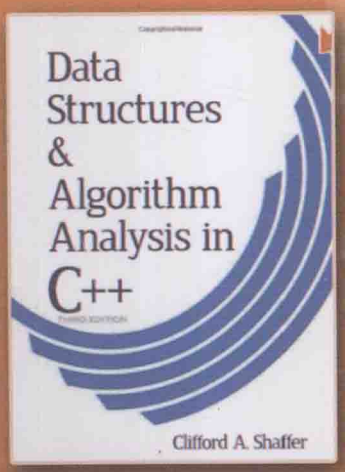
国外计算机科学教材系列

数据结构与算法分析

（C++版）（第三版）

Data Structures and Algorithm Analysis in C++

Third Edition



[美] Clifford A. Shaffer 著

张 铭 刘晓丹 等译

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

国外计算机科学教材系列

数据结构与算法分析

(C++版)(第三版)

Data Structures and Algorithm Analysis in C++

Third Edition

[美] Clifford A. Shaffer 著

张 铭 刘晓丹 等译

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

本书采用程序员广泛使用的面向对象C++语言来描述数据结构和算法，并把数据结构原理和算法分析技术有机地结合在一起，系统介绍了各种类型的数据结构及排序、检索的各种方法。作者非常注意对每一种数据结构的不同存储方法及有关算法进行分析比较。书中还引入了一些比较高级的数据结构与先进的算法分析技术，并介绍了可计算性理论的一般知识。书中分别给出了C++实现方法和伪码实现方法，便于读者根据情况进行选择。可从本书作者维护的网站上下载相关代码、编程项目和辅助练习资料。

本书适合作为大专院校计算机软件专业与计算机应用专业学生的教材和参考书，也适合计算机工程技术人员参考。

ISBN13：978-0-486-48582-9

ISBN10：0-486-48582-X

Dover出版社所出版的这个版本首印于2011年，是对 A Practical Introduction to Data Structures and Algorithm Analysis的第三版（可从作者网站 http://people. csvt. edu/~ shaffer/Book/下载）进行过修订与更新后首次以纸质书形式出版。

版权贸易合同登记号　图字：01-2012-7753

图书在版编目（CIP）数据

数据结构与算法分析：C++版：第3版/（美）谢弗（Shaffer，C. A.）著；张铭等译.

北京：电子工业出版社，2013.10

（国外计算机科学教材系列）

书名原文：Data Structures and Algorithm Analysis in C++ , Third Edition

ISBN 978-7-121-20958-1

Ⅰ. ①数…　Ⅱ. ①谢…②张…　Ⅲ. ①数据结构-高等学校-教材②算法分析-高等学校-教材　Ⅳ. ①TP311.12

中国版本图书馆CIP数据核字（2013）第150575号

策划编辑：马　岚

责任编辑：冯小贝

印　　刷：涿州市京南印刷厂

装　　订：涿州市京南印刷厂

出版发行：电子工业出版社

　　北京市海淀区万寿路173信箱　邮编　100036

开　　本：787×1092　1/16　印张：25.5　字数：703千字

印　　次：2013年10月第1次印刷

定　　价：59.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

服务热线：（010）88258888。

译 者 序

数据结构与算法分析是计算机专业十分重要的一门基础课，计算机科学各个领域及各种应用软件都要使用相关的数据结构和算法。

当面临一个新的设计问题时，设计者需要选择适当的数据结构，并设计出满足一定时间和空间限制的有效算法。本书作者把数据结构和算法分析有机地结合在一本教材中，有助于读者根据问题的性质选择合理的数据结构，并对算法的时间、空间复杂性进行必要的控制。

本书采用当前流行的面向对象C++程序设计语言来描述数据结构和算法，因为C++语言是程序员最广泛使用的语言。因此，程序员可以把本书中的许多算法直接应用于将来的实际项目中。尽管数据结构和算法在设计本质上还是很底层的东西，并不像大型软件工程项目开发那样，对面向对象方法具有直接的依赖性，因此有人会认为并不需要采用高层次的面向对象技术来描述底层算法。但是，采用C++语言能更好地体现抽象数据类型的概念，从而更本质地描述数据结构和算法。为了使本书清晰易懂，作者有意回避了C++的某些重要特性。

