TURING 图灵计算机科学丛书

CAMBRIDGE

数据结构与算法

C#语言描述

Data Structures and Algorithms Using C#

[美] Michael McMillan 著

吕秀锋 崔 睿 译

人民邮电出版社

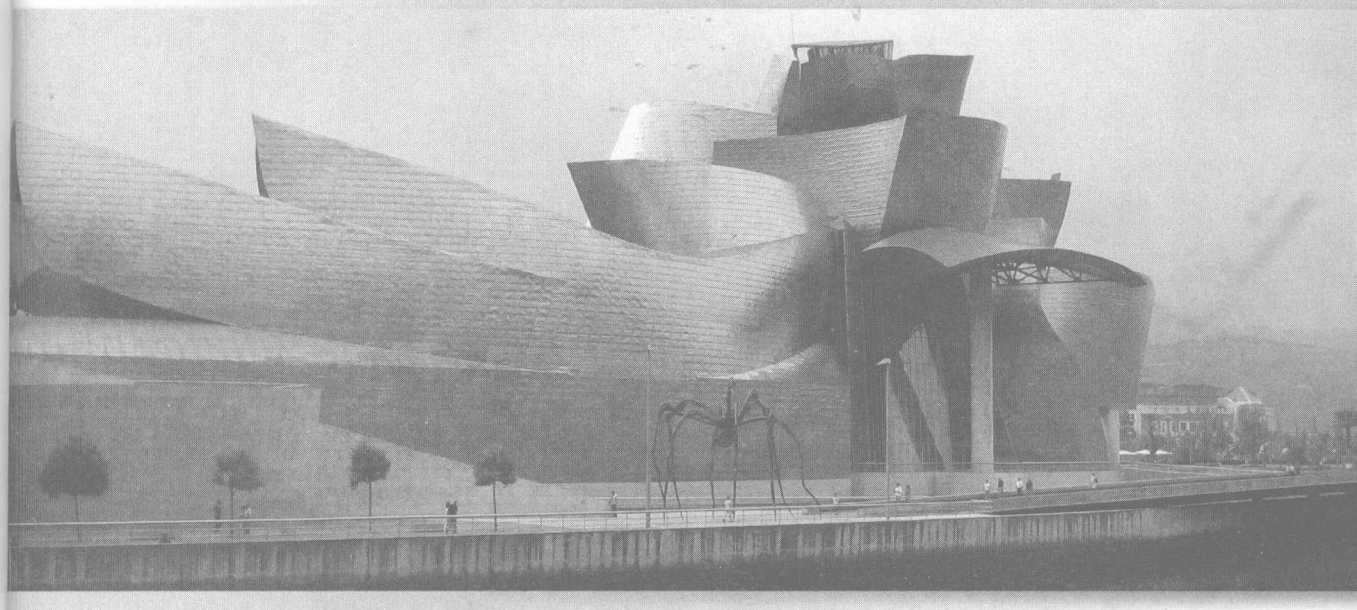
POSTS & TELECOM PRESS

TURING 图灵计算机科学丛书

数据结构与算法
C#语言描述

Data Structures and Algorithms Using C#

[美] Michael McMillan 著
吕秀锋 崔睿 译



人民邮电出版社
北京

图书在版编目（CIP）数据

数据结构与算法：C#语言描述/（美）麦克米伦（McMillan, M.）著；吕秀锋，崔睿译. —北京：人民邮电出版社，2009.5

（图灵计算机科学丛书）

书名原文：Data Structures and Algorithms Using C#

ISBN 978-7-115-20427-1

I. 数… II. ①麦… ②吕… ③崔… III. ①数据结构 ②算法分析 ③C语言－程序设计 IV. TP311.12 TP301.6 TP312

中国版本图书馆CIP数据核字（2009）第026780号

内 容 提 要

本书是在.NET框架下用C#语言实现数据结构和算法的第一本全面的参考书。本书采用了更加实用的时间测试方法代替常规的大O表示法来分析算法性能。内容除涵盖了数组、广义表、链表、散列表、树、图、排序、搜索等常规数据结构和算法外，还讨论了概率和动态规划等方面的高级算法。

