普通高等教育“十一五”国家级规划教材配套参考书

高等学校计算机科学与技术专业系列教材

离散数学实验与习题解析

傅　彦　王丽杰　尚明生　顾小丰

高等教育出版社
Higher Education Press

普通高等教育“十一五”国家级规划教材配套参考书
高等学校计算机科学与技术专业系列教材

离散数学实验与习题解析

傅　彦　王丽杰　尚明生　顾小丰

高等教育出版社

内容提要

本书是国家精品课程“离散数学”主讲教材《离散数学及其应用》的配套实验与习题指导书。本书根据离散数学课程教学的基本要求，为计算机以及相关专业的本、专科学生更好地完成离散数学课程的课后练习和应用实践而编写。全书分为两大部分，第一部分是离散数学应用及实验，帮助学生进行课程实践，培养对离散数学课程的兴趣和动手能力。第二部分为习题及其解答。

本书可作为高等学校计算机及相关专业离散数学课程学习指导及实验用书，也可供对离散数学感兴趣的人参考使用。

图书在版编目(CIP)数据

离散数学实验与习题解析/傅彦等 .—北京:高等教育出版社,2007.11

ISBN 978-7-04-022469-6

Ⅰ.离… Ⅱ.傅… Ⅲ.离散数学-高等学校-教学参考资料 Ⅳ.O158

中国版本图书馆 CIP 数据核字(2007)第 155689 号

策划编辑 刘 燕 **责任编辑** 崔梅萍 **责任设计** 于文燕 **责任绘图** 杜晓丹
版式设计 王 莹 **责任校对** 朱惠芳 **责任印制** 朱学忠

出版发行 高等教育出版社
社 址 北京市西城区德外大街 4 号
邮政编码 100011
总 机 010-58581000

经 销 蓝色畅想图书发行有限公司
印 刷 北京明月印务有限责任公司

开 本 787×1092 1/16
印 张 14.5
字 数 320 000

购书热线 010-58581118
免费咨询 800-810-0598
网 址 http://www.hep.edu.cn
http://www.hep.com.cn
网上订购 http://www.landraco.com
http://www.landraco.com.cn
畅想教育 http://www.widedu.com

版 次 2007 年 11 月第 1 版
印 次 2007 年 11 月第 1 次印刷
定 价 18.50 元

物料号 22469-00

前　言

本书是根据教育部高等学校计算机科学与技术教学指导委员会编制的“计算机专业规范”的有关要求，为计算机以及相关专业的本、专科学生更好地完成离散数学课程的课后练习和应用实践编写的配套辅导教材。全书分为两篇，第一篇是离散数学应用及实验，帮助学生进行课程实践，培养对离散数学课程的兴趣和动手能力。第二篇为习题及其解答。

离散数学应用及实验部分，分为六个章节，前五个章节分别从集合论、数理逻辑、二元关系、图论和代数系统五个方面，先简要叙述了离散数学各部分内容的主要应用领域，然后描述了它们在计算机中的表示，最后给出了各部分内容的综合实验的例题和习题。第六章汇编了适用于离散数学课程的课程设计实验，这些实验都给出了必要的基础知识和实验指导，以帮助学生更好、更快地完成实验。

习题部分题量较大，供读者自我测试和学习。

本书的撰写任务由电子科技大学计算机学院离散数学课程组承担。第一篇由王丽杰老师编写，第二篇第7.1～7.8节由尚明生老师编写，第7.9～7.11，7.15节由顾小丰老师编写，第7.12～7.14节由傅彦老师编写。

编写过程中，作者参考了国内外多种版本的离散数学教材以及实验软件相关的书籍和网站资料，从中受益匪浅，在此一并向有关作者致谢。

由于作者水平有限，书中难免出现疏漏和错误，敬请读者批评指正。

编　者

2007年9月于电子科技大学

目　录

第一篇　离散数学应用及实验

第二篇 离散数学习题解析

第一篇

离散数学应用及实验

第1章　集合论的应用和实验

1.1　集合论在计算机科学中的应用

19世纪末德国伟大的数学家康托尔(Georg Cantor)创立了集合论.集合的概念大大扩充了数学的研究领域,给数学结构提供了坚实的基础.因而,集合论是现代数学的基础.它有助于把复杂的事物整理起来,以静态的方式对其进行考察.下面从几个方面谈谈集合论在计算机科学中的作用.