本书正文包括五部分的内容，第一部分是预备知识，介绍了一些基本概念和术语，以及基础数学知识。在本书的改版中，作者加强了面向对象的讨论，特别是增加了设计模式的相关内容，例如享元、访问者、组合和策略等设计模式。设计模式像模板那样描述了一种解决方案的框架及具体实践，又有类似于数据结构的代价和收益，需要根据不同的应用场景做出权衡。

第二部分介绍了最基本的数据结构，依次为线性表(包括栈和队列)、二叉树和树。对每种数据结构的讲解都从其数学特性入手，先介绍抽象数据类型，然后再讨论不同的存储方法，并且研究不同存储方法的可能算法。值得赞赏的是，作者结合算法分析来讨论各种存储方法和算法的利弊，摒弃那些不适宜的方法，这样就调动了读者的思维，使其可以从中学到考虑问题的方法。这种“授人以渔”的策略使读者在今后设计和应用数据结构时能全面地考虑各种因素，选择最佳方案。

作为最常用的算法，排序和检索历来都是数据结构讨论的重点问题。这在第三部分的第7～10章中进行了详尽的讨论。排序算法最能体现算法分析的魅力，它的算法速度要求非常高：其中内排序主要考虑的是怎样减少关键码之间的比较次数和记录交换次数，以提高排序速度；而外排序则考虑外存的特性，尽量减少访问操作，以提高排序速度。第7章证明了所有基于比较的排序算法的时间代价是$\Theta(n \log n)$，这也是排序问题的时间代价。检索则考虑怎样提高检索速度，这往往与存储方法有关。书中介绍了几种高效的数据结构，如自组织线性表、散列表、B树和B^+树等，都具有极好的检索性能。

第四部分介绍了数据结构的应用与一些高级主题，其中包括图、跳跃表、广义表和稀疏矩阵等更复杂的线性表结构，还包括了Trie结构、AVL树等复杂树结构，以及k-d树、PR四分树等空间数据结构。

本书第三版在第五部分(第14～17章)以较大的篇幅新增了内容，从而加强了算法分析

方面的讨论。这一部分首先介绍了求和、递归关系分析和均摊分析等算法分析技术，这些技术对于提高程序员的算法分析能力具有重要作用。然后，讨论算法和状态空间下限的概念与实例，并介绍了对抗性下限证明方法。本书系统地介绍了重要的算法模式，包括动态规划、随机算法和变换，并介绍了傅里叶变换等数值算法。最后讨论了计算复杂性理论中的难解问题，利用归约把各种问题的难度联系起来。

本书的前言及第1～10章由张铭翻译，第11～17章由刘晓丹翻译。另外，肖之屏、刘智冲、方译萌、王子琪、王晟、盛达魁、刘金宝、贺一骏、桂欢、荣小松、陈云帆、张亦弛、刘卢琛、王卓等人也参加了本书的翻译工作，在此对他们的辛勤劳动表示感谢。由于译者水平有限，书中难免有不妥之处，欢迎读者批评指正。

前　言

人们研究数据结构的目的，是为了学会编写效率更高的程序。现在的计算机速度一年比一年快，为什么还需要高效率的程序呢？这是由于人类解决问题的雄心与能力是同步增长的。现代计算技术在计算能力和存储容量上的革命，仅仅提供了解决更复杂问题的有效工具，而对程序高效率的要求永远也不会过时。

程序高效率的要求不会也不应该与合理的设计和简明清晰的编码相矛盾。高效率程序的设计基于良好的信息组织和优秀的算法，而不是基于“编程小伎俩”。一名程序员如果没有掌握设计简明清晰程序的基本原理，就不可能编写出有效的程序。反过来说，对开发代价和可维护性的考虑不应该作为性能不高的借口。设计中的通用准则(generality in design)应该在不牺牲性能的情况下达到，但前提是设计人员知道如何去衡量性能，并且把性能作为设计和实现不可分割的一部分。大多数计算机科学系的课程设置都意识到要培养良好的程序设计技能，首先应该强调基本的软件工程原理。因此，一旦程序员学会了设计和实现简明清晰程序的原理，下一步就应该学习有效的数据组织和算法，以提高程序的效率。

