面向21世纪课程教材
Textbook Series for 21st Century

算法与数据结构
——C语言描述(第2版)

张乃孝 编著

高等教育出版社

面向21世纪课程教材

算法与数据结构

——C语言描述(第2版)

张乃孝 编著

高等教育出版社

内 容 提 要

本书以数据结构为主线，算法为辅线组织教学内容。全书共分 10 章：绪论、线性表、字符串、栈与队列、二叉树与树、集合与字典、高级字典结构、排序、图和算法分析与设计。本书体系完整，概念清楚，内容充实，取材适当。第一版在 2004 年被评为“北京市高等教育精品教材”。

这次再版，采用“数据结构作为抽象数据类型的物理实现”观点，在内容和形式上都进行了许多改进和扩充。提高了抽象数据类型在教学中的地位和作用；更加突出了重点，提高了全书的可读性，还补充了习题，增加了索引。

由于在编写中注意到知识模块的独立性和相关性，不同专业和不同水平的学生可以根据需要组合使用。本书既可以作为信息与计算机专业大学本科的“数据结构”教材，也可以作为一般理工科专业本科和计算机专业专科学生学习相关课程的教材或教学参考书。

图书在版编目（CIP）数据

算法与数据结构：C 语言描述 / 张乃孝编著. —2版. —北京：高等教育出版社，2006.1
ISBN 7-04-018576-8

Ⅰ.算... Ⅱ.张... Ⅲ.①算法分析－高等学校－教材②数据结构－高等学校－教材 Ⅳ.①TP301.6②TP311.12

中国版本图书馆CIP数据核字（2005）第159767号

策划编辑 刘 茜　　**责任编辑** 刘 茜
封面设计 于文燕　　**责任印制** 陈伟光

出版发行	高等教育出版社	**购书热线**	010-58581118
社　　址	北京市西城区德外大街 4 号	**免费咨询**	800-810-0598
邮政编码	100011	**网　　址**	http://www.hep.edu.cn
总　　机	010-58581000		http://www.hep.com.cn
		网上订购	http://www.landraco.com
经　　销	蓝色畅想图书发行有限公司		http://www.landraco.com.cn
印　　刷	涿州市星河印刷有限公司	**畅想教育**	http://www.widedu.com
		版　　次	2002 年 9 月第 1 版
开　　本	787×1092　1/16		2006 年 1 月第 2 版
印　　张	23.25	**印　　次**	2006 年 1 月第 1 次印刷
字　　数	440 000	**定　　价**	26.00 元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换。

物料号 18576-00

前　言

关于**算法**的研究已经有数千年的历史。计算机的出现，使得用机器自动解题的梦想成为现实，人们可以将算法编写成程序交给计算机执行，使许多原来认为不可能完成的算法变得实际可行。**数据结构**的概念最早由 C. A. R. Hoare 和 N. Wirth 在 1966 年提出。大量关于**程序设计**理论的研究表明：对大型复杂程序的构造进行系统而科学的研究，必须对这些程序中所包含的数据结构进行深入的研究。事实上，程序就是在数据的某些特定的结构和表示的基础上对于算法的描述。不清楚解决问题的算法就无法决定如何组织数据，反之算法的实现又在很大程度上依赖于数据的具体组织。简言之，算法与数据结构是程序设计中相辅相成、不可分割的两个方面。因此，1976 年 N. Wirth[6] 用"**算法 + 数据结构 = 程序**"这个公式表达了算法与数据结构的联系和它们在程序中的地位。

抽象数据类型起源于 20 世纪 70 年代，可以定义成：有一定行为（操作）的抽象（数学）类型。它**抽象**出数据类型的使用要求，而把它的具体表示方式和运算的实现细节都**隐藏**起来。它支持数据类型的实现与使用分离的原则，是一种十分有效的对问题进行抽象与分解的思维工具，后来发展成为**面向对象**技术与方法的主要理论基础。