首先,集合论本身构成了计算机科学的基础.这是由于数学是计算机学科的基础,而集合论又是数学的基础理论,因此集合论本身是计算机科学的基础之基础.

其次,集合论被应用在计算机科学研究的各个方面.集合是构造离散结构的基础,离散结构是计算机的基本结构.从集合构造而来的离散结构包括:计数时广泛使用的组合(即无序对象集)、表示对象之间相互关联的关系(即序偶集合)、图形(顶点集合以及连接顶点的边的集合)以及用户模拟计算机的有限状态机等.集合论在人工智能领域、逻辑学及程序设计语言等方面都有着重要的应用.

最后,集合论在新一代智能计算机的发展中具有重要的作用.1965年,美国控制论专家、数学家查德发表了论文《模糊集合》,标志着模糊数学这门学科的诞生.在模糊集合中,给定范围内元素对它的隶属关系不一定只有"是"或"否"两种情况,还存在中间过渡状态,可以用介于0和1之间的实数来表示隶属程度,比如"老人"是个模糊概念,70岁的肯定属于老人,它的隶属程度是1,40岁的人肯定不算老人,它的隶属程度为0,按照查德给出的公式,55岁属于"老"的程度为0.5,即"半老",60岁属于"老"的程度0.8.查德认为,指明各个元素的隶属集合,就等于指定了一个集合.当隶属于0和1之间值时,就是模糊集合.在计算机方面,最重要的关于模糊理论的研究就是计算机智能.利用模糊集合理论,把人类的语言和思维过程提炼成数学模型,使人类语言数量化、形式化,并通过对模糊逻辑、模糊控制、模糊识别、模糊聚类分析、模糊决策等方面的分析,使计算机能够模拟人脑的高级智能.目前,世界上发达国家正积极研究、试制智能化的模糊计算机,1986年日本山川烈博士首次试制成功模糊推理机,它的推理速度是1 000万次/秒.1988年,我国汪培庄教授指导的几位博士也研制成功一台模糊推理机——分立元件样机,它的推理速度为1 500万次/秒.这表明我国在突破模糊信息处理难关方面迈出了重要的一步.模糊理论当前还有相当大的发展空间,需要付出很多的努力.

1.2　集合的计算机表示

在计算机中表示一个集合可以有多种方式.在这里主要介绍三种常用的方法:数组法、链表法和位串法.

1.2.1　数组法

当集合中的每个元素占用相同的空间大小时,可以使用连续的空间来存放集合中的元素,这就是数组法.如图 1.2.1 给出了集合{abc,123,fff,9eg}的内存占用情况.

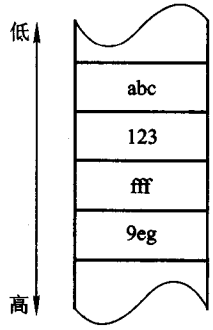

图 1.2.1

在 C 语言中,可使用下面的语句来定义这个集合.

```
char * strset[ ] = {"abc","123","fff","9eg"};
```

利用数组法表示集合,简单而又易于访问,但是不利于集合中元素的动态增删.例如,要删除图 1.2.1 中的元素"123",必须要先从头查找到"123"这个元素,然后将其后的"fff"和"9eg"前移.显然,进行的查找、比较与移动元素的次数会依赖于集合中的元素个数和要删除的元素的位置.并且,必须用集合可能的最大元素数来预分配存储空间.

1.2.2　链表法

当集合中的元素经常发生变化,并且各元素占用的空间大小有差异时,可采用链表的方式进行表示.如图 1.2.2 给出了集合{a,bc123,fff,9ggggggggggg}的链表示意图.

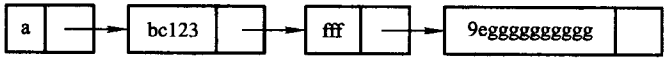

图 1.2.2

链表的好处是不需要连续的空间,并且添加和删除元素十分方便.下面给出了使用 C 语言定义的单链表结构.

```
struct setElement{
    char * element; //集合元素
    char * next;    //指向集合中下一个元素的指针
};
struct setElement * setTable = NULL; //链表头,初值为空
```

1.2.3　位串法

数组法和链表法的最大缺点就是集合的并、交和差运算变得很耗费时间,因为这些操作会需

要花费大量的时间进行查找.因此,位串法使用一种将全集中所有元素以任意次序存放的方式来保存集合中的元素.这会使得集合运算更简单一些.

