21世纪计算机专业大专系列教材

李大友 主编

数据结构习题与解答

彭 波 编著

清华大学出版社
http://www.tup.tsinghua.edu.cn

21世纪计算机专业大专系列教材
李大友　主编

数据结构习题与解答

彭　波　编著

清华大学出版社
北京

内 容 简 介

本书是《21世纪计算机专业大专系列教材》中《数据结构》一书的配套用书。书中除了给出主教材中所有的习题提示、解析和答案之外，对有些题目给出了多种算法解答，还针对各章的内容适当地补充了一定数量的习题(带*号的习题)，并给出答案及解析。全书共分9章，包括数据结构基础知识、线性表、栈和队列、串、数组和广义表、树与二叉树、图、查找和排序。

本书使用类C语言作为算法描述语言，且所有算法都可以在任何一种C语言的开发环境中实现。在随书的配套光盘中可以看到这些算法的C语言程序。

本书内容丰富、题型多样、涉及面广、实用性强，对开拓思路具有很好的启发作用。

本书可供计算机专业的学生学习使用，也可供教师或其他专业技术人员参考。

书　　名：数据结构习题与解答

作　　者：彭　波 编著

出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)

http://www.tup.tsinghua.edu.cn

责任编辑：徐跃进

印 刷 者：世界知识印刷厂

发 行 者：新华书店总店北京发行所

开　　本：787×1092　1/16　**印张**：12　**字数**：273千字

版　　次：2003年3月第1版　2003年3月第1次印刷

书　　号：ISBN 7-89494-012-7

印　　数：0001～6000

定　　价：17.00元　(含盘)

《21世纪计算机专业大专系列教材》
编辑委员会名单

序

这套教材为21世纪高等学校计算机专业大专系列教材。

我们从1995年开始组织《计算机专业大专系列教材》。当时根据中国计算机学会教育委员会与全国高等学校计算机教育研究会联合推荐的《计算机学科教学计划1993》的要求，组织了《计算机组成原理》等13本教材，并由清华大学出版社出版。这套教材出版后，受到了高等学校师生的广泛欢迎和好评。

在组织上述教材的时候，主要是按《计算机学科教学计划1993》的要求进行的。而1993教学计划主要是参照美国IEEE和ACM《计算机学科教学计划1991》并结合我国高等教育当时的实际情况制定的，反映的是20世纪80年代末计算机学科的发展状况。

计算机学科是一个飞速发展的新兴学科，发展速度之快可谓一日千里。近10年来，计算机学科已发展成为一个独立学科，计算机本身向高度集成化、网络化和多媒体化迅速发展。但从另一个方面来看，高等学校的计算机教育一直滞后于计算机学科的发展，特别是教材建设，由于受时间和软硬条件的限制，更是落后于现实需要，而大专层次的教材建设问题尤其严重。为了改变这种状况，高等学校的教育工作者和专家教授们应当仁不让地投入必要的时间和精力来完成这一历史使命。

为组织好这套教材，我们认真地研究了全国高等学校计算机专业教学指导委员会和中国计算机学会教育委员会联合推荐的《计算机学科教学计划2000》和美国IEEE和ACM两个学会最新公布的《计算机学科教学计划2001》。这两个教学计划都是在总结了从《计算机学科教学计划1991》到现在计算机学科十年来发展的主要成果的基础上诞生的。它们所提供的指导思想和学科所涵盖的内容，不仅适合于大学本科，也适合大学专科的需求，关键在于要对其内容的取舍进行认真的研究。

在我国的《计算机学科教学计划1993》和美国IEEE和ACM两个学会提出的《计算机学科教学计划1991》中，根据当时的情况，只提出了9个主科目。而在《计算机学科教学计划2001》中，根据学科的最新发展状况，提出了14个主科目，其中13个主科目又为核心主科目。这14个主科目是：算法与分析(AL)、体系结构(AR)、离散结构(DS)、计算科学(CN)、图形学与可视化计算(GV)、网络计算(NC)、人机交互(HC)、信息管理(IM)、智能系统(IS)、操作系统(OS)、程序设计基础(PF)、程序设计语言(PL)、软件工程(SE)、社会、道德、法律和专业问题(SP)，其中除CN为非核心主科目外，其他13个主科目均为核心主科目。