从抽象数据类型的观点出发，数据结构可以理解成为"**抽象数据类型的物理实现**"。对于数据结构的学习和研究，主要围绕两个问题：一个是如何具体表示抽象数据类型中的数学模型；另一个是如何给出抽象数据类型中需要的操作的具体实现。用上述观点，可以从更加抽象的角度理解数据结构与算法的关系，也容易解释数据的逻辑结构、存储结构与运算的三者关系。

目的与动机

"数据结构"（或称"数据结构与算法"或"算法与数据结构"）是计算机科学的基础课程，它被独立列入大学本科的教学计划有 30 多年的历史。其主要目的是使读者全面地理解数据结构和算法的概念、掌握设计数据结构与算法的主要原理和方法、比较不同数据结构和算法的特点。通过学习和实践，提高学生使用计算机解决问题的能力。

1998 年，北京大学把"算法与数据结构"作为第一批理科各系主干基础课列入学校的教学计划。经北大信息学院和数学学院共同推荐，著者连续六年被校长聘任为该课程的全校

主持人，负责组织理科各院系的主讲老师制定教学大纲、编写教材和教学辅导用书、交流教学经验等工作。

本书的第一版就是在这个背景下，邀请了北大物理学院、化学学院和计算中心的几位主讲老师与著者共同编写的。从2000年开始，本书作为北京大学主干课“算法与数据结构”的教材在校内试用，2002年得以在高等教育出版社正式出版，并得到广大校内外老师和学生的支持和鼓励，被评为“2004年北京市高等教育精品教材”。为了回报广大读者，使得这本书能够更上一个台阶，成为过得硬的“精品”教材，在高等教育出版社的大力支持下，著者决定认真修改后出此新版。

第二版的主要修改

- 采用“数据结构是抽象数据类型的物理实现”观点，改写了第一章和其他有关部分，提高了抽象数据类型在本书中的地位和作用；
- 增加了“优先队列”、“矩阵”、“广义表”和“动态存储管理”等内容；
- 增加了集合的“位向量”表示和字典的“字符树”表示，把原来第6章加以扩充后，分成两章；
- 增加了“外存数据的组织”、“散列文件”和“外排序”等基本文件知识的介绍；
- 增加了小标题，使得内容的重点更加突出，并且对于书中的叙述进行了加工调整，提高了全书的层次感、简明性和可读性；
- 进一步推敲了基本概念的表述，增加了主要概念的索引；
- 补充了习题；
- 删除了附录。

本书的组织

本书以数据结构为主线，算法为辅线组织教学内容。修改后，全书共分以下10章。

第1章**绪论**。首先通过一个实例讲解使用计算机处理实际问题的过程，然后对抽象数据类型、数据结构和算法等分别展开讨论。通过本章学习，可以对全书有个鸟瞰。

第2章**线性表**。首先介绍线性表的逻辑结构，引入线性表的抽象数据类型，接着具体给出两种线性表的实现：顺序表和链表。然后介绍一个线性表的应用实例。最后，作为线性表的扩展，讨论了矩阵和广义表的表示，并且介绍了结点的动态分配和回收等问题。

第3章**字符串**。介绍了字符串的抽象数据类型和表示方法，重点讲解无回溯模式匹配算法。

第4章**栈与队列**。除了讨论栈和队列的概念、抽象数据类型、表示方法和实现算法外，还将给出一些常见的应用例子。

第5章**二叉树与树**。首先引入二叉树的概念和抽象数据类型；具体给出两种二叉树的实现；介绍了二叉树的两个重要的应用实例。然后，介绍树的基本概念和抽象数据类型，讨论了树的实现问题；最后讨论了二叉树与树林的相互对应关系。

第6章**集合与字典**。首先介绍集合的概念、抽象数据类型、位向量表示法和实现算法；然后讨论字典的相关内容。重点介绍了字典的的顺序表示和散列表示。

第7章**高级字典结构**。首先论述了字典与索引的关系，然后进一步讨论字典的其他实现。包括字符树、二叉排序树和多级索引结构。其中多级索引结构主要用于组织外存文件的索引。