本书适合作为C#数据结构课程的教材，同时也适合C#专业人士阅读。

图灵计算机科学丛书

数据结构与算法：C#语言描述

◆ 著 [美] Michael McMillan

译 吕秀锋 崔 睿

责任编辑 杨海玲

执行编辑 罗 婧

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号

邮编 100061 电子函件 315@ptpress.com.cn

网址 http://www.ptpress.com.cn

北京艺辉印刷有限公司印刷

◆ 开本：787×1092 1/16

印张：17

字数：457千字 2009年5月第1版

印数：1-3 000册 2009年5月北京第1次印刷

著作权合同登记号 图字：01-2008-4085号

ISBN 978-7-115-20427-1/TP

定价：49.00元

读者服务热线：（010）88593802 印装质量热线：（010）67129223

反盗版热线：（010）67171154

版权声明

译 者 序

数据结构及算法是计算机科学技术领域重要的基础知识。目前国内介绍数据结构的书籍众多，描述数据结构及算法的语言也种类繁多，从早年的Pascal语言，到至今仍然流行的C语言，再到近几年出现的C++语言、Java语言等。而本书的作者独辟蹊径，在.NET框架下采用C#语言来描述数据结构中的基本原理及所有重要的算法，为本书平添了一抹独特的亮色。此外，本书在介绍算法的过程中引入了大量的代码实现，并采用了时间测试的方法进行算法分析，更加凸显了本书的实用性。

译者在翻译过程中对原书代码进行了测试运行，发现了一些问题，并进行了相应的修正。如有不当之处，恳请读者指正。

全书的翻译工作主要由吕秀锋负责，崔睿对本书代码进行了测试并做了大量的修正。在本书的翻译过程中，得到了许多人的帮助。首先要感谢人民邮电出版社图灵公司的杨海玲编辑在翻译过程中提供的各种建议和帮助，同时还要感谢罗婧编辑在编辑过程中所做的大量细致入微的工作。

受译者的水平所限，本书翻译中的疏漏或不当之处在所难免，敬请广大读者及同行批评指正。

译 者

2009年2月

前　言

在专业程序员的成长过程中，对于数据结构与算法的学习至关重要。虽然有许许多多关于数据结构与算法的书籍，但是这些书籍通常都是大学教材，并且是使用在大学里讲授的Java语言或C++语言编写的。C#语言正在成为一种广受欢迎的编程语言，这本书旨在面向C#程序员介绍数据结构与算法的基础知识。

C#语言根植在一个功能非常丰富的.NET框架开发环境中。在.NET框架类库中包含有一套数据结构类（也称为群集类），这套类的范围广泛，从Array类、ArrayList类和Collection类到Stack类和Queue类，再到HashTable类和SortedList类。学习数据结构与算法的学生在学习如何实现它们之前可以先看看如何使用数据结构。以前，老师在构建完整的栈数据结构之前只能抽象地讲解栈的概念，而现在老师可以通过向学生们展示如何用栈执行一些计算（如数制之间的转换），立即展示数据结构的实用工具。有了这些知识后，学生可以在课后学习数据结构（或算法）的基本原理，甚至可以构造他们自己的实现。

本书主要提供了对数据结构与算法的实用概述，这是所有计算机程序员们需要知道和了解的。基于这个原因，本书没有涵盖数据结构与算法的正规分析。因此，本书没有一个数学公式，也一次都没有提及大O分析（如果你不知道大O分析的含义，查看参考文献中提到的任何一本书都可以）。相反，本书把各种数据结构与算法作为求解问题的工具。书中讨论的数据结构与算法都用简单的时间测试进行了性能比较。

