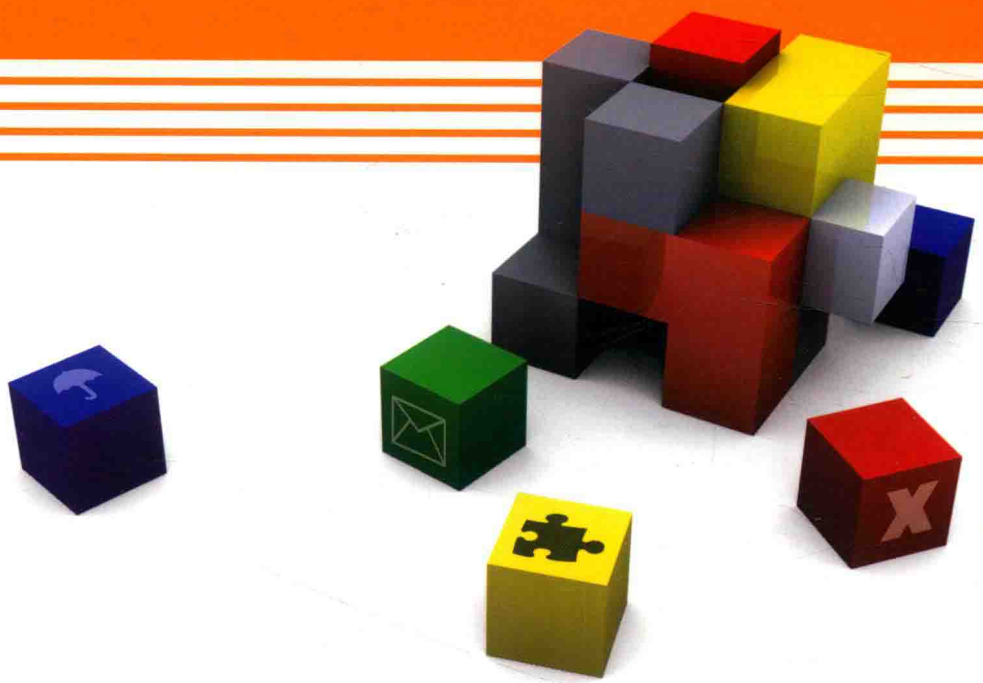
“十二五”普通高等教育本科国家级规划教材
高等学校数据结构课程系列教材

（第5版）

数据结构教程

学习指导

李春葆　主编



- 高度概括本章知识结构图、基本知识点和要点归纳。
- 提供主教材中练习题的参考答案。
- 提供各种题型的练习题及参考答案。
- 提供不同类型的考试试卷及参考答案。

清华大学出版社

“十二五”普通高等教育本科国家级规划教材
高等学校数据结构课程系列教材

（第5版）

数据结构教程学习指导

李春葆 主编

尹为民 蒋晶珏 喻丹丹 蒋林 编著

清华大学出版社
北 京

内 容 简 介

本书是《数据结构教程(第5版)》(李春葆等编著,清华大学出版社出版)的配套学习指导书。两书章节一一对应,内容包括绪论、线性表、栈和队列、串、递归、数组和广义表、树和二叉树、图、查找、内排序、外排序和文件。各章中除给出本章练习题的参考答案以外还总结了本章的知识体系结构,并补充了大量的练习题且予以解析,因此自成一体,可以脱离主教材单独使用。

本书适合高等院校计算机和相关专业的本科生及研究生使用。

图书在版编目(CIP)数据

数据结构教程(第5版)学习指导/李春葆主编. —北京:清华大学出版社,2017
(高等学校数据结构课程系列教材)
ISBN 978-7-302-45587-5

Ⅰ. ①数… Ⅱ. ①李… Ⅲ. ①数据结构—教学参考资料 Ⅳ. ①TP311.12

中国版本图书馆CIP数据核字(2016)第283889号

责任编辑:魏江江 王冰飞
封面设计:杨 兮
责任校对:时翠兰
责任印制:沈 露

出版发行:清华大学出版社
网 址:http://www.tup.com.cn, http://www.wqbook.com
地 址:北京清华大学学研大厦A座 邮 编:100084
社 总 机:010-62770175 邮 购:010-62786544
投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn
质 量 反 馈:010-62772015, zhiliang@tup.tsinghua.edu.cn
课 件 下 载:http://www.tup.com.cn,010-62795954
印 装 者:北京国马印刷厂
经 销:全国新华书店
开 本:185mm×260mm **印 张**:22.75 **字 数**:554千字
版 次:2017年7月第1版 **印 次**:2017年7月第1次印刷
印 数:1~2000
定 价:39.50元