途径：本书描述了许多表示数据的技术。这些技术包括以下原则：

1. 每一种数据结构和每一个算法都有其时间、空间的代价和效率。当面临一个新的设计问题时，设计者要透彻地掌握权衡时间、空间代价和算法有效性的方法，以适应问题的需要。这就需要懂得算法分析原理，而且还需要了解所使用的物理介质的特性(例如，当数据存储在磁盘上与存储在主存中时就有不同的考虑)。
2. 与代价和效率有关的是时空权衡。例如，人们通常增加空间代价来减少运行时间，或者反之。程序员所面对的时空权衡问题普遍存在于软件设计和实现的各个阶段，因此这个概念必须牢记于心。
3. 程序员应该充分了解一些现成的方法，以免做不必要的重复开发工作。因此，学生们需要了解经常使用的数据结构和相关算法，以及程序中常见的设计模式。
4. 数据结构服从于应用需求。学生们必须把分析应用需求放在第一位，然后再寻找一个与实际应用相匹配的数据结构。要做到这一点，需要应用上述三条原则。

笔者讲授数据结构多年，发现设计在课程中起到了非常重要的作用。本教材的几个版本中逐步增加了设计模式和接口。本书第一版完全没有提到设计模式。第二版有一些篇幅讲解了几个设计模式的例子，并且介绍了字典 ADT 和比较器类。编写本书第三版的基本数据结构和算法时，都直接介绍了一些相关的设计模式。

教学建议：数据结构和算法设计的书籍往往囿于下面这两种情形之一：一种是教材，一种是百科全书。有的书籍试图融合这两种编排，但通常是二者都没有组织好。本书是作为教材来编写的。我相信，了解如何选择或设计解决问题的高效数据结构的基本原理是十分重要的，这比死记硬背书本内容重要得多。因此，本书中涵盖了大多数、但不是全部的标准数据

结构。为了阐述一些重要原理，也包括了某些并非广泛使用的数据结构。另外，还介绍了一些相对较新、但即将得到广泛应用的数据结构。

在本科教学体系中，本书适用于低年级(二年级或三年级)的高级数据结构课程或者高年级的算法课程。第三版中加入了很多新的素材。通常，这本书被用来讲授一些超过常规一年级的 CS2 课程，也可作为基础数据结构的介绍。读者应该已有两个学期的基本编程经验，并具备一些C++ 基础技能。对已经熟悉部分内容的读者会有一些优势。学习数据结构的学生如果先学完离散数学课程，也颇有益处。不过，第 2 章还是给出了比较完整的数学预备知识，这些知识对理解本书的内容还是很有必要的。读者如果在阅读中遇到不熟悉的知识，可以回头看看相应的章节。

大二学生掌握的基本数据结构和算法分析的背景知识(相对于从传统 CS2 课程中获得的基础知识)并不太多，可以对他们详细地讲解第 1 ~ 11 章的内容，再从第 13 章选择一些专题来讲解，我就是这样来给二年级学生讲课的。背景知识更丰富的学生，可以先阅读第 1 章，跳过第 2 章中除参考书目之外的内容，简要地浏览第 3 章和第 4 章，然后详细阅读第 5 ~ 12 章。另外，教师可以根据程序设计实习的需要，选择第 13 章以后的某些专题内容。高年级的算法课程可以着重讲解第 11 章和第 14 ~ 17 章。

第 13 章是针对大型程序设计练习而编写的。我建议所有选修数据结构的学生，都应该做一些高级树结构或其他较复杂的动态数据结构的上机实习，例如第 12 章中的跳跃表(skip list)或稀疏矩阵。所有这些数据结构都不会比二叉检索树更难，而且学完第 5 章的学生都有能力来实现它们。

我尽量合理地安排章节顺序。教师可以根据需要自由地重新组织内容。读者掌握了第 1 ~6 章后，后续章节的内容就相对独立了。显然，外排序依赖于内排序和磁盘文件系统。Kruskal 最小支撑树算法使用了 6.2 节关于并查(UNION/FIND)的算法。9.2 节的自组织线性表提到了 8.3 节讨论的缓冲区置换技术。第 14 章的讨论基于本书的例题。17.2 节依赖于图论知识。在一般情况下，大多数主题都只依赖于同一章中讨论的内容。