第 8 章**排序**。介绍了 10 种排序方法，讨论了这些算法的复杂性及稳定性。

第 9 章**图**。介绍了图的基本概念和抽象数据类型，讨论了图的周游方法和图的相邻矩阵及邻接表的表示方法，重点讲解求图的最小生成树(林)和求图中结点间最短路径等算法，最后介绍了两个典型的应用。

第 10 章**算法分析与设计**。主要给读者概括地介绍算法的分析和设计的主要技术，以便于将本书所学到的算法归类整理，达到开阔思路、提高观点、增强兴趣的目的。

读者对象与使用方法

本书可以作为各种类型大学的本科和专科的“数据结构”、“数据结构与算法”或者“算法与数据结构”等课程的教材。本书在编写中注意到知识模块的独立性和相关性，不同专业的学生可以根据不同的需要进行组合使用。

基础部分

1 第 1 章绪论(除去 1.3.4、1.4.2 和 1.4.3)。

2 第 2 章线性表(除去 2.4、2.5 和 2.6)。

3 第 4 章栈与队列(除去 4.3 和 4.6)。

4 第 5 章二叉树与树(除去 5.4、5.5、5.6 和 5.7)。

5 第 6 章集合与字典(除去 6.2.1 和 6.5.4)。

6 第 8 章排序(除去 8.2.3、8.2.4、8.3.2、8.5 和 8.6.2)。

7 第 9 章图(除去 9.5.2、9.6. 和 9.7)。

提高部分

8 应用实例：分散在 2.4、4.3、4.6、5.4、9.6 和 9.7 等节中。

9 增强基础：学习在基础部分各章中被除去的那些(有关文件的内容除外)内容。

10 第 3 章字符串。

11 第 7 章高级字典结构。

任选部分

12 文件内容：分散在 1.3.4(外存数据的组织)、6.5.4(散列文件)、7.6(索引文件)和 8.6.2(外排序)。

13 第 10 章算法分析与设计。

除去基础部分中的内容应该顺序学习以外，其他部分均可以灵活安排。著者建议：

◆ 对于生物、化学和医农等专业的本科学生的相关课，可以主要学习 1 ~ 5 的内容，和 6 ~ 8 中部分内容。

◆ 对于一般理工科的本科学生，可以主要学习 1 ~ 9 的内容，和 10 ~ 11 中部分内容。

◆ 对于信息或计算机(或者与它们要求相似的)专业的本科学生，可以主要学习 1 ~ 11 的内容，和 12 ~ 13 中部分内容。

此外，在使用本书时，请读者注意以下几点：

语言 考虑到目前国内“计算概论”(或“基本程序设计”)课程的教学情况，同时也因为用 C 语言描述算法十分有效，分析使用 C 语言描述的算法又比较直观，因此本书采用 C 语言作为描述语言，并且假设读者都具备了这方面的基础知识。

索引　为了便于使用，对于比较重要的概念都建立了对应的索引，读者可以利用本书最后的索引查阅。

习题　学习算法与数据结构必须理论与实践相结合，所以在理解书本知识的同时，应该尽可能做更多的习题。在本书的每章最后，都配备了比较丰富的习题，根据不同的目的，分成复习题、算法题和应用（上机）题三种。这些题目的参考答案可以在本书参考文献[3]中找到。对于初学者而言，最难完成的是算法题。设计算法的关键在于设计的思路，不同的思路产生出不同的算法。读者通过自己的学习、思考和实践，就能从中体会到算法与数据结构的真谛，提高自己分析问题和解决问题的能力。