将美国IEEE和ACM的教学计划2001与1991计划进行比较可看出：在1991计划中，离散结构只是作为数学基础提出，未被列为主科目；而在2001计划中，不但列为主科

目，而且为核心主科目。可见，已将离散结构提升为本学科的基础。

在 1991 计划中，未提及网络计算，而在 2001 计划中，不但提出，而且被列为核心主科目，以适应网络技术飞速发展的需求。

图形学与可视化计算也是为适应发展需求新增的内容，并且列为主科目。

除此之外，2001 计划在下述 5 个方面做了增加或调整。

• 将程序设计语言引论调整为程序设计基础和程序设计语言两个核心主科目，显然，加强了对程序设计的要求。

• 将人-机通信调整为人机交互，反映了人-机通信的实质是人机交互。在图形界面迅速发展的今天，人机交互理论和方法的研究和应用变得十分重要。

• 将人工智能与机器人学调整为智能系统，拓宽了对智能系统的要求。

• 将数据库与信息检索调整为信息管理，因为后者不仅概括了前者，而且反映了数据库与信息检索的实质是信息管理。

• 将数值与符号计算调整为计算科学，更具有概括性。

总之，上述变化不仅更好地反映了计算机学科的发展现状，而且使 2001 教学计划具有更强的科学性和实用性。

由于这套系列教材主要面向的对象是计算机专业三年制大专(高职)学生，其培养目标也应属于高级技术人才的层次。他们既要有一定的理论基础(较本科弱)，又要更强调实用性，要有明确的应用方向。我们将应用方向定位在信息管理和计算机网络两个方向。这两个应用方向占计算机应用总计的 90%以上。

在系列教材的内容取舍上，2001 教学计划的 14 门主科目中，我们概括了除智能系统、计算科学和社会、道德、法律和专业问题之外的其他 11 个主科目。在每个主科目中，我们都以其中的基本概念、基本理论和基本方法作为主线组织教材，使学生既能掌握基本的基础理论和方法，又能为他们进一步深造打下必要的基础；在信息管理和计算机网络技术两个应用方向上，他们的应用能力将得到加强。

根据上述指导思想，初步确定组织 20 本左右的教材供各高校选用。这些教材包括：《离散数学》、《计算机应用基础》、《计算机组织与结构》、《微机系统与接口技术》、《计算机网络与通信》、《网络管理技术基础》、《计算机网络系统集成技术》、《数据结构》、《操作系统原理》、《实用软件工程基础》、《数据库原理与应用》、《管理信息系统原理与应用》、《办公自动化实用技术》、《多媒体技术及其应用》、《Internet 技术及其应用》、《计算机维护技术》、《C 语言程序设计》、《Java 语言程序设计》、《C++语言程序设计》、《VB 语言程序设计》、《计算机英语》等。

系列教材并不是教学计划，由于各高校情况不同，培养方向的侧重面也不一样，因此教学计划也不会雷同。教材按系列组织，力图能够反映计算机学科大专层次的总体要求，同时采用大拼盘结构，各校可根据自身情况选择使用。例如，语言类教材，我们就准备了多本，各校可选择其中的一本或两本，其他依此类推。

这套教材均由高等学校具有丰富教学实践经验的老师编写。所编教材体系结构严谨、层次清晰、概念准确、理论联系实际、深入浅出、通俗易懂。相信一定能够得到专科院校计算机专业师生的欢迎。

全国高等学校计算机教育研究会副理事长
课程与教材建设委员会主任
李大友
2001.6

前　言

“数据结构”在计算机科学中是一门综合性的专业基础课。数据结构的研究不仅涉及到计算机硬件(特别是编码理论、存储装置和存取方法等),而且和计算机软件的研究有着更密切的关系,无论是编译程序还是操作系统,都涉及到数据元素在存储器中的分配问题。在研究信息检索时也必须考虑如何组织数据,以便查找和存取数据元素更为方便。可以认为数据结构是介于数学、计算机硬件和计算机软件三者之间的一些核心内容,是从事计算机科学及其应用的科技工作者必须掌握的。