产品编号:072425-01

前言 Preface

本书是《数据结构教程(第5版)》(清华大学出版社,以下简称为《教程》)的配套学习指导书。全书分为12章,第1章为绪论;第2章为线性表;第3章为栈和队列;第4章为串;第5章为递归;第6章为数组和广义表;第7章为树和二叉树;第8章为图;第9章为查找;第10章为内排序;第11章为外排序;第12章为文件。本书各章次与《教程》的章次相对应。附录A给出了两份本科生期末考试试题及参考答案,附录B给出了两份研究生入学考试(单考)数据结构部分试题及参考答案,附录C给出了两份全国计算机学科专业考研题数据结构部分试题及参考答案。

每章包括以下内容。

- 本章知识体系:高度概括本章知识结构图、基本知识点和要点归纳。
- 教材中的练习题及参考答案:给出了《教程》中对应章节练习题的参考答案。
- 补充练习题及参考答案:列出了大量相关的练习题,并按单项选择题、填空题、判断题、简答题和算法分析题或算法设计题分类,同时给出了这些题目的参考答案。其中许多题目是多年来全国各高校计算机专业的数据结构考研题。

书中列出了全部的练习题题目,因此自成一体,可以脱离《教程》单独使用。

由于水平所限,尽管编者不遗余力,仍可能存在错误和不足之处,敬请教师和同学们批评指正。

编　者

2017年1月

目录 Contents

第1章 绪论

1.1 本章知识体系

1. 知识结构图

本章的知识结构如图 1.1 所示。

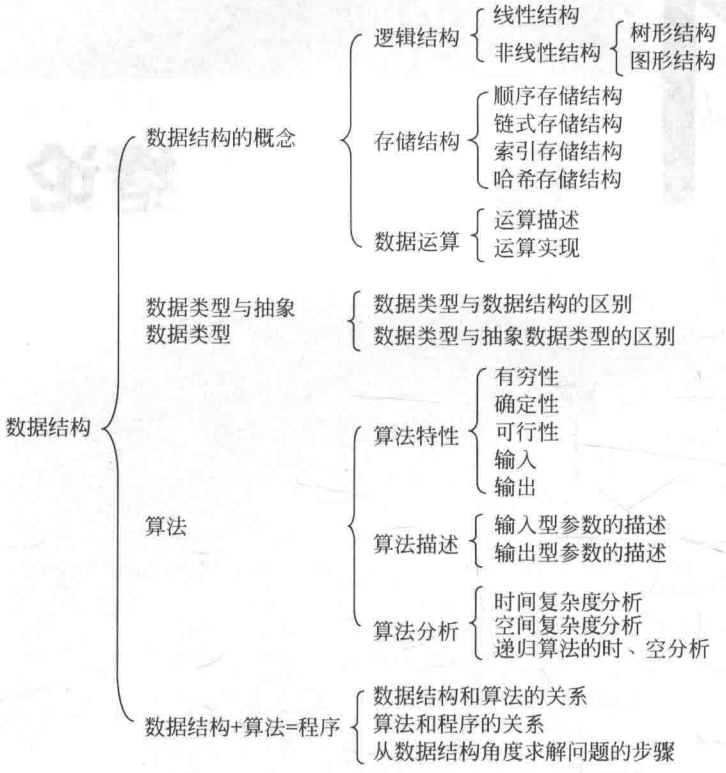


图 1.1 第 1 章知识结构图

2. 基本知识点

(1) 数据的逻辑结构、存储结构和数据运算三方面的概念及相互关系。

(2) 采用抽象数据类型描述求解问题。

(3) 算法描述中的输出型参数描述方法。

(4) 算法的时间和空间复杂度分析,特别是递归算法的时间和空间复杂度分析。

(5) 如何设计“好”的算法。

3. 要点归纳

(1) 数据结构是相互之间存在一种或多种特定关系的数据元素的集合。数据是由数据元素组成的,数据元素可以由若干个数据项组成,数据元素是数据的基本单位,数据项是数据的最小单位。

