Broadview
www.broadview.com.cn

夜光

算法之美

隐匿在数据结构背后的原理

C++版

左飞 / 著

探秘算法世界
求索数据结构之道
汇集经典问题
畅享编程技法之趣
点拨求职热点
敲开业界名企之门

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

算法之美

隐匿在数据结构背后的原理

左飞 / 著

C++版

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

内 容 简 介

本书围绕算法与数据结构这个话题，循序渐进、深入浅出地介绍了现代计算机技术中常用的40余个经典算法，以及回溯法、分治法、贪婪法和动态规划等算法设计思想。在此过程中，本书也系统地讲解了链表（包括单向链表、单向循环链表和双向循环链表）、栈、队列（包括普通队列和优先级队列）、树（包括二叉树、哈夫曼树、堆、红黑树、AVL树和字典树）、图、集合（包括不相交集）与字典等常用数据结构。同时，通过对22个经典问题（包括约瑟夫环问题、汉诺塔问题、八皇后问题和骑士周游问题等）的讲解，逐步揭开隐匿在数据结构背后的算法原理，力图帮助读者夯实知识储备，激活思维技巧，并最终冲破阻碍编程能力提升的重重藩篱。

本书适合作为大专院校相关专业学生研习算法与数据结构知识的课外参考书。对有意参加信息学竞赛的读者，本书亦有很强的参考价值。此外，鉴于算法与数据结构在求职过程中常常被视为考察重点，所以就临近毕业的学生或其他欲从事IT行业的求职者而言，阅读本书也将对面试备考大有裨益。

图书在版编目（CIP）数据

算法之美：隐匿在数据结构背后的原理：C++版 / 左飞著. —北京：电子工业出版社，2016.3
ISBN 978-7-121-27718-4

Ⅰ. ①算… Ⅱ. ①左… Ⅲ. ①数据结构－程序设计②C语言－程序设计 Ⅳ. ①TP311.12②TP312

中国版本图书馆CIP数据核字(2015)第286959号

责任编辑：付　睿
印　　刷：北京京科印刷有限公司
装　　订：三河市皇庄路通装订厂
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：26.75　字数：617.6千字
版　　次：2016年3月第1版
印　　次：2016年3月第1次印刷
印　　数：3000册　定价：79.00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。

前言

2014 年的冬天，一部讲述计算机科学之父艾伦·图灵传奇人生的传记电影在美国上映，这部影片就是《模仿游戏》。次年，该片荣获第 87 届奥斯卡金像奖最佳改编剧本奖，以及包括最佳影片、最佳导演、最佳男主角、最佳女配角在内的 7 项提名，一时风光无限。尽管现代计算机已经无处不在，但因图灵的时代离我们过于久远，现今人们对他的研究工作已经知之甚少。

要说起图灵的贡献，我们还得把时间再往前推。1900 年，德国数学家大卫·希尔伯特在巴黎举行的国际数学家大会上做了题为《数学问题》的演讲，在这篇重要的演讲中，他提出了著名的希尔伯特之 23 个问题。尽管此后的数学发展远远超过了希尔伯特的预料，但他所提出的 23 个问题仍然对 20 世纪的数学发展起到了非常积极的推动作用。

希尔伯特的第 10 个问题是要设计一个算法来测试多项式是否有整数根。他没有使用算法这个术语，而是采用了下面这种表述："通过有限多次运算就可以决定的过程"。有意思的是，从希尔伯特对这个问题的陈述可以看出，他明确地要求设计一个算法。因此，他显然是假设这样的算法是存在的，人们所要做的只是找到它。现在我们知道，这个任务是无法完成的，即它是算法上不可解的。但对那个时期的数学家来说，以他们对算法的直观认识，得出这样的结论是不可能的。

非形式地说，算法是为实现某个任务而构造的简单指令集。以日常用语来说，算法又称为过程或者方法。算法在数学中也起着非常重要的作用。古代数学文献中就包含有执行各种各样计算任务的算法描述。例如，我国古代数学经典《九章算术》中就记述了包括求最大公约数、最小公倍数、开平方根、开立方根等在内的诸多算法。现代计算机科学中更是充满了各种各样的算法。例如，求解最短路径的狄克斯特拉算法，进行字符串匹配的 KMP 算法等。

