

枕边算法书

| Algorithm stories which enhance programming imagination |

[韩] 林栢濬 著 崔盛一 译



TURING

图灵程序设计丛书

枕边算法书

| Algorithm stories which enhance programming imagination |

[韩] 林栢濬 著



人民邮电出版社

北京

图书在版编目(CIP)数据

枕边算法书/(韩)林栢濬著;崔盛一译.--北京:
人民邮电出版社,2018.2(2018.4重印)
(图灵程序设计丛书)
ISBN 978-7-115-47299-1

I. ①枕… II. ①林… ②崔… III. ①计算机算法
IV. ①TP301.6

中国版本图书馆CIP数据核字(2017)第284371号

内 容 提 要

本书第1章重点讲解各种常见算法,第2章主要介绍几种相对少见的算法,第3章和第4章探究其他程序员编写的代码,从中总结优秀算法应具备的特点,以及高级程序员应当持有的态度和必须培养的能力。书中以日常对话般浅显的叙述方式,帮助专业开发人员、刚刚踏入软件开发和编程门槛的初学者体会程序设计的创造性和成就感。

◆ 著 [韩]林栢濬

译 崔盛一

责任编辑 陈曦

责任印制 周昇亮

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号

邮编 100164 电子邮件 315@ptpress.com.cn

网址 <http://www.ptpress.com.cn>

北京隆昌伟业印刷有限公司印刷

◆ 开本:880×1230 1/32

印张:8

字数:169千字

2018年2月第1版

印数:3 001-5 000册

2018年4月北京第2次印刷

著作权合同登记号 图字:01-2016-2085号

定价:45.00元

读者服务热线:(010)51095186转600 印装质量热线:(010)81055316

反盗版热线:(010)81055315

广告经营许可证:京东工商广登字20170147号



算法轻松学

《枕边算法书》的重点在于“枕边”，而不在于“算法书”。我最幸福的时刻就是完成所有工作后，拿着一本通俗小说（或漫画）入睡。书名中的“枕边”旨在说明，这是一本能够营造最自在、最愉悦时刻的有趣读物。

本书的写作目的并不是想“介绍”算法理论或对算法进行“讲解”，而是想通过与专业编程人员进行“日常对话”，创造机会，使读者了解我们每天完成的“工作”多么有趣、多么美好、多么具有创造性。

书中涉及排序（sort）、搜索（search）、二叉树（binary tree）、列表（list）、回溯（backtracking）、散列（hash）、欧几里得算法、动态规划（dynamic programming）等比较常见的算法，以及回文（palindrome）、末日（doomsday）、Soundex、梅森素数（Mersenne prime）等比较少见的算法。第1章和第2章主要介绍这些算法，第3

章和第4章探究其他程序员编写的代码。

算法的重要性不言自明。前不久，我在 ZDNet Korea 上发表了题为“问题在于算法”的专栏文章。文章发表后，很多读者都产生了共鸣，并给予反馈。下面将部分内容分享给各位。

“高校不应该只注重培养学生的编码能力，而应当想办法培养能够利用算法解决问题的能力。像纹身一样，我们可以通过 MOOC、培训班、研究小组等方式学习编码技巧，必要时可以‘抹去’。但与自身融为一体的算法能力将会限制程序员的整体能力，因此，要在适当的时机掌握适当的知识，错过就很难再习得。”

当今时代，技术飞速变化，从属于特定技术、平台、语言、API 的编码技术的价值已大不如前，反倒是舍弃旧技术而掌握新技术的能力显得更加重要。战斗机的生命力在于其快速改变方向的机动力，程序员的生命周期也取决于是否具备能够迅速改变编码方式的能力，而算法正是培养这种能力的‘元能力’。

基于上述原因，美国的 IT 公司（除特殊情况外）在人才雇用方面并不会过多考虑精通特定技术或 API 的编程人员，而更倾向于在拥有基本能力（解决问题的能力，即算法）的基础上可以快速学习新技术并能应用于实际问题的人才。就像美剧《硅谷》中，主人公并不是非常高明的黑客或 Java 专家，而是一名开发过文件压缩‘算法’的普通程序员。

软件编程技术由名为‘算法’的‘细胞’组成，存在于‘算法细胞’内部的‘DNA’就是逻辑。对任何事都具备逻辑思维的人能够编

写非常优秀的代码，而缺乏这种逻辑能力的人无论多么努力学习‘编程’也无济于事。”

