。当交通灯允许一种颜色的转弯通行时，它也可以让这些附加转弯安全通行。

现在给问题建立了适当的数学模型，就可以依据这个模型严格地表述算法。一开始，算法的雏形常常是用普通的叙述语句表达的，这样的叙述语句必须逐步修改成更详细、更精确的指令。例如，在前面关于图着色的贪心算法的叙述中，有这样一句话："选一个尚未着色的顶点"。为了使这种非形式的算法变成一个程序，必须经过几个修改阶段，逐步将算法形式化（这一过程称为逐步加细），最后得到一个程序，其中的每条指令都是某一程序语言手册中定义过的。

假设需要着色的图是 G，集合 V1 包括图中所有未被着色的结点，着色开始时 V1 是 G 所有结点集合。NEW 表示已确定可以用新颜色着色的结点集合。

从 G 中找出可用新颜色着色的结点集：

```
NEW={ };
   for(第一个结点,每个属于 V1 的结点,下一个结点)
     if (v 与 NEW 中所有结点间都没有边)
       {  从 V1 中去掉 v;
          NEW=NEW∪{v};
        }
```

对集合和图的操作：

判断一个集合是否为空：isSetEmpty(V1);

置一个集合为空：emptySet(NEW);

从集合中去掉一个元素：removeFromSet(V1,v);

向集合里增加一个元素：addToSet(NEW,v);

检查结点 v 与结点 NEW 中各结点之间在图 G 中是否有边连接：

```
notAdjacentWithSet(NEW,v,G);
```

有了图、集合这样的结构和基于其上的操作，程序的实现非常简单。

则用类 C 语言描述的算法为：

```
int colorup( Graph G)
 {  int color=0;  V1=G.V;
    while(!isSetEmpty(V1))
     { emptySet(NEW);
       while(∃v∈V1 && notAdjacentWithSet(NEW,v,G))
          {
           addToSet(NEW,v);
           removeFromSet(V1,v);
          }
       ++color;
      }
     return(color);
   }
```

至于类 C 语言描述，将在 1.5 节中详细介绍它的指令代表的具体含义。

由以上的例子可知，算法设计和分析的步骤可概括为：