学时　根据著者的经验，对于比较熟悉 C 语言的对象，讲授全部内容需要 60～70 学时。

感谢

此书能够顺利与读者见面，首先应该感谢的是长期支持和鼓励著者从事教学和研究的北京大学（特别是北大的数学学院和计算机系）。能够在北大这片沃土上，实践著者“追求真理、探索人生”的理想是著者今生的最大满足。为此，著者数十年来孜孜以求，从来不敢懈怠；平平淡淡，确也无怨无悔。借此机会略表感谢与怀念之情。

感谢在近二十多年中，与著者合作编写教材的许多老师（按照时间顺序）：唐世渭、杨冬清、刘玉明、许卓群、邵维忠、孙玉方、裘宗燕、韩玉真、陈光和刘筠等。在与他们的合作过程中，著者不仅仅学到许多的知识，同时也学到了许多为人的道理。其中裘宗燕教授还阅读了本书的大部分修改稿，提出许多中肯的意见。

衷心感谢曾经和将要使用本教材的老师、同学和所有读者，你们的每条宝贵意见都是对著者最大的支持与鼓励。

十分感谢高等教育出版社计算机分社的刘建元社长和刘茜、康兆华、耿芳以及所有相关的工作人员，著者的几本主要教材，都是在他们的支持下得以顺利出版。

非常感谢著者所有的亲人，没有他们的理解、支持和关爱，著者很难能够平安度过这坎坷的前半生。特别要感谢的是赵素兰女士，在近四十年的共同生活中，是她的无私奉献构筑了如今幸福美满的家庭。最后，真诚地期望这个世界变得更加美好；衷心地祝愿我们的子孙后代生活得更加快乐。

张乃孝

znx@pku.edu.cn

2005 年 6 月 11 日

目　录

1 绪　论

“算法与数据结构”是学习计算机科学的基础课程。其主要目的是使读者较全面地理解算法和数据结构的概念、掌握应用数据结构与算法的主要原理和方法、比较不同数据结构和算法的特点。通过学习和实践，使学生能够提高使用计算机解决实际问题的能力。学好这门课对于所有研究或使用计算机的读者都是很有裨益的。

本章首先通过一个实例讲解使用计算机处理实际问题的过程，然后对一些最基本、最重要的概念分别展开讨论。

本章重点是：理解从问题到程序的主要过程；体会数据结构、算法和抽象数据类型在问题求解过程中的作用；了解数据结构的主要概念和分类方法；了解算法的概念和主要设计、分析方法。

通过本章学习，使读者对全书有个整体的鸟瞰，以使读者在后继各章学习时，能更好理解它们的地位和相互关系。

1.1 从问题到程序

用计算机实现**问题求解**，实质上就是在计算机中建立一个解决问题的模型。用来表示问题或处理问题的模型可以有不同的抽象形式：容易被人理解但不太严格的**需求模型**；比较抽象但很精确的**数学模型**；容易被计算机理解或执行的**实现模型**。

程序是使用程序设计语言精确描述的实现模型，它是问题求解的一个可以在计算机上运行的模型。程序中描述的数据用来表示问题中涉及的对象，程序中描述的过程表示了对于数据的处理算法；通过接受实际问题的输入，经过程序的运行，便可以得到实际问题的一个解。

为一个实际问题建立一个正确的求解程序，通常可以分成以下几个阶段：

分析阶段。这阶段的任务，首先是弄清用户的需求是什么，设计者根据它进行深入分析，使用规范说明语言（或数学语言等工具）给出系统的需求模型（或数学

模型)。

设计阶段。这阶段的任务是建立求解系统的实现模型,重点是算法的设计和数据结构的设计。对于大型的复杂的系统,这一阶段往往还包括抽象数据类型或者模块的设计。一般而言,设计过程需要从粗到细,经过多次精化才能完成。

编码阶段。它的主要任务是采用适当的程序设计语言(C 语言、C++语言或是 Java 语言等)把设计阶段的成果,编写成可执行的程序。

调试和维护。其任务包括使用足够的例子调试编写的程序,发现和排除程序代码中的错误;在计算机上执行程序,获得问题的解;也包括在系统投入运行后,解决在使用过程中发现的隐含错误和根据使用中提出的要求进行必要的维护和完善。