几乎每一章都是以“深入学习导读”一节结束的。它并不是这一章的综合参考索引，而是为了通过这些导读书籍或文章提供给读者更广泛的信息和乐趣。在有些情况下我还提供了作为计算机科学家应该知道的重要背景文章。

关于C++：本书中的所有示例程序都是用C++ 编写的，但是我并不想难倒那些对C++ 不熟悉的读者。在努力保持C++ 优点的同时，使示例程序尽量简明、清晰。C++ 在本书中只是作为阐释数据结构的工具。此外，特别用到了C++ 隐藏实现细节的特性，例如类(class)、私有类成员(private class member)、构造函数(constructor)、析构函数(destructor)。这些特性支持了一个关键概念：体现于抽象数据类型(abstract data type)中的逻辑设计与体现于数据结构中的物理实现相分离。

为了使得本书清晰易懂，我回避了C++ 中的某些最重要特性。书中有意排除或尽量少使用一些特性，而这些特性是经验丰富的C++ 程序员经常使用的，例如类的层次(class hierarchy)、继承(inheritance)和虚函数(virtual function)，运算符和函数的重载(operator and function overloading)也很少使用。在很多情况下，更倾向于使用 C 的原始语义，而不是C++ 所提供的一些类似功能。

当然，上述C++ 的特性在实际程序中是合理的程序设计基础，但是它们只能掩盖而不是

加强本书所阐述的原理。例如，对于程序员来说，类的继承在避免重复编码和降低程序错误率方面是一个很重要的工具；但是从教学标准的观点来看，类的继承在若干类中分散了单个逻辑单元的描述，从而使程序更难理解。因此，仅当类的继承对阐述文章的观点有明显作用时，我才使用它(见5.3.1节)。避免代码重复和减少错误是很重要的目标，请不要把本书中的示例程序直接复制到自己的程序中，而只是把它们看成对数据结构原理的阐释。

一个痛苦的选择是，在示例代码中是否使用模板(template)。在编写本书的第一版时，我决定不使用模板，因为考虑到它们的语义对于不熟悉C++语言的人来说掩盖了代码的含义。在随后几年中，使用C++的计算机科学课程急剧地增加了，因此我假设现在的读者比以前的读者更熟悉模板的语义。因此本书在示例代码中大量使用了模板。

本书中的C++程序提供了有关数据结构原理的真实阐释，是对文字阐述的补充。不宜脱离相关文字阐述而孤立地阅读或使用示例程序，因为大量的背景信息包含在文字阐述中，而不是包含在代码中。代码是对文字阐述的完善，而不是相反。这不是一系列具有商业质量的类的实现。如果读者想寻找一些标准的数据结构的完整实现，或者要在你的代码中使用这些数据结构，那么应该在Internet上寻找。

例如，这些例子中所做的参数检查，比起商业软件要少得多，因为这种检查将降低算法的清晰度。某些参数检查和约束检查(例如是否从一个空容器中删除值)是以调用**Assert**的形式完成的。**Assert**的输入是一个布尔表达式，一旦这个表达式的值为假(**false**)，程序就立即终止。函数遇到一个坏参数就终止程序，这在真实程序中通常是不必要的，但有益于理解一个数据结构是怎样工作的。在实际的程序应用中，C++异常处理机制(exception handling features)用来处理一些输入数据错误。然而，**Assert**提供了一种机制，既有益于阐明一个数据结构的工作条件，也有利于采用真正的异常处理机制来代替。读者可参阅附录A中的**Assert**实现。

在示例程序中，我严格区分了“C++实现”和“伪码”(pseudocode)。一个标明“C++实现”的示例程序在一个以上的编译器中被真正编译过。伪码的示例通常具有与C++接近的语法，但是一般包含一行以上更高级的描述。当我发现简单的、尽管并不十分精确的描述具有更好的教学效果时，就使用伪码。

习题和项目设计：只靠读书是不能学会灵活使用数据结构的。一定要编写实际的程序，比较不同的数据结构技术，从而观察在一种给定的条件下哪一种数据结构更有效。

数据结构课程最重要的教学安排之一，就是学生应该在什么时候开始学习使用指针和动态存储分配，从而编程实现链表和树这样的数据结构。这也是学生们学习递归的时机。在教学体系中，这是学生学习重大设计(significant design)的第一门课，因为通常需要使用真实数据结构来引出重大设计练习。最后，基于内存和基于硬盘的数据访问的本质区别，必须要在编程实践中才能理解。基于以上原因，一门数据结构课程没有大量的编程环节是不能成功的。在计算机系，数据结构是课程计划中最难的一门编程课程。