前提条件

阅读本书的唯一前提条件就是需要读者对C#语言有大概的了解，尤其是要了解一些用C#进行面向对象编程的知识。

章节组织

第1章向读者介绍数据结构作为数据群集的概念。介绍了线性和非线性群集的概念。示范说明了Collection类。这章还介绍了泛型编程的概念，它允许程序员编写一个类或一种方法，然后用于众多数据类型。泛型编程是C#语言一种重要的新特性（在C# 2.0及更高版本中可用）。这种特性是如此重要以至于在System.Collections.Generic命名空间中存在一个专门的泛型数据结构库。当数据结构具有在此库中能找到的泛型实现时，我们就会讨论它的用途。这章结尾处介绍了衡量书中讨论的数据结构与算法性能的方法。

第2章回顾了构造数组的方法，并且用示例说明了Array类的特性。Array类把许多与数组相关的函数（UBound函数、LBound函数等）封装到单独一个包中。ArrayList是数组的一种特殊类型，它支持动态调整大小。

第3章介绍了基础排序算法，如冒泡排序和插入排序；而第4章则研究了用于内存查找的最基本的算法，即顺序查找和二叉查找。

第5章探讨了两种经典的数据结构：栈和队列。这章的重点是这些数据结构在解决日常

数据处理问题中的实际应用。第6章介绍了BitArray类，这种类可用来有效地表示大量整型数值，比如测试成绩。

数据结构的书中通常不介绍字符串，但是本书第7章介绍了字符串、String类和StringBuilder类。这是因为在C#语言中许多的数据处理是在字符串上执行的，读者应该了解基于这两种类的特殊方法。第8章分析了用于文本处理和模式匹配的正则表达式的使用。与较传统的字符串函数和方法相比，正则表达式常常会提供更强大且更有效的处理。

第9章向读者介绍把字典作为数据结构来使用。字典和基于字典的不同数据结构把数据作为键值对来存储。这章向读者说明了如何创建基于DictionaryBase这个抽象类的自定义类。第10章介绍了散列表和HashTable类。HashTable类是字典的一种特殊类型，它在内部用散列算法存储数据。

链表作为另外一种经典的数据结构是在第11章介绍的。C#语言中的链表并不像在C++这样的基于指针的语言中的链表那样重要，但是它仍然在C#编程中发挥着作用。第12章为读者介绍了另一种经典数据结构——二叉树。二叉查找树作为二叉树的特殊类型将是这章的主要内容，其他二叉树类型在第15章进行介绍。

第13章向读者说明在集合中存储数据的方法。这种方法在数据结构只存储具有唯一性的数据值的情况下是很实用的。第14章涵盖了更多高级排序算法，包括流行且高效的快速排序算法QuickSort，此算法是大多数在.NET框架库中实现的排序过程的基础。第15章会看到3种数据结构：AVL树、红黑树和跳跃表。在无法使用二叉查找树的时候，这3种数据结构对查找是很有用的。

第16章讨论了图以及图的算法。图在表示许多不同的数据类型时非常有用，特别是网络。最后，第17章向读者介绍算法设计技巧的精髓：动态算法和贪心算法。

致谢

在这里我要感谢帮助我完成本书的各界人士。首先，我要感谢听我讲授数据结构与算法开发课程的学生们，他们是（排名不分先后）：Matt Hoffman、Ken Chen、 Ken Cates、Jeff Richmond和Gordon Caffey。此外，要感谢我在Pulaski技术学院（Pulaski Technical College）的同事Clayton Ruff，他多次旁听我的课程并且提出了极好的建议和批评。我还要感谢院长David Durr和系主任Bernica Tackett支持我的写作工作。同样，我还要感谢我的家人在我全身心投入研究和写作时对我的宽容与支持。最后，我要感谢剑桥大学出版社的编辑Lauren Cowles和Heather Bergman，感谢他们容忍我的许多问题、内容更改，以及习惯成自然地交稿延迟。

目　录

第1章 Collections类、泛型类和Timing类概述

本书采用C#语言来讨论数据结构与算法的开发和实现。书中用到的数据结构都可以在.NET框架类库System.Collections中找到。本章会逐步展开群集的概念，首先讨论自身特有的Collection类的实现（采用数组作为我们实现的基础），接着会介绍.NET框架中Collection类的内容。