显然,与本课程关系最为密切的是设计阶段,但由于设计阶段介于分析和编码这两个阶段之间,因此讨论中也需要牵涉到它们。下面通过一个实例,具体展开问题求解的主要过程,重点讨论问题的分析和算法与数据结构的设计等。

1.1.1 问题分析与抽象

为了能正确地解决问题,必须首先深刻地理解需要解决的问题。只有在深刻地认识了这个问题以后,才能着手确定这个问题的解决方法。

信号灯问题

考虑一个多叉路口(见图 1.1),在这个路口中,共有五条道路相交,其中 C 和 E 是单行线,其他为双行线。提出的任务是:为这个路口设计一个安全有效的交通信号灯的管理系统。

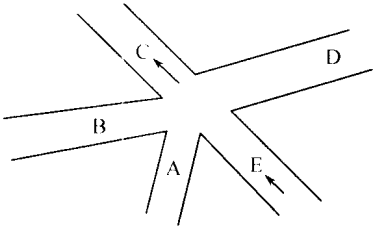

图 1.1　一个交叉路口的模型

分析

为了完成上述任务,首先需要研究一下这个路口所有车辆的行驶路线,找出存在的冲突。这可以归结为对车辆的可能行驶方向作某种分组,分组的结果满足:任一个组中各个方向行驶的车辆可以同时安全行驶(不发生碰撞)。显然,对这个问题存在许多不同分组方案,其中最简单的方案就是把每个可能的行驶路线分为一组。例如:根据这个路口的实际情况,可以确定 13 个可能通行方向:A→B,A→C,A→D,B→A,B→C,B→D,D→A,D→B,D→C,E→A,E→B,E→C,E→D。如果把它们分在 13 个组中,每组只包含 1 种通行方向,当然非常安全。但是,这种方案是不会在实际中采用的。因为如果分组越少,可以同时行驶的车辆也就越多,路口的运行效率就越高。我们需要的就是如何能够找到一个比较高效的方案。

抽象

根据这个路口的实际情况，在 13 个可能通行方向中，有些方向明显不能同时进行，如 A→B 与 B→C 等。为了叙述方便，我们下面把每个通行方向，诸如 A→B 简写成 AB，并且用一个小椭圆的图把它框起来，看成一个结点；在不能同时行驶的路线间画一条连线（表示它们互相冲突），便可以得到图 1.2 所示的网状图形。

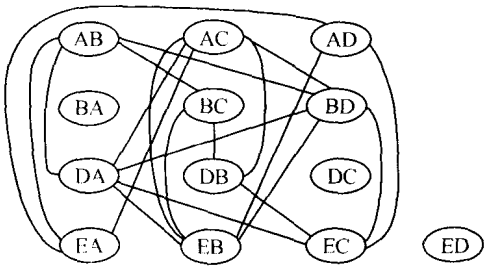

图 1.2 交叉路口行驶冲突的模型

经过上面讨论，把原来的一个实际（信号灯）问题转变成了另一个比较抽象问题。这个问题可以借助图的模型清楚而严格地表达出来：要求将图 1.2 中的结点分组，使有线相连（互相冲突）的结点不在同一个组里。根据前面同样理由，我们也希望对于这个问题求得一个分组数比较少的解。习惯上把只要满足安全性的普通解称为是这个问题的一个**可行解**；把分组数最少的解称为**最优解**；而分组数接近最优解的可行解称为**次优解**。

着色问题

由于上面用了图的抽象表示，使得表达出的问题已经比最初提出的问题更一般、更抽象了。如果把图 1.2 中的一个结点理解为一个国家，结点之间的连线表示两国有共同边界，上述问题就是著名的**着色问题**：即如何使用尽量少的几种颜色，将图中所有国家着色，使得任意两个相邻的国家颜色都不相同。解决了着色问题，也就解决了包括信号灯在内的一大类问题。