虽然算法在数学中已有很长的历史，但在 20 世纪之前，算法概念本身一直没有精确的定义。数学家们面对希尔伯特的第 10 个问题，显得束手无策。由于缺乏对于算法本身的精确定义，所以要证明某个特定任务不存在算法则完全不可能。要想破解希尔伯特的第 10 个问题，人们不得不等待算法之精确定义的出现。

直到 1936 年，曙光似乎出现了。图灵向伦敦的权威数学杂志递交了一篇题为《论数字计算在决断难题中之应用》的论文。该文最终于 1937 年正式发表，并立即引起了广泛的注意。在论文中，图灵描述了一种可以辅助数学研究的机器，也就是后来被称为"图

灵机”的抽象系统。与此同时，另外一位数学家阿隆佐·丘奇也独立地提出了另外一套系统，即所谓的λ演算。图灵采用他的图灵机来定义算法，而丘奇则采用λ演算来定义算法，后来图灵证明这两个定义是等价的。由此，人们在算法的非形式概念和精确定义之间建立了联系，即算法的直觉概念等价于图灵机算法，这就是所谓的丘奇－图灵论题。

丘奇－图灵论题提出的算法定义是解决希尔伯特第 10 个问题所必需的。而第 10 个问题的真正解决则要等到 1970 年，借助于丘奇与图灵的杰出贡献，马提亚塞维齐在戴维斯、普特纳姆和罗宾逊等人工作的基础上，最终证明检查多项式是否有整数根的算法是不存在的。

从图灵开始，算法已然同计算机科学之间产生了密不可分的联系。当然，本书的内容并不打算从图灵机开始讲起。回顾建立算法形式化定义和破解希尔伯特第 10 个问题的那段风起云涌的历史，更多地是想说明算法之于我们的世界是多么重要。

无论你是信息技术的从业人员，还是计算机专业的在校学生，再或者是从事相关专业的研究人员，熟练掌握一门计算机语言的重要性都不言而喻。但是不是掌握了这其中的语法规则就能写出漂亮的程序了呢？答案当然是否定的。因为你还需要另外一样至少同等重要的工具——算法。算法和语言的关系，其实很像是“道”和“术”的关系。掌握一门语言，就如同习得一门技艺，可以成为一名工匠。但要想从工匠一跃成为大师，单单停留在“术”的层面显然不够，更重要的是悟“道”。而算法无疑就是计算机程序设计中的“道”。

谈到算法的重要性就不得不提及计算机科学家安德鲁·艾派尔在 1985 年所开展的一项研究工作，这也是程序性能优化领域的经典案例。彼时，艾派尔编写了一个用于计算重力场中天体间相互作用之问题的程序。给定场中物体质量、初始位置和速度等条件，该程序即可对 10000 个天体相互作用时其中两个天体的运行状态进行模拟和仿真。由于计算量太大，最初的程序要完成该项计算大约需要耗时一年。在一系列的改进之后，艾派尔最终将程序耗时有效地缩短到了一天！而在这个改进过程中，算法和数据结构的调优占了主要比重。

再说一个发生在笔者身上的例子。曾经在上学的时候，老师布置了一道编程作业，用于模拟一个猜三和弦的游戏。一个三和弦是指从 A、B、C、D、E、F、G 这 7 个音中任选 3 个组成的一个旋律，而每个音又有高音、中音、低音 3 种情况（分别用 1、2、3 来表示）。现在假设一名作曲家心中有了一个心仪的旋律，然后一个钢琴演奏者试图猜测这个答案。每当演奏者给出一个猜测，例如“A1、B2、C3”。那么作曲家将只能答复这其中完全猜中的音调（即音符和音高都猜对）有几个，除了完全猜中的音调以外，音符猜中了几个，音高猜中了几个。然后演奏者继续猜测，直到完全猜中为止。要知道，全部的组合可能有 1330 种之多！而我们希望用越少的次数猜中越好。不知道本书的各位读者心中是否已想到什么方法来解决这个问题。不过笔者最终实现的程序可以做到平均 4.2 次便猜中答案。而在这个过程中，设计一个绝佳的算法无疑是不二之选。