泛型是C#语言2.0版新增加的一个重要补充，它允许C#程序员可以独立地或者在一个类中编写函数的某一个版本，而且不需要为了不同的数据类型而多次重载此函数。C#语言2.0版还为几种System.Collections数据结构实现泛型提供了一个专门的库System.Collections.Generic。本章将向读者介绍泛型编程。

本章最后会介绍一种用户定制的类——Timing类，后续的几个章节将会用它来衡量数据结构与/或算法的性能。这个类将取代大O分析法，这不是因为大O分析法不重要，而是因为本书采取了一种更为实用的方法来分析数据结构与算法。

1.1 群集的定义

群集（Collection）是一种结构化的数据类型，它存储数据并且提供数据向/从群集中添加、删除和更新操作，以及对群集的不同属性值的设置与返回操作。

群集分为线性群集和非线性群集。线性群集是一张元素列表，表中的元素顺次相连。线性群集中的元素通常由位置来决定次序（例如，第一个元素、第二个元素、第三个元素，依此类推）。在现实世界中，购物清单就是很好的线性群集实例，而在计算机中（当然这也是真实世界）则把数组设计成线性群集。

非线性群集所包含的元素在群集内没有位置次序之分，组织结构图就像用架子垒好的台球一样，是一个非线性群集的实例。而在计算机中，树、堆、图和集都是非线性群集。

无论是线性群集还是非线性群集都拥有一套定义好的属性和操作的集合。其中，属性用来描述群集，而操作就是群集上所能执行的内容。群集Count就是群集属性的一个实例，它保存着群集中数据项的数量。这里把群集的操作称为方法，它包括Add（向群集添加新元素），Insert（在群集指定的索引位置添加新元素）、Remove（从群集中移除指定元素）、Clear（从群集中移除所有元素）、Contains（确定指定元素是否是群集的成员），以及IndexOf（确定指定元素在群集中的索引位置）。

1.2 群集的描述

在上述两类群集中还有几个子类。线性群集可能是直接存取群集，也可能是顺序存取群集。而非线性群集既可以是层次群集，也可以是组群集。本小节就来讨论这些群集。

1.2.1　直接存取群集

直接存取群集最常见的实例就是数组。这里把数组定义为具有相同数据类型的元素的群
2 集，而且所有数组元素如同图1-1说明的那样，可以通过整数型索引直接进行存取访问。

图1-1　数组

数组可以是静态的，这样当声明数组的时候可以在程序范围内固定元素的数量。数组也可以是动态的，通过ReDim或者ReDim Preserve语句就可以增加数组元素的数量。

在C#语言中，数组不只是内置的数据类型，它还是一种类。在本章后续部分详细分析数组使用的时候，将会讨论如何把数组作为类对象来使用。

我们可以用数组来存储一个线性的群集。向数组添加新元素是很容易的，只要把新元素放置在数组尾部第一个空位上就可以了。但是，在数组中插入一个元素就不是这么容易（或高效）的了，因为要给插入的元素空出位置需要按顺序向后移动数组元素。从数组的尾部删除一个元素也是很容易的，只要移除掉最后一个元素的值就可以了。但是，删除数组中任何其他位置上的元素就没有这么容易了，就像处理插入操作一样，为了保持数组中元素的连续性，可能需要先调整许多数组元素的位置。这些情况将在本章后续内容中进行讨论。.NET框架为简化线性群集的编程提供了一种专门的数组类——ArrayList，第3章将会对此类进行分析研究。

字符串是直接存取群集的另外一种类型，它是字符的群集。和存取数组元素的方式一样，它也可以基于字符的索引对其进行存取。在C#语言中，字符串也是作为类对象来实现的。这个类包含一个在字符串上执行标准操作的庞大的方法集合，例如串连接、返回子串、插入字符、移除字符，等等。第8章会讨论String类。

C#字符串是不可变的，这意味着一旦对字符串进行了初始化，就不能再改变它了。当要修改字符串的时候，不是改变原始的字符串，而是创建一个字符串的副本。在某些情况下这种行为可能会导致性能下降，所以.NET框架提供了能让用户处理可变字符串的StringBuilder类。第8章也会对StringBuilder进行介绍。