1.1.2 程序的设计与实现

一般来说，一个问题的解决可以有许多办法，由于使用的工具是计算机，所以在选择方法时必须充分考虑到计算机的特点和条件，才能找到比较巧妙的办法，比较快而且准确地计算出需要的结果。

对于前面抽象出来的着色问题，下面介绍一种简单的求解方法，目的在于使读者从中体会从问题抽象的模型到实现模型的设计过程。

选择算法

穷举法

要求一般着色问题的最优解，可以采用穷举法，具体做法是：从分为 1，2，3，… 组开始考察，逐个列举出所有可能的着色方案，检查这样的分组方案是否满足要求。首先满足要求的分组，自然是问题的最优解。

具体来讲，假设求解的图中有 n 个结点，首先考察如果放在一个组里（这时显然只有一种着色方式）行不行？即考察组里的结点是否有线相连？如果没有，最优解已经找到；否则考察如果放在两个组里行不行？这时需要逐个穷举出所有可能的分成两个组的方案，这个数可能很大。例如：两个组按 $1:n-1$ 分配，就有 $C_n^1=n$ 种方案，按 $2:n-2$ 分配，就有 $C_n^2=n(n-1)/2$ 种方案等等。当考察了所有放在两个组里可能的方案后，如果都不行的话，接着考虑分为三组、四组的各种组合。依此类推，最多到分成 n 组，总能成功。

但是，对规模大的问题，由于求解时间会随着实际问题规模的增长而呈指数级上升，这类穷举法可能使计算机无法承受（有耐心的读者，不妨可以对上述信号灯问题亲自动手模拟一下这个穷举法的求解过程）。

当然，对于一般（任意规模）的着色问题，存在一些比穷举法快的求最优解的算法，因为超出本课教学范围，不在此展开讨论。实际上，对于一个具体的着色问题（例如本节开头提出的信号灯问题），并没有必要一定找到其最优解。下面给出的是一个比穷举法快得多的，但是通常能够求出次优解的算法。

贪心法

求着色问题的次优解，一种简单的方法称为**贪心法**：先用一种颜色给尽可能多的结点上色，只要这些结点之间没有边相连，然后再用另一种颜色在未着色结点中给尽可能多的结点上色，如此反复直到所有结点都着色为止。

把这种方法应用到前面交叉路口的例子中，依次选红、蓝、绿和白色，可能得到如下一种分组（这个分组已经是问题的一个最优解了）：

红色：AB AC AD BA DC ED

蓝色：BC BD EA

绿色：DA DB

白色：EB EC

选择抽象数据类型

为了便于给出上述算法的实现，可以按照抽象数据类型的观点，先把被处理的对象加以抽象。在着色问题中具体解决：使用什么抽象数据类型来表示地图，以及使用什么样的抽象数据类型表示一组国家等。

在前面的讨论中，我们已经十分自然地使用一个**图**（图是**图论**研究的对象，也是一种重要的抽象数据类型）表示地图。对于国家的分组，因为每一组内的国家之间没有必要区分先后顺序，所以可使用国名（图中结点名）的**集合**表示。

本章后面将会详细介绍抽象数据类型的概念，并且指出不同抽象数据类型的主要特征是由它们具有的**行为**（或称操作）决定。这里读者不妨把抽象数据类型理解为常用初等数据类型（例如整数类型等）的扩充。对于这里使用的图和集合而

言,它们的行为在后面均有说明。

算法的描述

选择了抽象数据类型以后,我们可以比较精确地描述前面的算法。假设需要着色的图是 G,G 中所有结点的集合记为 $G.V$,集合 $V1$ 存放图中所有未被着色的结点,集合 NEW 存放可以用某颜色着色的所有结点。