数据结构课程的教学目标是要求学生学会分析研究计算机加工的数据对象的特征,以便在实际应用中选择适当的数据结构、存储结构和相应算法,初步掌握算法的时间与空间性能分析技巧,并培养复杂程序设计的技能。

数据结构课程的主要特点如下:

- 内容丰富、学习量大;
- 隐含在各部分内容中的方法和技术多;
- 贯穿全书的动态链表存储结构和递归技术往往不容易掌握,算法设计具有动态性和抽象性,看懂听懂与掌握会用之间具有相当大的距离。

这些特点使得这门课程的学习难度很大,许多学生在解答习题所需要的各种方法和技术都在教科书中,只不过其形式多样,需要仔细体会才能掌握。

本书是《21世纪计算机专业大专系列教材》中《数据结构》的配套教材,除了给出主教材中所有的习题提示、解析和答案之外,还针对各章的内容适当地补充了一定数量的习题(带*号的习题),并给出答案及解析。全书共分9章,包括数据结构基础知识、线性表、栈和队列、串、数组和广义表、树与二叉树、图、查找和排序。本书与“数据结构”课程主要内容紧密结合,对开拓思路具有启发作用,有助于对课程的掌握。在该配套教材中,使用类C语言作为算法描述语言,且所有算法都可以在任何一种C语言的开发环境中实现。在随书的配套光盘中可以看到这些算法的C语言程序。

本书的出发点是帮助学生进一步学好“数据结构”这门课程,所以在使用本书的过程中需要注意以下几点。

(1) 同步学习、打好基础。与课程内容学习同步,有利于进一步理解掌握各种知识点和巩固课堂教学的效果。

(2) 认真理解,动手练习。算法设计具有非惟一性,本书对算法设计题目给出了一种或者几种算法答案,要在学习、理解、领会的基础上动手设计算法程序,这样才会取得良好的效果。

(3) 举一反三、触类旁通。课程的内容是有限的,但是运用所介绍的知识、方法和技术解决的实际问题是无限的,重在掌握基本原理、基本方法和基本技术,并注意灵活运用。

由于习题较多，在解答上可能存在疏漏之处，欢迎读者批评指正；如果有问题需要与作者联系，请发送电子邮件到：pengbo_cau@sina.com。

本教材由彭波编写，在上机实现教材中所有算法的过程中得到了孙一林、许振文、胡治国、吕小晴、崔永普、王茜等同志的帮助，在此表示感谢。

编 者

2002.9

目　录

第1章 绪 论

1.1 配书习题

1. 简述下列术语：数据、数据项、数据元素、数据对象、数据逻辑结构、数据物理结构、数据结构、数据类型、算法。

解

数据：数据是对客观事物的符号表示，在计算机科学中是指所有能输入到计算机中并被计算机程序处理的符号的总称，它是计算机程序加工的“原料”。

数据项：数据项是数据处理中的最小单位，若干个数据项可以组成一个数据元素。

数据元素：数据元素是数据的基本单位，也称为结点，在计算机程序中通常作为一个整体进行考虑和处理。

数据对象：数据对象是性质相同的数据元素的集合，是数据的一个子集。

数据逻辑结构：数据逻辑结构定义是对操作对象的一种数学描述，也就是说，是从操作对象抽象出来的数学模型。它反映的是数据元素之间的逻辑(数学)关系，并不依赖于计算机。

数据物理结构：数据物理结构，又称存储结构，是数据在计算机存储器中的表示。它包括数据本身在计算机中的存储方式，以及数据之间的逻辑关系在计算机中的表示。与数据逻辑结构不同，它是依赖于计算机的。其主要有两种：顺序存储结构和链式存储结构。

数据结构：数据结构是带有结构特性的数据元素的集合。它研究的是数据逻辑结构和数据物理结构以及它们之间的相互关系，并对这种结构定义相适应的运算，设计出相应的算法，而且确保经过这些运算以后所得到的新结构仍然是原来的结构类型。

数据类型：数据类型是和数据结构密切相关的一个概念，它最早出现在高级程序设计语言中，用以刻画(程序)操作对象的特性。在程序中，形式不同的数据采用数据类型来标识。变量的数据类型说明了变量可能取得的值的集合，以及可能施于变量的操作的集合。所以数据类型不仅定义了一组形式相同的数据集，也定义了对这组数据可施行的一组操作集。