最后一种直接存取的群集类型是结构（在一些编程语言中也称为记录）。它是一种复合
3 数据类型，其所包含的数据可能拥有许多不同的数据类型。例如，一名雇员记录就是由雇员的姓名（字符串）、薪水（整数）、工号（字符串或整数）以及其他属性组成的。由于把这些数据的值分别存储在分散的变量内是很容易产生混淆的，所以编程语言采用结构来存储此类数据。

C#语言的结构所增加的强大功能就是为执行存储在数据上的操作定义了方法。尽管不能从结构继承或推导出一种新的类型，但是这种做法使得结构在某些地方很像一个类。下面的代码说明了C#语言中结构的一个简单应用。

```
using System;
public struct Name
{
    private string fname, mname, lname;
    public Name(string first, string middle, string last)
```

```
{
    fname = first;
    mname = middle;
    lname = last;
}
public string firstName
{
    get
    {
        return fname;
    }
    set
    {
        fname = firstName;
    }
}
public string middleName
{
    get
    {
        return mname;
    }
    set
    {
        mname = middleName;
    }
}
public string lastName
{
    get
    {
        return lname;
    }
    set
    {
        lname = lastName;
    }
}
public override string ToString()
{
    return (String.Format("{0} {1} {2}", fname, mname,lname));
}
public string Initials()
{
    return (String.Format("{0}{1}{2}", fname.Substring(0, 1),
            mname. Substring (0, 1), lname.Substring(0, 1)));
}
```

```
}
public class NameTest
{
    static void Main()
    {
        Name myName = new Name("Michael", "Mason", "McMillan");
        string fullName, inits;
        fullName = myName.ToString();
        inits = myName.Initials();
        Console.WriteLine("My name is {0}.", fullName);
        Console.WriteLine("My initials are {0}.", inits);
    }
}
```

虽然.NET环境中许多元素都是用类来实现的（如数组和字符串），但是语言的几个主要元素还是用结构来实现的，比如数字数据类型。例如，整数数据类型就是用Int32结构来实现的。采用Int32的方法之一就是把用字符串表示的数转换成为整数的Parse方法。例如：

```
using System;
public class IntStruct
{
    static void Main()
    {
        int num;
        string snum;
        Console.Write("Enter a number: ");
        snum = Console.ReadLine();
        num = Int32.Parse(snum);
        Console.WriteLine(num);
    }
}
```


1.2.2　顺序存取群集

顺序存取群集是把群集元素按顺序存储的表，这里也把此类群集称为线性表。线性表在创建时没有大小限制，这就意味着它们可以动态地扩展和收缩。不能对线性表中的数据项进行直接存取访问，而要像图1-2表示的那样通过数据项的位置对其进行存取。线性表的第一个元素在表头的位置，而最后一个元素在表尾的位置。

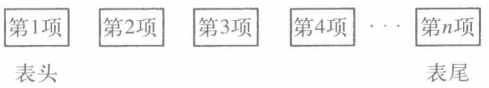


图1-2　线性表

由于不能直接存取线性表的元素，所以为了访问某个元素就需要遍历线性表直到到达要找元素的位置为止。线性表的实现通常允许两种遍历表的方法：一种是单向从前往后遍历，而另一种则是双向遍历，即从前向后和从后向前遍历。

线性表的一个简单实例就是购物清单。顺次写下要购买的全部商品就会创建一张购物清单。在购物时一旦找到某种商品就把它从清单中划掉。

线性表既可以是有序的，也可以是无序的。有序线性表具有顺次对应的有序值。如下列人名所表示的情况：

Beata Bernica David Frank Jennifer Mike Raymond Terrill

而无序线性表则是由无序元素组成的。在第2章对二叉查找算法与简单线性查找算法进行讨论时，就会发现线性表的顺序会在查找表中数据时产生很大的差异。

6

线性表的某些类型限制访问数据元素。这类线性表有栈和队列。栈是一种只允许在表头（或顶端）存取数据的表，在表的顶端放入数据项，而且也只能从表的顶端移出数据项。正是基于这种原因，栈也被称为后进先出结构。这里把向栈添加数据项的操作称为入栈（push），而把从栈移出数据项的操作称为出栈（pop）。图1-3展示了栈的这两种操作。

