“十二五”普通高等教育本科国家级规划教材配套参考书
国家精品在线开放课程主讲教材

Learning and Experimental Guidance of Data Structure

数据结构学习与实验指导

（第2版）

主编　陈　越
编著　何钦铭　徐镜春　魏宝刚　杨　枨

高等教育出版社

“十二五”普通高等教育本科国家级规划教材配套参考书
国家精品在线开放课程主讲教材

Learning and Experimental Guidance of Data Structure

数据结构学习与实验指导

（第2版）

主编 陈 越
编著 何钦铭 徐镜春 魏宝刚 杨 枨

高等教育出版社·北京

内容提要

“数据结构”是计算机类专业最重要的专业基础课之一，主要讲授数据的有效组织方法以及解决实际问题的各种经典算法。而经典算法的威力，往往是在处理大规模数据量时才真正体现。只有让学生动手解决规模较大的问题，才能帮助学生建立感性认识，更好地理解数据结构和算法存在的意义。

本书第0章概要介绍了本书的特点和内容结构；第1章围绕时空复杂度分析与比较提供练习；第2章提供对C语言关键内容的复习性练习；第3章针对线性表的知识点设计应用问题进行练习；第4章围绕树的存储、重要性质与应用进行练习；第5章是对散列表和经典哈希映射技术的应用；第6章设计了对图的各种表示方法和相关算法的训练；第7章通过对各种类型的大规模排序问题的求解，帮助读者理解各种经典排序算法的特点和适用范围；最后第8章给出的题目均涉及多个知识点的综合应用，帮助读者更深刻体会数据结构的灵活运用。希望读者能通过本书的学习提高实践能力，使数据结构与算法成为用计算机解决实际问题的有效工具。

本书可作为高等学校计算机类专业“数据结构”课程的参考用书。

图书在版编目（CIP）数据

数据结构学习与实验指导/陈越主编；何钦铭等编著.--2版.--北京：高等教育出版社，2019.7

ISBN 978-7-04-051550-3

Ⅰ.①数… Ⅱ.①陈… ②何… Ⅲ.①数据结构-高等学校-教学参考资料 Ⅳ.①TP311.12

中国版本图书馆CIP数据核字（2019）第042631号

策划编辑 张 龙	责任编辑 武林晓	封面设计 王 琰	版式设计 马敬茹
插图绘制 于 博	责任校对 高 歌	责任印制 田 甜	

出版发行	高等教育出版社	网 址	http://www.hep.edu.cn
社 址	北京市西城区德外大街4号		http://www.hep.com.cn
邮政编码	100120	网上订购	http://www.hepmall.com.cn
印 刷	三河市华润印刷有限公司		http://www.hepmall.com
开 本	787mm×1092mm 1/16		http://www.hepmall.cn
印 张	20.75	版 次	2013年5月第1版
字 数	460千字		2019年7月第2版
购书热线	010-58581118	印 次	2019年7月第1次印刷
咨询电话	400-810-0598	定 价	39.00元