算法：算法是对特定问题求解步骤的一种描述，它是指令的有限序列，其中每一条指令表示一个或者多个操作。算法设计依赖于数据的存储结构，对于确定的问题，人们往往寻求在适宜的存储结构上设计一种效率较高的算法。

2. 将数量级 $O(1)$，$O(n)$，$O(n^2)$，$O(n^3)$，$O(n\log n)$，$O(\log n)$，$O(2^n)$按增长率从小到大排列。

解

在题目给出的7种类型的数量级中：$O(1)$为常量型，$O(n)$为线性型，$O(n^2)$为平方型，$O(n^3)$为立方型，$O(n\log n)$为线性对数型，$O(\log n)$为对数型，$O(2^n)$为指数型。这7种类型按增长率从小到大排列如下：

$$O(1)<O(\log n)<O(n)<O(n\log n)<O(n^2)<O(n^3)<O(2^n)$$

3. 求下列程序段的时间复杂度。

```
(1) for (i=1; i<=n; i++)
      for (j=1; j<=n; j++)
        s++;
```

解

时间复杂度为$O(n^2)$。

```
(2) for(i=1; i<=n; i++)
      for (j=1; j<=i; j++ )
        s++;
```

解

时间复杂度为$O(n^2)$。

```
(3) for (i=1; i<=n; i++)
      for (j=i; j<=n; j++)
        s++;
```

解

时间复杂度为$O(n^2)$。

```
(4) for (i=0; i<m; i++)
      for (j=0; j<n; j++)
        A[i][j]=i * j;
```

解

时间复杂度为$O(m\times n)$。

```
(5) for (i=0; i<n-1; i++)
      for (j=n; j<i; j--)
        x++;
```

解

时间复杂度为$O(n^2)$。

```
(6) x=n;        // n>1
    y=0;
    while (x>=(y+1) * (y+1))
      y=y+1;
```

解

时间复杂度为 $O(n^{1/2})$。

```
(7) i=0;
    s=0;
    while (s<n) {
      i++;
      s=s+i;
    };
```

解

时间复杂度为 $O(n^{1/2})$。

```
(8) for (i=0; i<m; i++)
      for (j=0; j<t; j++)
        c[i][j]=0;
      for (i=0; i<m; i++)
        for (j=0; j<t; j++)
          for (k=0; k<n; k++)
            c[i][j]=c[i][j]+a[i][k] * b[k][j];
```

解

时间复杂度为 $O(m\times n\times t)$。

```
(9) i=1;
    k=0;
    n=100;
    do {
      k=k+10 * i;
      i=i++;
    } while (i !=n);
```

解

时间复杂度为 $O(n)$。

```
(10) i=1;
     k=0;
     while (i<=n-1) {
       k=k * 10 * i;
       i++;
     }
```

解

时间复杂度为 $O(n)$。

```
(11) i=1;
     j=0;
```

```
while ((i+j)<=n)
  if (i>j)
    j++;
  else  i++;
```

解

时间复杂度为 $O(n)$。

(12)
```
i=1;
do  {
  j=1;
  do  {
    printf ("% d \ n", i*j);
    j++;
  } while (j>n)
  i++;
} while (i>n);
```

解

时间复杂度为 $O(1)$。

(13)
```
m=91;
n=100;
while (n>0)
    if (m>0)  {
      m=m-10;
      n=n-1;
    }
    else  m=m+1;
```

解

时间复杂度为 $O(n)$。

(14)
```
for (i=0; i<n; i++)
    for (j=0; j<i; j++)
      for (k=0; k<j; k++)
        for (t=0; t<k; t++)
          x=x+1;
```