(2) 数据结构一般包括数据逻辑结构、数据存储结构和数据运算 3 个方面。数据运算

分为抽象运算(运算功能描述)和运算实现两个层次。

(3) 数据的逻辑结构分为集合、线性结构、树形结构和图形结构,树形结构和图形结构统称为非线性结构。

(4) 数据的存储结构分为顺序存储结构、链式存储结构、索引存储结构和哈希(散列)存储结构。

(5) 在设计数据的存储结构时既要存储逻辑结构的每个元素,又要存储元素之间的逻辑关系。同一逻辑结构可以设计相对应的多个存储结构。

(6) 描述一个求解问题的抽象数据类型由数据逻辑结构和抽象运算两部分组成。

(7) 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。运算实现通过算法来表示。

(8) 算法具有有穷性、确定性、可行性、输入和输出 5 个重要特征。

(9) 算法满足有穷性,程序不一定满足有穷性。算法可以用计算机程序来描述,但并不是说任何算法必须用程序来描述。

(10) 在用 C/C++语言描述算法时,通常算法采用 C/C++函数的形式来描述,复杂算法可能需要多个函数来表示。

(11) 在设计一个算法时先要弄清哪些是输入(已知条件)、哪些是输出(求解结果),通常将输入参数设计成非引用型形参,将输出参数设计成引用型形参。在有些情况下,算法求解结果可以用函数返回值表示。

(12) 对于算法的输入通常需要判断其有效性,当输入有效并正确执行时返回 true(真),否则返回 false(假)。

(13) 算法分析包括时间复杂度和空间复杂度分析,其目的是分析算法的效率以求改进,并不是要分析算法的绝对执行时间。

(14) 在分析算法的时间复杂度时通常选取算法中的基本运算,求出其频度,取最高阶并置序数为 1 作为该算法的时间复杂度。

(15) 递归算法的时间复杂度分析方法是先由递归算法推导出执行时间的递推式,然后计算 $T(n)$,再用 O 表示。

(16) 递归算法的空间复杂度分析方法是先由递归算法推导出占用空间的递推式,然后计算 $S(n)$,再用 O 表示。

(17) 通常算法是建立在数据存储结构之上的,设计好的存储结构可以提高算法的效率。

(18) 求解问题的一般步骤是建立其抽象数据类型,针对运算的实现设计出合理的存储结构,在此基础上设计出尽可能高效的算法。

1.2 教材中的练习题及参考答案

1. 简述数据与数据元素的关系与区别。

答:凡是能被计算机存储、加工的对象统称为数据,数据是一个集合。数据元素是数据的基本单位,是数据的个体。数据元素与数据之间的关系是元素与集合之间的关系。

2. 采用二元组表示的数据逻辑结构 $S=<D,R>$，其中 $D=\{a,b,\cdots,i\}$，$R=\{r\}$，$r=\{<a,b>,<a,c>,<c,d>,<c,f>,<f,h>,<d,e>,<f,g>,<h,i>\}$，问：关系 r 是什么类型的逻辑结构？哪些结点是开始结点，哪些结点是终端结点？

答：该逻辑结构为树形结构，其中 a 结点没有前驱结点，它是开始结点，b、e、i 和 g、结点没有后继结点，它们都是终端结点。

3. 简述数据逻辑结构与存储结构的关系。

答：在数据结构中，逻辑结构与计算机无关，存储结构是数据元素之间的逻辑关系在计算机中的表示。存储结构不仅将逻辑结构中的所有数据元素存储到计算机内存中，而且还要在内存中存储各数据元素间的逻辑关系。通常情况下，一种逻辑结构可以有多种存储结构，例如线性结构可以采用顺序存储结构或链式存储结构表示。

4. 简述数据结构中运算描述和运算实现的异同。

答：运算描述是指逻辑结构施加的操作，而运算实现是指一个完成该运算功能的算法。它们的相同点是运算描述和运算实现都能完成对数据的“处理”或某种特定的操作，不同点是运算描述只是描述处理功能，不包括处理步骤和方法，而运算实现的核心是设计处理步骤。

5. 数据结构和数据类型有什么区别？

答：数据结构是相互之间存在一种或多种特定关系的数据元素的集合，一般包括 3 个方面的内容，即数据的逻辑结构、存储结构和数据的运算。数据类型是一个值的集合和定义在这个值集上的一组运算的总称，如 C 语言中的 short int 数据类型是由－32 768～32 767（16 位机）的整数和＋、－、＊、/、%等运算符构成的。