物料号 51550-00

数据结构学习与实验指导

（第2版）

主编　陈　越

编著　何钦铭　徐镜春

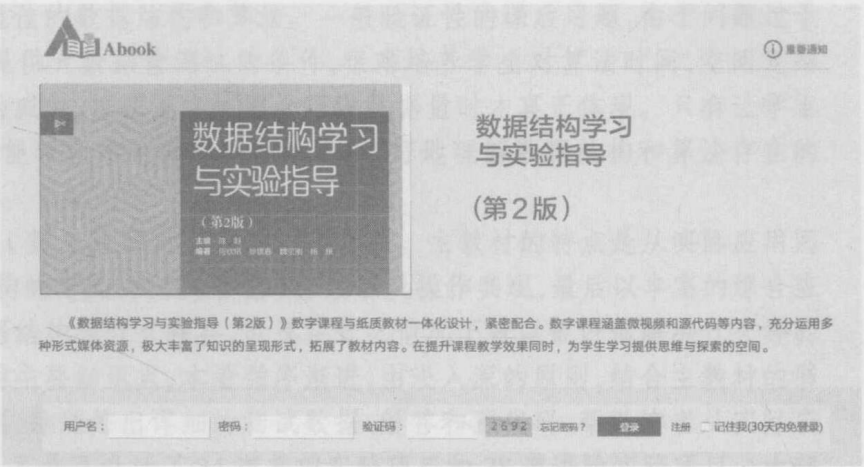
魏宝刚　杨　枨

1　计算机访问http://abook.hep.com.cn/1858125，或手机扫描二维码、下载并安装Abook应用。

2　注册并登录，进入“我的课程”。

3　输入封底数字课程账号（20位密码，刮开涂层可见），或通过Abook应用扫描封底数字课程账号二维码，完成课程绑定。

4　单击“进入课程”按钮，开始本数字课程的学习。



课程绑定后一年为数字课程使用有效期。受硬件限制，部分内容无法在手机端显示，请按提示通过计算机访问学习。

如有使用问题，请发邮件至abook@hep.com.cn。

http://abook.hep.com.cn/1858125

前　言

“数据结构”是计算机类专业最重要的专业基础课之一，主要讲授数据的有效组织方法以及解决实际问题的各种经典算法。在“数据结构”课程学习中，比较常见的问题是，学生了解了所有重要的数据结构和配套的算法，但是不清楚它们为什么重要，于是在遇到实际问题时，往往难以判断和选择什么是最佳的数据结构和算法。一般验证性的课后习题，由于问题过于简单，或只是纸上谈兵，没有提供大数据量测试的条件，很难培养学生对算法时间、空间复杂度的感性认识。而经典算法的威力，往往是在处理大规模数据量时才真正体现。只有让学生动手解决规模较大的问题，才能帮助学生建立感性认识，更好地理解数据结构和算法存在的意义。

本书是主教材《数据结构（第2版）》的辅助学习指导书。主教材的特点是从实际应用问题出发，导出各种经典数据结构的定义、实现（存储）方法以及操作实现，最后以丰富的综合应用案例帮助读者理解这些数据结构为什么存在，以及在什么情况下可以最好地解决什么样的问题。本书围绕着主教材中的主要知识点，本着循序渐进，由浅入深的原则，结合主教材的书后习题，设计了40道实验案例，每题给出详细的测试数据、解答和源代码，帮助读者从实际应用的角度更好地理解知识点。另外还设计了34道基础实验项目和28道进阶实验项目。大部分基础实验项目是实验案例的延伸和提高，可以在实验案例已经提供的解答程序基础上完成。进阶实验项目为学有余力的读者准备，更具挑战性。每题给出详细的测试数据和解题提示，希望有兴趣的读者能在精读实验案例的基础上，自己动手实现一部分实验项目，达到充分锻炼分析问题、解决问题能力的目的。对于能力更强的读者，还可以尝试解决每章最后略带研究性质的思考题，提高自己独立思考和研究的能力。

与第1版相比，第2版的内容更加丰富。题量增加了37道，配有22段部分算法详解的微视频。与主教材的联系也更紧密，给出了主教材书后编程练习题的绝大部分详解。

本书继续提供丰富的系统支持。配套资源中包含书中全部74段源代码。同时提供对外公开的在线系统“拼题A”，有全部实验案例和实验项目（3道开放性实验项目除外）题的在线评判，全天候为广大读者提供免费服务。读者使用本书封四提供的验证码即可登录“拼题A”网站进行在线练习。此外，高校教师还可以发邮件到chenyue@zju.edu.cn，申请将个人在拼题A系统中注册的账号升级为“教师账号”，免费享有出题/用题、布置作业/考试、导入学生、导出教学统计数据等基本的教学组织功能。

本书由浙江大学计算机科学与技术学院教师编写。陈越教授撰写第0、1、5、8章，并负责全书统稿；第2、3章由何钦铭教授编写；第4章由魏宝刚教授编写；第6章由徐镜春副教授编写；第7章由杨枨副教授编写。微视频由陈越教授、何钦铭教授提供。