解

时间复杂度为 $O(n^4)$。

4. 已知输入 x、y、z 3 个不相等的整数，试设计一个算法，使这 3 个数按从小到大的顺序输出，并考虑此算法的比较次数和元素的移动的次数。

解

(1) 算法编写

```
void sort ()  {
// 比较 x、y、z 3 个数的大小,并按照从小到大的顺序输出
  int x, y, z, t;
  scanf ("%d%d%d\n", &x, &y, &z);
  if (x>y)  {
  // 将 x 与 y 比较,较大的数放在 y 中,较小的数放在 x 中
     t=x;
     x=y;
     y=t;
  } // if 结束
  if (y>z)  {
  // 将 y 与 z 比较,较大的数放在 z 中,较小的数放在 t 中
     t=z;
     z=y;
     if (x<t)
     // 将 x 与 t 比较,较大的数放在 y 中,较小的数放在 x 中
       y=t;
     else  {
       y=x;
       x=t;
     } // else 结束
  printf ("%d%d%d\n", x, y, z);
} // sort
```

(2) 算法分析

从以上算法可知:按照从小到大的顺序输出 3 个整数 x、y、z 需要进行 3 次比较,在最坏的情况下需要移动 7 次记录。

5. 猴子吃桃问题。

猴子第一天摘下若干个桃子,当即吃了一半,还不过瘾,又多吃了一个;第二天早上又将剩下的桃子吃掉了一半,又多吃了一个;以后每天早上都吃了前一天剩下的一半零一个,到第十天早上想再吃时,发现只剩下一个桃子了。求第一天共摘了多少个桃子。

解

(1) 算法编写

```
main ()  {
// 猴子吃桃问题
  int day, x1, x2;
  day=9;
  x2=1;
  while (day>0)  {
    x1=(x2+1) * 2;
    // 前一天的桃子数是今天桃子数加 1 后的 2 倍
    x2=x1;
```

```
    day--;
  }
  printf("桃子总数=%d\n", x1);
} // main
```

(2) 执行结果：桃子总数为 1534。

1.2 补充习题

***6.** 指出下列各算法的时间复杂度。

```
(1) void prime(int n) {
    // n 为一个正整数
      int i=2;
      while((n % i)!=0 && i*1.0<sqrt(n))
        i++;
      if(i*1.0>sqrt(n))
        printf("%d 是一个素数\n", n);
      else
        printf("%d 不是一个素数\n", n);
    } // prime
```

解

算法的时间复杂度是由嵌套最深层语句的执行次数决定的。prime 算法的嵌套最深层语句为：

```
i++;
```

它的执行次数由条件((n % i)!=0 && i * 1.0<**sqrt**(n))决定，显然执行次数小于 sqrt(n)，所以 prime 算法的时间复杂度是 $O(n^{1/2})$。

```
(2) sum1(int n) {
    // n 为一个正整数
      int p=1, sum=0, i;
      for(i=1; i<=n; i++) {
        p *=i;
        sum += p;
      }
      return (sum);
    // sum1
```

解

算法的时间复杂度是由嵌套最深层语句的执行次数决定的。sum1 算法的嵌套最深层语句为：

```
p *=i;
```

```
sum += p;
```

它的执行次数为 n 次，所以 sum1 算法的时间复杂度是 $O(n)$。

```
(3) sum2 (int n)  {
    // n 为一个正整数
      int sum=0, i, j;
      for (i=1; i<=n; j++)  {
        p=1;
        for (j=1; j<=i; i++)
          p *=j;
        sum +=p;
      }
      return (sum)
    } // sum2
```

解

算法的时间复杂度是由嵌套最深层语句的执行次数决定的。sum2 算法的嵌套最深层语句：

```
p *=i;
```

它的执行次数为 $1+2+3+\cdots+n=n(n+1)/2$ 次，所以 sum2 算法的时间复杂度是 $O(n^2)$。

***7.** 求两个 n 阶矩阵的乘法 $C=A\times B$，其算法如下：

```
#define MAX 100
void matrixmult (int n, float a[MAX][MAX],  float c[MAX][MAX] )  {
  int i, j, k;
  float x;
  for (i=1; i<=n; i++)  {
    for (j=1; j<=n; j++)  {
      x=0;
      for (k=1; k<=n; k++)
        x+=a[i][k] * b[k][j]
      c[i][j] = x;
    } // for ( j = 1) 结束
  } // for ( i = 1) 结束
} // matrixmult
```

试分析该算法的时间复杂度。

解

该算法中的主要语句：	执行次数：
`for (i=1; i<=n; i++)`	$n+1$
`for (j=1; j<=n; j++)`	$n(n+1)$