说起算法又不得不提及数据结构，二者是相辅相成、密不可分的。一方面，算法一定要借助相应的数据结构才能得以实现，另一方面我们在定义一个数据结构的同时其实也已经定义了与之相关的操作。这些操作本身执行的步骤就是算法。

总的来说，本书围绕算法与数据结构这个话题，循序渐进、深入浅出地介绍了现代计算机技术中常用的 40 余个经典算法（包括模式匹配算法、排序算法、散列算法、最短路径算法等），以及回溯法、分治法、贪婪法和动态规划等算法设计思想。本书也系统地讲解了链表（包括单向链表、单向循环链表和双向循环链表）、栈、队列（包括普通队列和优先级队列）、树（包括二叉树、哈夫曼树、堆、红黑树、AVL 树和字典树）、图、集合（包括不相交集等）与字典等常用数据结构。同时，通过对 22 个经典问题（包括约瑟夫环问题、汉诺塔问题、八皇后问题和骑士周游问题等）的讲解，逐步揭开隐匿在数据结构背后的算法原理，力图帮助读者夯实知识储备，激活思维技巧，并最终冲破阻碍编程能力提升的重重藩篱。

更值得一提的是，算法与数据结构知识是技术类求职过程中的必考内容。希望广大读者，尤其是处于求职应聘阶段的毕业生，在夯实基础、培养能力的同时，亦能设法将知识转化为生产力，求得一份称心如意的职位。若能事半功倍、一石二鸟，何乐而不为？为此，笔者特别在附录中整理出了一些求职面试中的经典题目，供有相关需求的读者参考学习。该套题目主要以算法与数据结构问题为主线，并穿插以 C/C++相关的编程问题，具有较高的实用性，对提高应聘竞争力很有帮助。特别地，在正文中涉及相关考点之处，笔者均采用旁注的形式点明了可以参考的题目编号，便于读者在阅读过程中，边学边练，知行合一。

纸上得来终觉浅，绝知此事要躬行。锤炼数据结构的运用能力和深化算法思想的理解程度都有赖于编程实践活动。本书采用 C++作为描述语言，并提供有涉及的全部数据结构和算法之实现代码，供读者参考学习。这些代码均在基于 TDM-GCC 4.9.2 的 DEV-C++ 5.11 和 Visual Studio 2013 环境下编译通过。本书特别提供了一个在线支持资源，地址 http://blog.csdn.net/baimafujinji，从中读者可以下载得到全书的配套代码和附录题库的参考答案，本书的勘误也将实时发布在此博客上。同时欢迎读者就本书中的问题和不足与笔者展开讨论，有关问题请在上述博客中留言。

最后，刘航、吴凯、姜萌、何鹏、胡俊、李召恒、初甲林等人也参与了本书编写工作，笔者在此表示由衷的感谢。

自知论道须思量，几度无眠一文章。由于时间和能力有限，书中纰漏在所难免，真诚地希望各位读者和专家不吝批评斧正。

目录

45 个算法

22 个经典问题

第1章

从数据到算法

欢迎学习数据结构与算法。数据结构是计算机科学中的基础理论之一。任何问题解决方案都不可能脱离数据结构而单独存在。本章作为全书的导引章节，将从数据与数据结构的概念谈起，帮助读者对将要学习的内容有一个初步的认识。

1.1 数据与数据结构

数据是一切有意义信息的基本存在形式。如何有效地组织和利用数据，长久以来一直是计算机科学家们思考的问题。无论是学术研究还是工程应用，数据和数据结构都扮演着极其重要的角色。本节主要向读者介绍有关数据和数据结构的一些概念。

1.1.1 数据及其类型

数据（Data）是信息的载体。有序的数据组织就形成了信息。信息是人类可以直接利用或感知的意识形式。而数据则是用来被计算机识别、存储和处理的，它是计算机利用或感知的基本单位。