6. 在 C/C++中提供了引用运算符，简述其在算法描述中的主要作用。

答：在算法设计中，一个算法通常用一个或多个 C/C++函数来实现，在 C/C++函数之间传递参数时有两种情况，一是从实参到形参的单向值传递；二是实参和形参之间的双向值传递。对形参使用引用运算符即在形参名前加上“&”，不仅可以实现实参和形参之间的双向值传递，而且使算法设计简单、明晰。

7. 有以下用 C/C++语言描述的算法，说明其功能：

```
void fun(double &y,double x,int n)
{   y=x;
    while (n>1)
    {   y=y*x;
        n--;
    }
}
```

答：本算法的功能是计算 $y=x^n$。

8. 用 C/C++语言描述下列算法，并给出算法的时间复杂度。

（1）求一个 n 阶整数数组的所有元素之和。

（2）对于输入的任意 3 个整数，将它们按从小到大的顺序输出。

（3）对于输入的任意 n 个整数，输出其中的最大和最小元素。

答：(1) 算法如下。

```
int sum(int A[N][N],int n)
{   int i,j,s=0;
    for (i=0;i<n;i++)
        for (j=0;j<n;j++)
            s=s+A[i][j];
    return(s);
}
```

本算法的时间复杂度为 $O(n^2)$。

(2) 算法如下。

```
void order(int a,int b,int c)
{   if (a>b)
    {   if (b>c)
            printf("%d,%d,%d\n",c,b,a);
        else if (a>c)
            printf("%d,%d,%d\n",b,c,a);
        else
            printf("%d,%d,%d\n",b,a,c);
    }
    else
    {   if (b>c)
        {   if (a>c)
                printf("%d,%d,%d\n",c,a,b);
            else
                printf("%d,%d,%d\n",a,c,b);
        }
        else printf("%d,%d,%d\n",a,b,c);
    }
}
```

本算法的时间复杂度为 $O(1)$。

(3) 算法如下。

```
void maxmin(int A[],int n,int &max,int &min)
{   int i;
    min=min=A[0];
    for (i=1;i<n;i++)
    {   if (A[i]>max)  max=A[i];
        if (A[i]<min)  min=A[i];
    }
}
```

本算法的时间复杂度为 $O(n)$。

9. 设 3 个表示算法频度的函数 f、g 和 h 分别为：

$$f(n) = 100n^3 + n^2 + 1000$$

$$g(n) = 25n^3 + 5000n^2$$

$$h(n) = n^{1.5} + 5000n\log_2 n$$

求它们对应的时间复杂度。

答：$f(n)=100n^3+n^2+1000=O(n^3)$，$g(n)=25n^3+5000n^2=O(n^3)$

当 $n\to\infty$ 时，$\sqrt{n}>\log_2 n$，所以 $h(n)=n^{1.5}+5000n\log_2 n=O(n^{1.5})$。

10. 分析下面程序段中循环语句的执行次数。

```
int j = 0,s = 0,n = 100;
do
{   j = j + 1;
    s = s + 10 * j;
} while (j<n && s<n);
```

答：$j=0$，第 1 次循环 $j=1$，$s=10$。第 2 次循环 $j=2$，$s=30$。第 3 次循环 $j=3$，$s=60$。第 4 次循环 $j=4$，$s=100$。while 条件不再满足。所以其中循环语句的执行次数为 4。

11. 设 n 为正整数，给出下列 3 个算法关于问题规模 n 的时间复杂度。

(1) 算法 1：

```
void fun1(int n)
{   i = 1,k = 100;
    while (i<=n)
    {   k = k + 1;
        i += 2;
    }
}
```

(2) 算法 2：

```
void fun2(int b[],int n)
{   int i,j,k,x;
    for (i = 0;i<n-1;i++)
    {   k = i;
        for (j = i + 1;j<n;j++)
            if (b[k]>b[j]) k = j;
        x = b[i];b[i] = b[k];b[k] = x;
    }
}
```

(3) 算法 3：

```
void fun3(int n)
{   int i = 0,s = 0;
    while (s<=n)
    {   i++;
        s = s + i;
    }
}
```