假定全集 U 中的元素是有限的并且是合理的大小(这样才能够保证全集的大小不会超过计算机使用的内存大小).首先,为全集中的元素指定一个任意的次序,如 $a_1,a_2,\cdots,a_n$.对全集 U 的任意一个子集 A 来说,即可使用长度为 n 的二进制位串来表示它,规则是如果 a_i 是集合 A 中的元素,则这个二进制串的第 i 位为 1,否则为 0.

例 1.2.1　全集 $U=\{1,2,3,4,5,6,7,8,9,10\}$,全集元素按照递增的方式排序,那么 U 中所有奇数构成的子集,所有偶数构成的子集,不超过 5 的元素构成的子集分别是什么?

分析　全集中含有 10 个元素,因此子集需要用一个 10 位的位串来表示,所有奇数构成的子集即在 10 位位串的第 1,3,5,7,9 位为 1,其余位为 0;所有偶数的位串则是 2,4,6,8,10 位为 1,其余位为 0;不超过 5 的元素所组成的位串则为 1,2,3,4,5 位为 1,其余位为 0.

解　所有奇数构成的子集可表示为 $S_1=10101\ 01010$;

所有偶数构成的子集可表示为 $S_2=01010\ 10101$;

不超过 5 的元素构成的子集可表示为 $S_3=11111\ 00000$.

利用位串表示集合便于进行集合的运算.如果要计算两个集合的交集和并集,可以利用两个集合的位串进行按位操作来实现.

例 1.2.2　子集 $A=\{1,2,4\}$ 和 $B=\{2,3,4,5,7,8,9\}$ 可分别表示为 11010 00000 和 01111 01110,求出它们的交集和并集.

分析　根据集合的并运算规则,对两个集合并集 $A\cup B$ 的第 i 位而言,只要 A 和 B 的第 i 位中有一个 1 即为 1,因此对应于二进制位串的按位或操作.同理对交运算而言,则必须同时为 1 才能为 1,因此对应于二进制位串的按位与操作.

解　$A\cup B$ 可表示为

$$11010\ 00000 \vee 01111\ 01110 = 11111\ 01110,$$

它表示的子集是 $\{1,2,3,4,5,7,8,9\}$.

$A\cap B$ 可表示为

$$11010\ 00000 \wedge 01111\ 01110 = 01010\ 00000,$$

它表示的集合是 $\{2,4\}$.

同样可以利用位串的按位操作来求得集合的补集.

例 1.2.3　所有偶数构成的集合 A 表示为 01010 10101,求它的补集.

分析　根据补运算的规则,对集合 A 的补集的第 i 位而言,如果集合 A 的第 i 位为 1,则集合 A 的补集的第 i 位应该为 0,反之则为 1.因此,补运算可利用位串的按位取反操作来实现.

解　将集合 A 的位串按位取反,可得到集合 A 的补集对应的位串为 10101 01010,它对应的集合是 $\{1,3,5,7,9\}$.

需要补充说明的是,并不是所有语言都能提供位串的定义和操作方法.比如在 C 语言中,虽然提供了按位的操作,不过并未提供位串的定义方式,但仍可借助数组来定义.短的位串也可使

用基本数据类型(char,short,long)来定义.

上面介绍了利用位串法来求解集合的并、交和补运算.同理,也可以得到求解集合的差运算和对称差运算的方法.这些运算可归纳成一个表格,如表 1.2.1 所示.

表 1.2.1

集合的操作	方　法	C 语言表达式
并运算	按位或	$A\|B$
交运算	按位与	$A\&B$
补运算	按位取反	$\sim A$
差运算	第二操作数按位取反之后与第一操作数按位与	$A\&\sim B$
对称差运算	按位异或	$A^{\wedge}B$

1.3　集合论实验

1.3.1　实验目的及要求

(1) 掌握集合在计算机中的表示.