在计算机科学中，更加具体的数据定义可以表述为计算机加工和处理的对象，这里数据可以是数值数据，也可以是非数值数据。数值数据包括一些整数、实数或复数等，它们主要用于工程和科学计算等；非数值数据包括文字、图像、声音等。在计算机中，这些数据都以二进制数的形式存放在物理介质上。每个二进制位称作一个位（bit），每 8 个二进制位组合起来称作一个字（byte）。

那么这些 0 和 1 是如何组成复杂而有意义的数据的呢？这就需要用到数据类型这个概念了。所谓数据类型就是一个值的集合和定义在这个值集上的一组操作的总称。一般而言，数据类型可分为原子型和结构型。在程序设计语言中，每一个数据都属于某种数据类型。类型明显或隐含地规定了数据的取值范围、存储方式以及允许进行的运算。这些已经事先定义好的数据类型就是所谓的原子型。经过数据类型规定后的若干有序 0 和 1 的组合就在计算机中呈现出一定的意义，最初的表现形式就是原子型数值数据。而在某些时候，一旦需要引入某种新的数据类型时，总是借助编程语言所提供的原子型数据类型来描述或定义新数据的存储结构，这也就是所谓的结构型。换句话说，原子型数据经过一定规则重组后即可形成结构型数据。

那些非数值型的数据也是用 0 和 1 来存储和表示的，只不过规则要复杂得多，其内容已经超出了本书的研究范围，这里不打算具体展开，有兴趣的读者可以参阅本章参考文献[1]以了解更多。下面首先来看看 C++语言中的几种常用原子数据类型，见表 1-1。这些数据在内存中存储时占据了多少内存空间呢？通过下面这段程序可以测得一些结果。

求职面经点拨

请试回答附录题库中的第 1 题。

```
#include <iostream>

using namespace std;

int main(int argc, char** argv) {

    cout << "bool: " << sizeof(bool) << endl;
    cout << "char: " << sizeof(char) << endl;
    cout << "short: " << sizeof(short) << endl;
    cout << "int: " << sizeof(int) << endl;
    cout << "long: " << sizeof(long) << endl;
    cout << "float: " << sizeof(float) << endl;
    cout << "double: " << sizeof(double) << endl;

     return 0;
}
```

求职面经点拨

请试回答附录题库中的第 2 题。

表 1-1 C++语言中的常用原子数据类型

类 型 名	描 述	取 值 范 围
bool	布尔型	true, false
char (signed char)	字符	–128 ~ 127
unsigned char	无符号字符	0 ~ 255
short (signed short)	短整型	–32768 ~ 32767
unsigned short	无符号短整型	0 ~ 65535
int (signed int)	整型	–2147483648 ~ 2147483647
unsigned int	无符号整型	0 ~ 4294967295
long (signed long)	长整型	–2147483648 ~ 2147483647
unsigned long	无符号长整型	0 ~ 4294967295
float	浮点型	3.4×10^{-38} ~ 3.4×10^{38} （绝对值精度）
double	双精度浮点型	1.7×10^{-308} ~ 1.7×10^{308} （绝对值精度）
long double	长双精度浮点型	1.7×10^{-308} ~ 1.7×10^{308} （绝对值精度）

编译并运行上述程序，结果如图 1-1 所示。可见布尔型数据在内存中仅占 1 字节，字符型数据也占 1 字节；短整型占 2 字节，整型、长整型和浮点型占 4 字节；双精度浮点型占 8 字节。同样一组 0 和 1 的序列如果数据类型不同，那么解释出来的结果将是不同的。

仅有原子型数据仍然是不够的。将原子型数据按照一定规则重组，就可以形成结构型数据。换句话说，结构化的数据可以由简单的数据组成。将某人的联系地址想象成一个结构化的数据，那么在这个结构中可能有若干个项目，它们都是一些简单的基本数据。例如，居住地的门牌号，然后是所在街道的名字、城市的名称、省份的名称和邮