从 $V1$ 中找出可用某新颜色着色的所有结点的工作,可以用伪码描述如下:

```
置 NEW 为空集合;
for 每个 v ∈ V1 do
    if v 与 NEW 中所有结点间都没有边
    将 v 加入 NEW ; 从 V1 中去掉 v ;
```

这个伪码描述如果能够执行,集合 NEW 中就得到一组可以用新颜色着色的结点。着色程序可以反复调用这段伪码,直到 $V1$ 为空,每次调用选择一种新颜色,这段伪码执行的次数就是需要的不同颜色个数。下面给出的一段伪码里涉及对集合和图的若干操作(行为),例如,判断元素 v 是否属于集合 $V1$ 表示为 $v \in V1$;从集合 $V1$ 中去掉一个元素 v 表示为 remove(V1,v);向集合 NEW 里增加一个元素 v 用 add(NEW,v)表示,判断集合 $V1$ 是否空集合表示为 isEmpty(V1)等(假设这些都是抽象数据类型集合应该提供的行为);另外,检查结点 v 与结点集合 NEW 中各结点之间在图 G 中是否有边连接,用函数 notAdjacentWith (NEW,v,G)表示。有了这些操作,贪心法着色的实现,可以非常直接地用下面方式给出。

算法 1.1 贪心法着色

```
int colorUp(Graph G){
    int color = 0;                    /* 记录使用的颜色数 */
    set V1 = G.V;                     /* V1 初始化为图 G 的结点集 V */
    set NEW;
    while (! isEmpty(V1)){
        NEW = {};
        while ( ∃v ∈V1. notAdjacentWith(NEW,v,G)){
            add(NEW,v);
            remove(V1,v);
        }
        ++color;
    }
    return color;                     /* 返回使用的颜色数 */
}
```

数据结构的设计

如果集合和图是程序设计语言中预定义的类型，remove(V1,v)和 add(NEW,v)等就应该是语言中预定义的内部函数，算法 1.1 就几乎可以直接上机运行。否则程序员需要自己用语言所提供的类型机制实现这些抽象数据类型(集合、图等)，这些正是数据结构设计要讨论的内容。

算法的精化与代码生成

在数据结构确定以后，算法的描述可以进一步根据设计的数据结构进行精化。例如，在集合和图的数据结构确定后，notAdjacentWith(NEW,v,G)就可以通过 G 的所有边中是否存在一个端点属于 NEW，而另一个端点是 v 的边来判断。如果这个问题仍然是个比较复杂的问题，就还需要选择算法，也可能存在需要新抽象数据类型和数据结构。经过这种反复的精化过程，最后将算法中所有部分都细化为能用程序设计语言描述的成分，得到的就是我们希望的程序。

1.2 抽象数据类型

抽象数据类型的概念最早出现在 20 世纪 70 年代，它是面向对象方法的重要理论基础。但是本书不是讲解面向对象的教材，也不要求读者具有使用面向对象程序设计语言的基础。所以在内容的组织中仅仅使用了抽象数据类型的概念，而没有严格采用面向对象的程序设计语言的描述机制(例如 class)。

本节介绍的基本概念，对于读者把抽象数据类型的观点用于算法与数据结构的学习，理解抽象数据类型与数据结构的关系、算法与数据结构的关系会有所帮助。

1.2.1 什么是抽象数据类型

类型

一般而言，**类型**(type)是一组值(或者对象)的集合。例如，布尔作为一种类型是由真(true)和假(false)两个值组成的集合；布尔向量也可以作为一种类型，它的每个值是一个由 true 和 false 构成的向量。不同的值构成的类型也不同，例如，type1 = {1,2,3,4,...}和 type2 = {-1,-2,-3,-4,...}是两个不同的类型。

数据类型

数据类型(data type)通常是指在计算机(语言)中可以使用的一个类型，它不但包括这个类型的值的集合，还包括定义在这个类型上的一组操作。例如，整数作