特别鸣谢杭州百腾教育科技有限公司和网易公司，他们开发维护的拼题A系统为本书提供了强大的技术支持，使得本书能为读者提供更好的服务。

由于作者水平所限，书中不当之处在所难免，敬请广大读者批评指正。

作　者

2018年5月2日

目　录

第0章

概　论

“数据结构”是计算机类专业的重要专业基础课，一般介绍有关数据组织、算法设计、时间和空间效率的概念和通用分析方法。在解决实际问题时，选择合适的数据结构可以带来更高运行或存储效率的算法，反之，选择了特定的算法后也需要设计合适的数据结构与之配合，达到最佳效果。所以在进行程序设计时，必须将数据结构和与之相关的算法结合起来考虑。通过该课程的学习，读者应学会数据的组织方法和现实世界问题在计算机内部的表示方法，针对问题的应用背景，选择合适的数据结构，从而培养高级程序设计技能。

要真正扎实地掌握数据结构的知识，并且更重要的，要能根据实际情况灵活应用数据结构知识去解决问题，仅仅了解理论知识是远远不够的，需要大量动手能力的训练，才能在解决真实问题时得心应手。

《数据结构学习与实验指导（第2版）》是“数据结构”课程的辅助教材，主要围绕数据结构的基本知识点，设计大量练习题目辅助读者学习。通过详细分析一套实验**案例**的解法，帮助读者完成从对知识点的理解，到应用其解决比较复杂的问题，到动手实现解决问题的算法并设计实施有效的测试。由于这些案例大多取自本书作者们的另一本教材《数据结构》（以下简称“主教材”），故与主教材配套使用，效果更佳。在详解案例的基础上，还设计了两套实验——**基础实验**和**进阶实验**，供读者自己思考并练习，从而达到全面提高读者分析问题、解决问题能力的目的。

后续每一章都会给出该章实验题目及涉及知识的列表，随后的内容基本上分为两大部分：

1. **案例**及详细分析与解答。包括对实验内容和输入输出的明确要求以及一套测试样例；对问题的分析以及编程实现的要点；给出完整的解题源代码；最后在此题基础上给出实验思考题供读者做进一步深入的思考。

2. **实验**及分析与提示。同样包括对实验内容和输入输出的明确要求以及一套测试样例；与案例的指导不同的是，这里只比较详细地给出解题思路，对问题的分析以及编程实现的要点，而把具体的编程实现留给读者自己去完成；最后也是以若干实验思考题结尾。注意：实验又分**基础**和**进阶**两个层次，适用于不同的学习者群体。

全部**案例**和**实验**可于在线系统**拼题A**(网址 https://pintia.cn/)找到,该系统全天候为广大读者提供免费服务,读者可自行注册账号,随时提交测试自己的解决方案。本实验教材中大部分**案例**为主教材的书后习题,收录在主教材题目集(浙大版《数据结构(第2版)》题目集)中,不在本实验教材题目集(浙大版《数据结构学习与实验指导(第2版)》题目集)中重复收录。

本书配套的资源中,提供了书中程序的电子版作为案例习题的参考答案,读者可以直接提交参考答案,观察其时间、空间效率,并尝试自己做局部优化,以得到更高效的解决方案。

此外,部分章的进阶实验中有一些开放性题目。开放性题目一般有多种解决方案,且有可能涉及海量数据,不适合使用自动判题系统评判。有兴趣的读者可以自行设计测试方案。

第1章

算法与复杂度

本章实验内容主要围绕二分查找算法的实现及其应用，进行时空复杂度分析与比较，包括1个案例和基础实验、进阶实验各1项。这些题目涉及的知识内容如表1.1所示。

表1.1　本章实验涉及的知识点

序号	题目名称	类别	内容	涉及的主要知识点
1-1.1	二分查找	案例	分别用递归和非递归方法实现二分查找	时空复杂度分析与比较
1-2.1	有序数组的插入	基础实验	将给定整数插入有序数组，使得结果依然有序	二分查找的应用、时空复杂度分析
1-3.1	两个有序序列的中位数	进阶实验	求两个等长的非降序序列的中位数	二分查找的应用、时空复杂度分析