对于软件开发人员而言，学习算法的目的不应定位于就业或得到更高职位，而应该像玩拼图或猜谜语一样，使学习成为一种有趣的游戏。本书就是采用了这种趣味性的介绍方法，所以受到广大读者的欢迎。我希望不仅是专业的开发人员，刚刚踏入软件开发和编程门槛的初学者也能阅读本书。

编程的本质是算法，算法的本质是解决问题，解决问题的本质是令人兴奋的成就感。希望各位阅读本书时，能够真正感受到这种成就感。

虽然我撰写本书内容和代码时一直尽力消除各种错误，但相信还有不少“漏网之鱼”逃不开读者们敏锐的视线，请登录“图灵社区”本书首页（<http://www.ituring.com.cn/book/1785>）提送勘误。

特别感谢给予本书出版机会的 Hanbit Media 金泰轩社长和刘海龙总编，以及提供了很多重要想法及建议的任成春组长，还要感谢朴炫进代理和洪元奎组长。由衷感谢妻子金喜盛，她对下班回家后总是拿着笔记本电脑不知所踪的我始终给予鼓励和关爱。

林栢濬

2015 年 9 月写于美国新泽西州

第 1 章

爵士乐开启晨间香气

- 1.1 枕边的算法 2
- 1.2 用谜题解开算法世界 13
- 1.3 定义数据结构 19
- 1.4 输出 P 的程序 P 29
- 1.5 找出隐藏的 Bug 33
- 1.6 设计精妙算法 42
- 1.7 回文世界 49
- 1.8 康威的末日算法 56

第 2 章

摇滚乐伴随正午活力

- 2.1 排序算法 66
- 2.2 搜索算法与优化问题 73
- 2.3 动态规划法 84
- 2.4 散列算法 89
- 2.5 Soundex 算法 93
- 2.6 修道士梅森 100
- 2.7 程序员的“成就感”本质 104
- 2.8 文学编程 108

第 3 章

硬核朋克点燃午后激情

- 3.1 欧几里得算法 118
- 3.2 递归的魔术 124
- 3.3 李维斯特、夏米尔、阿德曼的数学游戏 131
- 3.4 RSA 算法 135
- 3.5 下午茶时间 137
- 3.6 RSA 算法 (续) 146
- 3.7 3 行 Perl 程序 152
- 3.8 赏析黑客们的诗 163
- 3.9 2 行 RSA 算法 177

第 4 章

古典音乐带来夜晚安逸

- 4.1 N 皇后问题 180
- 4.2 通过肉眼解答 N 皇后问题 184
- 4.3 藏在问题中的分立的算法 191
- 4.4 递归与栈 198
- 4.5 Jeff Somers 的算法 207
- 4.6 复习位运算符 225
- 4.7 2 的补码 229
- 4.8 分析 Jeff Somers 的算法 233
- 参考文献及网址 244
- 索引 245

第1章

爵士乐开启晨间香气



本章内容深度较浅、轻松易懂，包括几种拼图类问题、短小但需要认真思考的算法，以及回文算法和康威教授的末日算法等。这些算法都比较简单，不会对学习造成负担。



1.1

枕边的算法

我曾经为位于美国新泽西州的朗讯科技公司开发过一种基于 Web 的网络管理软件，使用该款管理软件的客户公司遍布欧洲、亚洲、南美洲及北美各地。其中，加拿大的一家公司提出要求，希望软件能够限制同一用户同时打开的网页数量。

为朗讯设计的网络管理软件中，用户通过网页浏览器每打开一个网页时，网站服务器就会生成一个与之对应的对象（此对象会应用于网络适配器、数据缓存、数据转换等场合）。这种模式下，若用户打开太多网页，则服务器端必将生成多个对象，从而导致服务器整体性能下降。因此，限制用户可同时打开网页数量的要求非常合理。

这个要求实现起来看似不难。收到一个打开网页的请求时，服务器端首先计算此用户当前已打开网页的数量。如果这个数量大于等于已设定的某个阈值，那么服务器端就会发送错误信息，而不是用户请求的网页。用户每打开一个网页就让计数器加 1，每关闭一个网页就