学生还需要在解决问题中锻炼分析能力。本书提供了450个习题和编程项目，希望读者能够好好利用它们。

与作者联系的方法及相关资料的获取：本书难免有一些错误，有些方面还有待进一步研究。非常欢迎读者指正，并提出建设性意见。作者的E-mail地址是**shaffer@ vt.edu**，也可以给以下的地址写信：

Cliff Shaffer
Department of Computer Science
Virginia Tech
Blacksburg, VA 24061

本书的电子版和上课中使用的一些幻灯片材料，可以从以下网站获取：

http://www.cs.vt.edu/~shaffer/book.html

示例代码也可以从上面的网站得到。弗吉尼亚理工大学二年级数据结构课程网页的 URL 为

http://courses.cs.vt.edu/~cs3114

致谢：本书得到了许多友人的帮助。我想特别感谢其中的几位，他们对本书的出版贡献最大。对于没有被提及的朋友，在此表示歉意。

弗吉尼亚理工大学在 1994 年秋季的学术休假中使得整个出书的事情成为可能，我是从那时开始着手准备的。在编写这本书的第二版时，系主任 Dennis Kafura 和 Jack Carroll 对本书给予了重要的精神支持。Mike Keenan、Lenny Heath 和 Jeff Shaffer 对本书最初版本的内容提供了有价值的意见。尤其是 Lenny Heath 多年来一直与我深入地讨论算法设计和分析的有关问题，以及怎样把二者讲授给学生的方法。十分感谢 Steve Edwards 花了很多时间帮助我几次重写了第二版、第三版的C++ 代码和 Java 代码，并与我讨论程序设计的原则。Layne Watson 提供了有关 Mathematica 软件的帮助，Bo Begole、Philip Isenhour、Jeff Nielsen 和 Craig Struble 提供了一些技术上的帮助。感谢 Bill McQuain、Mark Abrams 和 Dennis Kafura 回答了一些有关C++ 和 Java 的问题。

对于许多评阅了本书手稿的朋友，本人欠情甚深。这些评阅者是：J. David Bezek（伊凡斯维尔大学），Douglas Campbell（杨百翰大学），Karen Davis（辛辛那提大学），Vijay Kumar Garg（得克萨斯大学奥斯汀分校），Jim Miller（堪萨斯大学），Bruce Maxim（密歇根大学迪尔本分校），Jeff Parker（Agile Networks/Harvard），Dana Richards（乔治梅森大学），Jack Tan（休斯敦大学）和 Lixin Tao（加拿大肯高迪亚大学）。要不是他们的热心帮助，本书会出现更多技术上的错误，内容也将更加浅显。

关于这本第二版的出版，我想感谢下列评阅者：Gurdip Singh（堪萨斯州立大学），Peter Allen（哥伦比亚大学），Robin Hill（怀俄明大学），Norman Jacobson（加州大学欧文分校），Ben Keller（东密歇根大学）和 Ken Bosworth（爱达荷州立大学）。另外，我要感谢 Neil Stewart 和 Frank J. Thesen 对改进本书提出的意见和建议。

第三版的评阅者包括 Randall Lechlitner（休斯敦大学 Clear Lake 分校）和 Brian C. Hipp（约克技术学院）。感谢他们的建议。

Prentice Hall 是本书第一版和第二版的出版方。没有出版社众多朋友的帮助，不可能有本书的出版，因为作者不可能自己印出书来。因此，几年来我要感谢 Kate Hargett、Petra Rector、Laura Steele 和 Alan Apt 这几位编辑。感谢本书第二版的责任编辑 Irwin Zucker，还有本书C++ 版的责任编辑 Kathleen Caren 和 Java 版的责任编辑 Ed DeFelippis，他们在本书接近出版的最忙乱的日子里，保持各个方面运作良好。感谢 Bill Zobrist 和 Bruce Gregory 使我着手此事。感谢 Prentice Hall 的 Truly Donovan、Linda Behrens 和 Phyllis Bregman 在本书出版过程中给予的帮助。感谢 Tracy Dunkelberger 在交回版权给我时提供的帮助。可能还有许多没有被提及的 Prentice Hall 出版社的朋友，他们也默默地提供了帮助。