案例1-1.1：二分查找（主教材习题1.8）

1. 实验目的

（1）熟练掌握一般时空复杂度分析技巧。

（2）熟练掌握递归程序的时空复杂度分析技巧。

2. 实验内容

查找算法中的“二分法”是这样定义的：给定N个从小到大排好序的整数序列Data[ ]以及某待查找整数X，目标是找到X在Data[ ]中的下标。即若有Data[i]=X，则返回i；否则返回失败标记NotFound表示没有找到。二分法是先找到序列的中点Data[Mid]，与X进行比较，若相等则返回中点下标Mid；否则，若Data[Mid]>X，则在左边的子系列中查找X；若

Data[Mid]<X,则在右边的子系列中查找X。试用一个函数实现二分查找的功能,并分析最坏、最好情况下的时间、空间复杂度。

3. 实验要求

(1) 函数接口说明:

```
Position BinarySearch(List L, ElementType X);
```

其中List结构定义如下:

```
typedef int Position;
typedef struct LNode *List;
struct LNode {
    ElementType Data[MAXSIZE];
    Position Last;/* 保存数组 Data[] 中最后一个元素的位置 */
};
```

L是用户传入的一个线性表,其中ElementType元素可以通过>、==、<进行比较,并且题目保证传入的数据是递增有序的。函数BinarySearch要查找X在Data[ ]中的位置,即数组下标(注意:元素从下标1开始存储)。找到则返回下标,否则返回一个特殊的失败标记NotFound。

(2) 测试用例:

序号	传入参数值			返回	说明
	Data	Last	X		
0	12 31 55 89 101	5	31	2	小规模一般情况
1	26 78 233	3	31	NotFound	查找失败
2	11 15 18 22 23	5	18	3	奇数个,正中间找到
3	2 3 5 8 12 25	6	5	3	偶数个,正中间找到
4	略	$>10^4$	略	1	大数据,在头部找到
5	略	$>10^4$	略	Last的值	大数据,在尾部找到
6	略	$>10^4$	略	NotFound	大数据,查找失败

4. 实验分析

(1) 问题分析

虽然在原始的二分法描述中,给出的问题背景是在“整数序列”中查找,但事实上二分法适用于各种元素序列的查找,只要这些元素是可以比较大小的。例如实数序列、字符串序列等等。所以这里给出了一个更为抽象,即更为通用的函数接口,元素类型ElementType可以由用户根据自己的需要定义为任何具体的数据类型。

方法一：递归实现

根据题目中的算法描述，很显然递归可以给出最直接的实现，先用伪码来描述二分法的递归实现：

```
Position 二分法 (List L, ElementType X, Position Left, Position Right)
{/* Left 和 Right 分别是当前要处理的 L->Data[] 中最左和最右的下标值 */
   Mid =(Left + Right)/ 2;/* 计算中间元素坐标 */
   if(L->Data[Mid] > X)
     return 二分法 (L, X, Left, Mid-1);/* 在左边的子系列中查找 */
   else if(L->Data[Mid] < X)
     return 二分法 (L, X, Mid+1, Right);/* 在右边的子系列中查找 */
   else /* L->Data[Mid] == X */
     return Mid;
}
```

上面的伪码是算法的直接翻译，但是并不完全正确，因为缺少了一种特殊情况的处理：如果 X 根本不在 Data[ ]中，怎样才能知道呢？

二分法的查找过程就是不断将搜索区段折半，缩小搜索范围的过程。当范围内一个元素都没有了，X 还没有找到，就说明它不存在了。而“一个元素都没有”用程序语言来表达，就是 Left>Right，即左右两端的下标值错位。所以在上述伪码的最开始，应该加入一个判断，如果传入的左右两端的下标值错位，就应该返回 NotFound。完整的实现请见代码 1.1。

二分法的最好情况是 X 正好位于中间 Mid 位置上，只要 1 次查找就找到了，时间和空间复杂度显然都是 $O(1)$。最坏情况是使得折半的次数最大，即 X 根本不在 Data[ ]中，这时的复杂度分析略复杂。