(2) 根据不同的表示法,可利用计算机程序实现集合的各类运算.

(3) 可根据实际情况的不同,灵活选用集合的三种表示法来解决实际问题.

1.3.2　实验内容及步骤

1.3.2.1　验证性实验

(1) 给出一个集合,分别用数组法、链表法和位串法表示出来.

(2) 写一个程序,验证集合的各类运算(并,交,补,差,对称差).

(3) 写一个程序,验证集合的包含关系.

1.3.2.2　综合实验

一、集合的确定性算法

1. 背景知识介绍

考虑一个问题:假定包含 64 个元素的全集为{0,1,2,3,…,63},A 为它的任意一个子集.设计一个算法,求出 A 中的最小元素.

这个问题看起来是不难的,一种可行的方案就是对集合 A 中的元素进行一次遍历,即可得到其中的最小元素.假设 A 中元素个数为 n,需要比较的次数为 $n-1$ 次.因此这种方案的问题

在于遍历所花费的时间具有不确定性，它会依赖于 A 中的元素个数．在某些特定领域中，希望算法的执行时间是确定的，即无论 A 中有多少个元素，算法执行效率不变．下面就讨论一个能够解决上面提出的问题的确定性算法．

在这里使用位串法来表示集合 A，因此需要用到 64 位的二进制位串（注意：此处采用的位串次序与前面介绍的有所不同．在实际使用中，为方便运算，会以计算机中的比特序来作为位串次序，即比特 0 为第一个元素）．在 C 语言中可以用如下的方法来定义这个位串．

```
char ATable[8];
```

每个 char 类型有 8 个二进制位，因此数组 ATable 就可表示出所有 64 个元素隶属子集 A 的情况．下面就分步骤来介绍这个算法．

（1）为辅助进行判定，需定义一个变量 AGroup 如下

```
char AGroup;
```

AGroup 与 ATable 的关系如图 1.3.1．

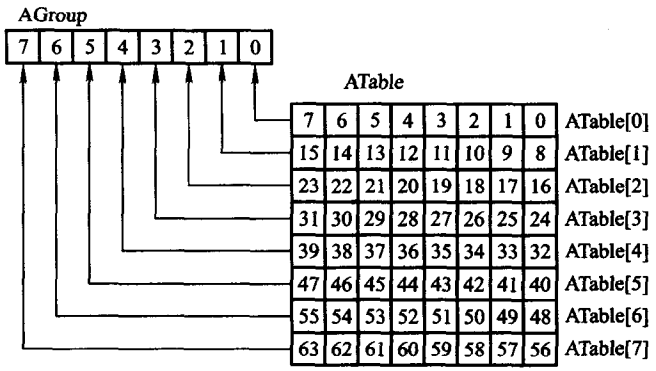

图 1.3.1

ATable 的每个数组元素对应 64 个元素中的 8 个．如 ATable[0]对应元素 0～7（从低位开始，以后类同），ATable[5]对应元素 40～47．比如 ATable[5]的值为 0x12 则表示元素 41 和 44 属于集合 A．

AGroup 中的每个二进制位与 ATable 中的每个数组元素对应．即 AGroup 的第 i 位（i 值从 0 到 7）对应 ATable[i]．ATable[i]为 0 当且仅当 AGroup 的第 i 位为 0；ATable[i]不为 0 当且仅当 AGroup 的第 i 位为 1．也就是说 AGroup 的第 i 位表示了 ATable[i]中的 8 个元素是否至少有一个元素属于集合 A．

（2）集合元素与 AGroup 和 ATable 的关系

集合元素与 AGroup 和 ATable 的关系如图 1.3.2 所示．将全集中的元素写成二进制之后，发现所有元素都可以使用 6 个二进制位来表示出来，其中高三位为该元素在 AGroup 的位置（假设为第 i 位），低三位为该元素在 ATable[i]中的位置．如元素 35（100011b），高三位 100，表示 AGroup的第 4 位为 1，对应 ATable[4]，低三位为 011，则对应 ATable[4]的第 3 位，即元素 35．