本人非常感谢 Dover 出版社的 Shelley Kronzek 在第三版的出版过程中付出的一切。第三版中有许多扩展，包括 Java 和C++ 代码，以及一些改正。我相信这是迄今为止最好的一版。但是不知道学生会不会希望有一本免费的在线教材，或是低价的印刷版本。最终，我相信两个版本会提供更多的选择。责任编辑 James Miller 和设计经理 Marie Zaczkiewicz 为确保本书的高质量出版付出了辛勤的工作。

我十分感激 Hanan Samet 传授给我有关数据结构的知识，我从他那里学到了许多原理与知识，当然本书中可能出现的错误并不是他的责任。感谢我的妻子 Terry 对我的爱与支持，还有两个女儿 Irena 和 Kate 带给我的欢乐，可以让我从艰苦的工作中解脱出来。最后，也是最重要的，感谢这些年来选修数据结构的学生，是他们使我知道了在数据结构课程中什么是重要的而什么应该忽略，许多深入的问题也是他们提供的。谨以此书献给他们。

Clifford Shaffer

Blacksburg, Virginia

目　录

第一部分　预 备 知 识

第二部分　基本数据结构

第三部分　排序与检索

第四部分 高级数据结构

第五部分 算 法 理 论

第六部分 附 录

第一部分

预 备 知 识

第1章　数据结构和算法

得克萨斯州达拉斯市方圆500英里[①]之内有多少个人口超过250 000人的城市？一个公司里有多少人年收入超过100 000美元？用不超过1000英里长的电缆是否能把所有电话用户都连起来？现在就要正确回答类似这样的问题是不可能的，因为这里还没有给出回答这些问题所必须要了解的信息。必须以能及时找到答案的方式来有效地组织信息，才能回答上述问题。

信息表示是计算机科学的基础。大多数计算机程序的主要目标与其说是完成运算，倒不如说是存储信息和尽快地检索信息。因此，研究数据结构和算法就成为了计算机科学的核心问题。本书的目的就是帮助读者理解怎样组织信息，以便支持高效的数据处理。

本书有三个主要目的。第一个目的是介绍常用的数据结构，这些数据结构形成了一个程序员基本数据结构工具箱(toolkit)。对于许多问题，工具箱里的数据结构是理想的选择。

第二个目的是引入并加强“权衡”(tradeoff)的概念，每一个数据结构都有相关的代价和效率的权衡。本书通过列举出不同的数据结构，并指出它们应用于实际问题时的代价和效率来讨论“权衡”的概念。

第三个目的是讲解如何评估一个数据结构或算法的有效性。只有通过这样的分析，才能确定对一个新问题最合适的数据结构是工具箱中的哪一个。这种技术也使得程序员能够判断自己或别人发明的新数据结构的价值。

一个问题通常有多种解法，应该选择哪一种呢？计算机程序设计的核心有两个目标(有时它们互相冲突)：

1. 设计一个容易理解、编码和调试的算法；
2. 设计一个能有效利用计算机资源的算法。

在理想情况下，最终的程序都能实现这两个目标，有时把这样的程序称为是“完美的”。本书给出的算法和程序实例在这种意义上来说是趋于完美的。与目标1有关的问题不是本书的目的，它们主要涉及的是软件工程原理。而本书主要讲的是与目标2有关的问题。

怎样度量效率呢？第3章将给出估算一个算法或者一个计算机程序效率的方法，称为算法分析(algorithm analysis)。算法分析还可以度量一个问题的内在复杂程度。以后的章节中对算法的时间代价进行衡量时，都会用到算法分析方法。利用这种方法可以清楚地看到在解决同一个问题时不同算法在效率上的差异。

这一章提出与选择和使用数据结构相关的话题，通过这些话题为后面的内容做准备。本章将先审视设计者为要完成的任务选择一个合适的数据结构所经历的过程，然后讨论在程序设计中如何进行抽象，最后将探索问题、算法和程序三者之间的关系。

---

① 1英里=1.609千米。