分析递归函数时间复杂度的技巧是，首先假设当前要处理的数据规模是 N，对应的时间复杂度与当前规模有关，记为 $T(N)$。如果递归调用的子问题的规模为 cN（其中 c 是常数），则 $T(N)$就等于递归调用花费的时间 $T(cN)$和其他非递归处理步骤所花费的时间的总和。就二分法而言，每次递归调用时，问题的规模都会减半，即变为 $N/2$，此外其他步骤可以用常数时间完成。所以有递推公式：$T(N)=T(N/2)+c$（其中 c 是常数）。根据这个公式，很容易推出：

$$T(N)=T(N/2)+c=T(N/2^2)+2c=\cdots=T(N/2^k)+kc \qquad (公式\ 1.1)$$

这里的 k 就是递归的次数。在最坏情况下，必须一直递归到搜索范围中没有元素，即 $k=\log_2 N+1=O(\log N)$。代入公式 1.1，就得到 $T(N)=O(\log N)$。

关于空间复杂度，由于每次递归只是传入 List 结构的指针，并不在函数内部复制整个数组 Data[ ]，所以函数内部使用的空间只是个常量。但是递归会占用系统堆栈，空间复杂度是跟递归的次数成正比的。从时间复杂度的分析中，已经知道最坏情况下递归的次数是 $O(\log N)$，所以递归实现的空间复杂度就是 $S(N)=O(\log N)$。

微视频1-1：递归对系统空间的占用

递归实现算法一般代码较为简单，比较容易理解；缺点是当数据规模较大、递归层次较深时，有可能过多占用系统空间。所以对于如二分法这么简单的算法，人们总是尽可能地采用非递归的方法实现。

方法二：非递归实现

把“缩小查找范围”这个动作理解为“调整左边或右边的边界”，就可以用一个 while 循环解决问题。从初始状态的左边界为 1、右边界为 Last 开始。每次循环中，如果 X 有可能在左半边，就保持左边界不动，把右边界移到 Mid 左边；如果 X 有可能在右半边，就保持右边界不动，把左边界移到 Mid 右边。如此反复直到找到 X 返回 Mid，或者直到范围内一个元素都没有了，就跳出循环返回 NotFound。完整的实现代码请见代码 1.2。

注意到函数传入的参数仍然只是一个 List 指针，并不在函数内部复制整个数组 Data[ ]，所以函数内部使用的空间一直都是个常量，即 $S(N)=O(1)$。这是比递归实现占优的地方。

考虑时间复杂度时，注意整个函数的运行时间取决于 while 循环的次数。最好情况下，X 正好位于中间 Mid 位置上，只要 1 次循环就找到了，时间复杂度是 $O(1)$。最坏情况是一直循环到 Left>Right，发现 X 找不到。这时的循环次数也就是将查找范围折半的最大次数，与递归次数分析同理，就是 $O(\log N)$。所以与递归实现一样，也有 $T(N)=O(\log N)$。

（2）实现要点

由于递归函数需要传入当前搜索范围的左右边界下标值，所以原始的函数接口不能直接用于递归。为此需要另外实现一个递归函数，并在原始函数中正确调用递归函数：在初始状态下，搜索范围是整个 Data[ ]，所以最左边的下标是 1，最右边的下标就是 Last 的值。

另外特别需要注意的是，在缩小查找范围时，一定不要把被调整的边界值调整为 Mid，必须调整为 Mid 左边或者右边的一个下标值。

5. 实验参考代码

```
Position BS(List L, ElementType X, Position Left, Position Right)
{ /* Left 和 Right 分别是当前要处理的 L->Data[] 中最左和最右的下标值 */

   if(Left > Right)/* 如果当前范围内没有元素了 */
      return NotFound;/* 返回查找不成功的标识 */

   Mid =(Left + Right)/ 2;/* 计算中间元素坐标 */
   if(L->Data[Mid] > X)
      return BS(L, X, Left, Mid-1);/* 在左边的子系列中查找 */
   else if(L->Data[Mid] < X)
      return BS(L, X, Mid+1, Right);/* 在右边的子系列中查找 */
```