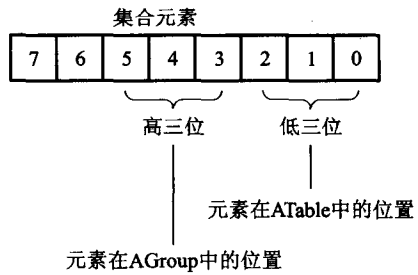

图 1.3.2

(3) 设置最小元素判定表

```
char ADecisionTable[256] = {
    0,0,1,0,2,0,1,0,3,0,1,0,2,0,1,0,    /* 0x00-0x0F */
    4,0,1,0,2,0,1,0,3,0,1,0,2,0,1,0,    /* 0x10-0x1F */
    5,0,1,0,2,0,1,0,3,0,1,0,2,0,1,0,    /* 0x20-0x2F */
    4,0,1,0,2,0,1,0,3,0,1,0,2,0,1,0,    /* 0x30-0x3F */
    6,0,1,0,2,0,1,0,3,0,1,0,2,0,1,0,    /* 0x40-0x4F */
    4,0,1,0,2,0,1,0,3,0,1,0,2,0,1,0,    /* 0x50-0x5F */
    5,0,1,0,2,0,1,0,3,0,1,0,2,0,1,0,    /* 0x60-0x6F */
    4,0,1,0,2,0,1,0,3,0,1,0,2,0,1,0,    /* 0x70-0x7F */
    7,0,1,0,2,0,1,0,3,0,1,0,2,0,1,0,    /* 0x80-0x8F */
    4,0,1,0,2,0,1,0,3,0,1,0,2,0,1,0,    /* 0x90-0x9F */
    5,0,1,0,2,0,1,0,3,0,1,0,2,0,1,0,    /* 0xA0-0xAF */
    4,0,1,0,2,0,1,0,3,0,1,0,2,0,1,0,    /* 0xB0-0xBF */
    6,0,1,0,2,0,1,0,3,0,1,0,2,0,1,0,    /* 0xC0-0xCF */
    4,0,1,0,2,0,1,0,3,0,1,0,2,0,1,0,    /* 0xD0-0xDF */
    5,0,1,0,2,0,1,0,3,0,1,0,2,0,1,0,    /* 0xE0-0xEF */
    4,0,1,0,2,0,1,0,3,0,1,0,2,0,1,0,    /* 0xF0-0xFF */
};
```

判定表以 AGroup 和 ATable 数组元素的值为索引，获取该值对应二进制表示中 1 出现的最低二进制位的序号(0～7). 因为 AGroup 和 ATable 数组元素都是 8 位，所以它们的值的范围为 0～0xFF. 比如值 0x12 的二进制表示为 1 的最低位是第 1 位，因此对应的判定表的值为 1.

当 AGroup 和 ATable 以及 ADecisionTable 的判定表建立之后，就可以得到获取任意集合 A 中最小元素的算法了.

```
high3bit = ADecisionTable[AGroup];
low3bit = ADecisionTable[ATable[high3bit]];
```

```
smallestElement = (high3bit<<3) + low3bit;
```

可以看到,这个算法并不依赖于 A 中的元素数目,是个具有确定性的算法.

进一步考虑当集合 A 中的元素发生变化的情况.当集合 A 中添加或者删除一个元素时,就需要考虑如何修改对应的 AGroup 和 ATable.定义集合元素到 AGroup 的映射表

```
char AMapTable[8];
```

AMapTable 的内容如表 1.3.1 所示.

表 1.3.1

下　标	二进制值	下　标	二进制值
0	00000001	4	00010000
1	00000010	5	00100000
2	00000100	6	01000000
3	00001000	7	10000000

① 当添加一个元素 addElement 时

```
AGroup |= AMapTable[addElement>>3];
    ATable[addElement]>>3] |= AMapTable[addElement * 0x07];
```

② 当删除一个元素 delElement 时

```
if((ATable[delElement]>>3] &= ~AMapTable[delElement * 0x07])==0)
    AGroup &= ~AMapTable[delElement>>3];
```

2. 实验内容

在支持多任务的嵌入式实时操作系统中,各任务根据重要程度,会被赋予一个优先级.优先级最高的任务能够最先获得运行.假设任务最多具有 64 个优先级(0～63,值越小优先级越高),并且每个任务的优先级不同.试设计一个基于优先级的确定性调度算法.要求:

(1) 针对输入的优先级序列,能够选出具有最高优先级的任务.如输入 56,8,22,2,14,即可得到输出 2.