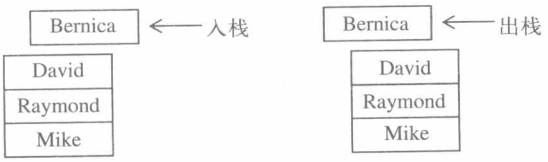


图1-3 栈操作

栈是一种非常常见的数据结构，特别是在计算机系统编程中尤为普遍。在栈的众多应用中，它常用于算术表达式的计算和平衡符号。

队列是一种在表尾添加数据项并在表头移除数据项的表，它被称为是先进先出结构。这里把向队列添加数据项称为入队，而把从队列移出数据项称为出队。图1-4展示了队列的这两种操作。

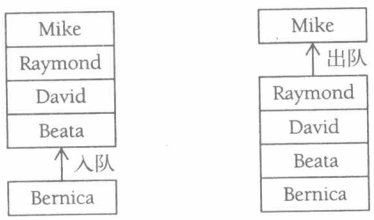


图1-4 队列操作

队列既可用于调度操作系统任务的系统编程，也可用于模拟研究的编程。在每一种可能的少量情况下，队列都可以为模拟等待队列产生极好的结构。优先队列是队列的一种特殊类型，它允许最先移出队列的数据项具有最高的优先级。例如，优先队列可以用来研究医院急诊室的操作，这里应该先对心脏病突发患者进行救护，然后再处理手臂骨折患者。

最后要讨论的一类线性群集被称为通用的索引群集。这类群集的第一种就是散列表，它存储了一组与键相关联的数据值。在散列表中有一个被称为散列函数的特殊函数，它可以取走一个数据值，并且把此数据值（称为键）转换成用来检索数据的整数索引。然后此索引会用来存取访问与键相关联的数据记录。例如，一条雇员记录可能由雇员姓名、薪水、工作年限以及所工作的部门组成，结构如图1-5所示。此数据记录的键就是雇员的姓名。C#语言有一个称为HashTable的类用来存储散列表的数据，第10章会讨论此结构。

7

“Paul E. Spencer”
37500
5
“信息系统”部门

图1-5　散列的记录

另外一种通用的索引群集就是字典，也称为联合，它是由一系列键值对构成的。此结构与词典类似，词典中的词是键，而词的定义则是与键相关联的值。键就是与其相关联的值内的索引。虽然索引不一定是整数，但是由于上述这种索引方案，还是常把字典称为联合数组。第11章会对.NET框架内容的几种Dictionary类进行讨论。

1.2.3　层次群集

非线性群集主要分为两大类：层次群集和组群集。层次群集是一组划分了层次的数据项集合，位于某一层的数据项可能会有位于下一较低层上的后继数据项。

树是一种常见的层次群集，它看上去像是一棵倒置树，其中一个数据项作为根，而其他数据值则作为叶子挂在根的下面。树的元素被称为节点，在特定节点下面的元素被称为是子节点。图1-6展示了一棵树的实例。

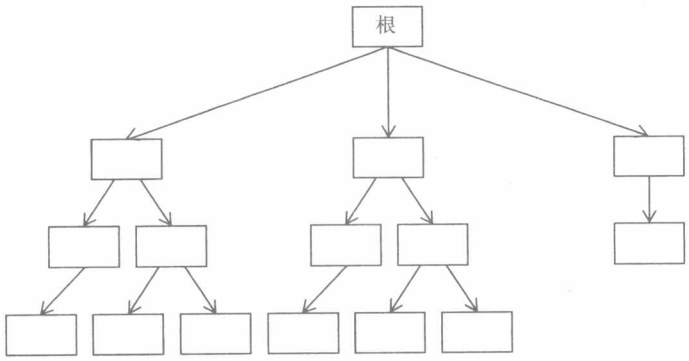


图1-6　树群集

树在几种不同的领域都有应用。大多数现代操作系统的文件系统都是采用树型群集设计而成的，其中一个目录作为根，而其他子目录则作为根的孩子。

二叉树是树群集的一种特殊类型，树中每个节点最多只有两个孩子。二叉树可以变成二叉查找树，这样做可以极大地提高查找大量数据的效率。实现的方法是依据从根到要存储数据的节点的路径为最短路径的方式来放置节点。