答：(1) 设 while 循环语句的执行次数为 $T(n)$，则有以下关系。

$i=2T(n)+1\leqslant n$，即 $T(n)\leqslant(n-1)/2=O(n)$。

(2) 算法中的基本运算语句是 if ($b[k]>b[j]$) $k=j$，其执行次数 $T(n)$为：

$$T(n)=\sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1}1=\sum_{i=0}^{n-2}(n-i-1)=\frac{n(n-1)}{2}=O(n^2)$$

(3) 设 while 循环语句的执行次数为 $T(n)$，有：

$$s=1+2+\cdots+T(n)=\frac{T(n)(T(n)+1)}{2}\leqslant n$$

则 $T(n)=O(\sqrt{n})$。

12. 有以下递归算法用于对数组 $a[i..j]$的元素进行归并排序：

```
void mergesort(int a[],int i,int j)
{   int m;
    if (i!= j)
    {   m = (i + j)/2;
        mergesort(a,i,m);
        mergesort(a,m + 1,j);
        merge(a,i,j,m);
    }
}
```

求执行 mergesort(a,0,$n-1$)的时间复杂度。其中，merge(a,i,j,m)用于两个有序子序列 $a[i..m]$和 $a[m+1..j]$的合并，是非递归函数，它的时间复杂度为 O(合并的元素个数)。

答：设 mergesort(a,0,$n-1$)的执行时间为 $T(n)$，分析得到以下递归关系。

$$T(n)=\begin{cases}O(1) & n=1\\ 2T(n/2)+O(n) & n>1\end{cases}$$

其中，$O(n)$为 merge()所需的时间，设为 cn(c 为常量)。因此：

$$T(n)=2T\left(\frac{n}{2}\right)+cn=2\left(2T\left(\frac{n}{2^2}\right)+\frac{cn}{2}\right)+cn=2^2T\left(\frac{n}{2^2}\right)+2cn=2^3T\left(\frac{n}{2^3}\right)+3cn$$

$$\vdots$$

$$=2^kT\left(\frac{n}{2^k}\right)+kcn=2^kO(1)+kcn$$

由于$\frac{n}{2^k}$趋近于 1，$k=\log_2 n$。所以 $T(n)=2^{\log_2 n}O(1)+cn\log_2 n=n+cn\log_2 n=O(n\log_2 n)$。

13. 描述一个集合的抽象数据类型 ASet，其中所有元素为正整数，集合的基本运算包括：

(1) 由整数数组 $a[0..n-1]$创建一个集合。

(2) 输出一个集合的所有元素。

(3) 判断一个元素是否在一个集合中。

(4) 求两个集合的并集。

(5) 求两个集合的差集。

(6) 求两个集合的交集。

在此基础上设计集合的顺序存储结构,并实现各基本运算的算法。

答:抽象数据类型 ASet 的描述如下。

```
ADT ASet
{   数据对象: D={d_i|0≤i≤n,n 为一个正整数}
    数据关系: 无.
    基本运算:
        createset(&s,a,n): 创建一个集合 s;
        dispset(s): 输出集合 s;
        inset(s,e): 判断 e 是否在集合 s 中;
        void add(s1,s2,s3): s3 = s1∪s2;              //求集合的并集
        void sub(s1,s2,s3): s3 = s1 - s2;            //求集合的差集
        void intersection(s1,s2,s3): s3 = s1∩s2;     //求集合的交集
}
```

设计集合的顺序存储结构类型如下:

```
typedef struct                  //集合结构体类型
{   int data[MaxSize];          //存放集合中的元素,其中 MaxSize 为常量
    int length;                 //存放集合中的实际元素个数
} Set;                          //将集合结构体类型用一个新类型名 Set 表示
```

采用 Set 类型的变量存储一个集合。对应的基本运算算法设计如下:

```
void createset(Set &s,int a[],int n)          //创建一个集合
{   int i;
    for (i=0;i<n;i++)
        s.data[i]=a[i];
    s.length=n;
}
void dispset(Set s)                           //输出一个集合
{   int i;
    for (i=0;i<s.length;i++)
        printf("%d ",s.data[i]);
    printf("\n");
}
bool inset(Set s,int e)                       //判断 e 是否在集合 s 中
{   int i;
    for (i=0;i<s.length;i++)
        if (s.data[i]==e)
            return true;
    return false;
}
void add(Set s1,Set s2,Set &s3)               //求集合的并集
{   int i;
    for (i=0;i<s1.length;i++)                 //将集合 s1 中的所有元素复制到 s3 中
        s3.data[i]=s1.data[i];
    s3.length=s1.length;
```