(2) 能够处理任务的动态添加和删除的情况.

提示:64 个优先级可看作具有 64 个元素的全集,而每一时刻具有不同优先级的任务则构成全集的一个子集 A,然后便可使用前面介绍的算法来完成了.

备注:这个算法在嵌入式操作系统中广泛使用,称为优先级位图算法.

3. 实验思考题

a. 本实验的算法确定性是如何得到保证的?

b. 若任务有 256 个优先级,那么算法应该做什么样的改动?

二、线性数据结构

1. 背景知识介绍

线性数据结构在计算机科学中使用相当广泛,这是由于线性数据结构的构成简单,操作起来

也很方便.常用的线性数据结构包括线性表、栈和队列.

线性表是最简单的一种数据结构.它是由 n 个数据元素构成的有限序列.一个线性表可记为

$$(a_1,a_2,a_3,\ldots,a_{i-1},a_i,a_{i+1},\cdots,a_n).$$

对于线性表中的元素,不仅可以进行访问,还可以进行添加和删除.

线性表的基本操作包括:

① 初始化一个线性表.

② 销毁一个线性表.

③ 判断线性表是否为空.

④ 获得线性表长度.

⑤ 在线性表中查找指定元素.

⑥ 插入一个元素到线性表指定位置.

⑦ 从线性表中删除指定元素.

对于线性表中元素大小相同并且表的长度在一个合理的范围之内(这和计算机的存储空间有关),通常可以使用顺序表示法(即数组法)表示.因为数组很容易访问.但对于那种需要经常性添加和删除元素的应用,就需要使用链表法进行存储,因为顺序表示法需要大量移动元素.

对线性表的操作稍作限制,就可以得到另外两种线性数据结构:栈和队列.

栈是仅能在表的一端进行元素的添加和删除的一种线性表.栈的原理可用一个生活中的常识做比喻:在一个杯子中放入比杯子直径稍小的苹果,不管是放入或者拿出,每次都只能拿一个苹果.最后放入的苹果是第一个可以拿出来的,最先放入的苹果最后才能拿出来,这种特性叫做后进先出 LIFO(或先进后出 FILO).符合这种情况的数据结构就称为栈.

可进行添加和删除操作的一端称为栈顶,另外一端称为栈底.

栈的基本操作包括:

① 初始化一个栈.

② 销毁一个栈.

③ 判断栈是否为空.

④ 获取栈的长度.

⑤ 出栈(删除栈顶元素).

⑥ 入栈(在栈顶插入一个元素,使其成为新的栈顶).

和线性表类似,栈也有顺序和链表两种存储表示方法.

栈的应用非常多,最重要的就是栈可用于实现函数调用和递归.在程序设计中,经常出现一个函数 a 需要调用另一个函数 b 的情况(若 b 和 a 是同一个函数,这就是递归了).栈的方式可以很方便地保存调用函数 a 的运行状态,并且恢复起来也很容易.

和栈不同,队列是一种在表的一端插入,而另一端进行删除的线性表.它的这种特性类似于我们生活中的排队,新来的人排入队尾,而最早进入队列的最早离开,这种特性也叫做先进先出 FIFO.

队列的基本操作包括：

① 初始化一个队列.

② 销毁一个队列.

③ 判断队列是否为空.

④ 获得队列长度.

⑤ 入队(在队尾插入新的元素,成为新的队尾).

⑥ 出队(删除队头元素,下一个元素成为新的队头).

队列通常使用链表表示法.但有一种特殊的队列,称作循环队列,也会采用顺序表示法.这里的循环是指循环利用队列的存储空间,循环队列的结构类似一个环,它的队头和队尾元素的位置并不固定.当队列满时,队头和队尾元素刚好相邻.

队列结构在计算机中也是经常使用的.比如操作系统处理多个作业时,就可以采用队列的方式,先进入队列的作业会先得到处理.

2. 实验内容

a. 利用顺序表示法或链表方式实现线性表的基本操作.

b. 利用顺序表示法实现栈的基本操作.

c. 利用循环队列的方式实现队列的基本操作.

3. 实验思考题

顺序表示和链表方式的区别是什么？如何选用？