还有一种树类型就是堆，这样组织就是为了便于把最小数据值始终放置在根节点上。在删除时会移除根节点。此外，堆的插入操作和删除操作总是会导致堆的重组，只有这样才能把最小值放在根节点上。我们经常会用堆来排序，这被称为是堆排序。通过反复删除根节点以及重组堆的方式就可以对存储在堆内的数据元素进行排序。

第12章将对几种不同类型的树进行讨论。

1.2.4 组群集

无序的数据项组成的非线性群集被称为组。集合、图和网络是组群集的三种主要类型。

集合是一种无序数据值的群集，并且集合中每一个数据值都是唯一的。当然，就像整数一样，一个班级的学生的列表就是一个集合的实例。在集合上执行的操作包括联合和交叉。图1-7显示了集合操作的实例。 9

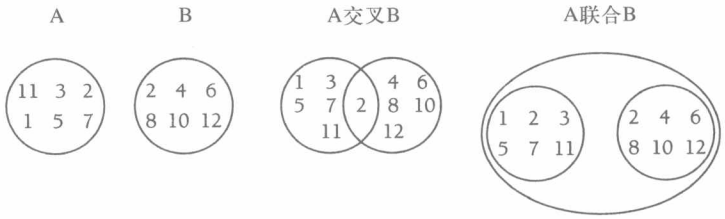


图1-7 集合群集操作

图是由节点集合以及与节点相连的边集合组成的。图用来对必须访问图中每个节点的情况进行建模，而且有些时候还要按照特定顺序进行访问，这样做的目的是为了找到“遍历”图的最有效的方法。图在物流管理和作业调度方面广泛应用，研究它的人既有计算机专业人士也有数学工作者。大家可能听说过“旅行商”问题，这就是一类特殊的图问题。它要求在旅行预算允许的条件下为商人确定最有效的完整旅行路线，使其走遍要去的所有城市。此问题的一个例子如图1-8。

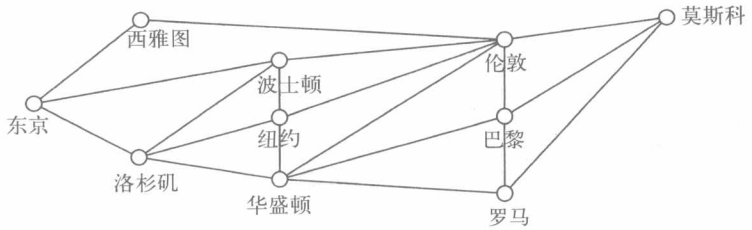


图1-8 旅行商问题

此问题是所谓NP-完全问题的一部分内容，这就意味着大量此类型的问题是无法知道确切解决方案的。例如，为了找到图1-8所示问题的解决方案，需要检查10的阶乘这么多条线路，这等于是3 628 800条线路。如果把问题扩展为100座城市，就需要检查100的阶乘条线路，这是无法用现有方法实现的，因此需要另找一种近似的解决方案。

网络是一种特殊类型的图，其中每一条边都被赋予了权。权与使用某边从一个节点移动到另一个节点所花费的代价相关。图1-9描述了带权的城市网络，其中的权是两座城市（节点）之间的公里数。

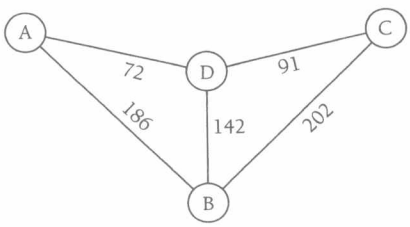


图1-9 网络群集

至此已经对将要在本书中讨论的不同群集类型做了总体的概述。下面就准备实际看一看 10