让计数器减 1，这样就能实现这种算法。其本身将会非常简单。

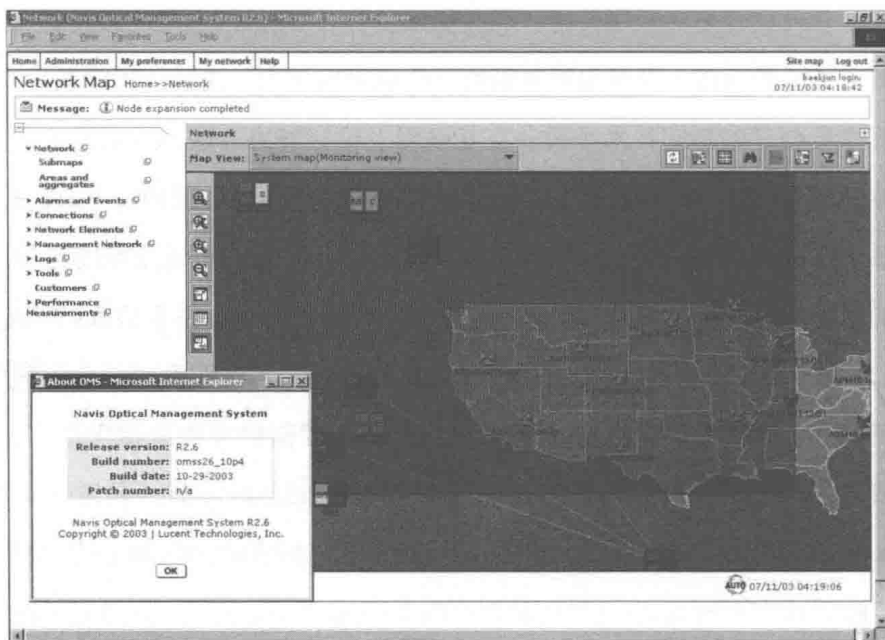
不过世事难料，即使是再简单的算法，实现过程也不会一帆风顺。浏览器为了打开一个网页，需要向服务器端发送 HTTP 请求命令，所以比较容易实时跟踪。但如何跟踪浏览器窗口的关闭或用户跳转到其他新网页（比如打开 www.google.com）的瞬间呢？如果服务器端不能跟踪网页的关闭或跳转瞬间，那么上述算法将无法实现。

从此刻开始，程序员将“痛并快乐着”。越难的问题越能激发程序员对成就的欲望，就像绷得越紧的弓弦威力也越大。遇到这种问题后，程序员会开始翻阅技术手册、在互联网上检索解决方法，或向身边的编程高手请教“如何能够实时跟踪网页关闭或网页跳转瞬间”。如果具备丰富的网络编程经验，那么上述问题就不值一提；而对于没有经验的读者而言，寻找解决方案的过程会成为寻找新知识的快乐旅程。

实际解决问题时，根据架构或编程语言的“语法”及“指令”的不同，实现方法可能略有不同。不过，无论用何种架构或编程语言，其基本概念都大同小异。接下来，我将按照自己的习惯，利用 Java Applet 解决此问题。关闭网页窗口或跳转到新网页时，与网页浏览器进程同时工作的 Java 虚拟机必将调用 `stop` 和 `destroy` 方法。因此，在方法内部向网络服务器发送信息，即可实时跟踪网页关闭或跳转瞬间。

既然能够跟踪网页打开和关闭瞬间，那么实现前面的算法就非常轻松。不过，具体实现过程并没有想象当中的那样顺利，有些问题仍然需要经过艰难而复杂的过程才能解决。

领会了算法的整体架构和实现方法的具体细节后，我向管理员承诺了一个完成工作的期限。既然许下承诺，就要遵守这个期限。因为对于程序员来讲，这就像生命一样重要。程序员大体上可以分为两种，一种人编程水平高，而且能按时完成任务；另一种编程水平不怎么高，但也能按时完成。至于不能在承诺时间内完成任务的人，则压根算不上是程序员。



▲ 朗讯基于 Web 的网络管理软件界面（www.lucent.com）

虽然客户要求的开发时间比较紧，但我简单梳理了算法原型（prototype）并记录如下。由此，我认为能够向客户保证开发进度。

- ❶ 接收从浏览器发出的页面访问请求。
- ❷ 比较表示已打开网页个数的 `current_count` 和预设值 `MAX_COUNT`。
- ❸ 若 `current_count` 小于 `MAX_COUNT`，则传送请求的网页后，`current_count` 加 1。
- ❹ 若 `current_count` 大于等于 `MAX_COUNT`，则传送错误信息。
- ❺ 从网页浏览器接收到网页关闭或跳转信息，则 `current_count` 减 1。

下面用代码实现此算法。

```
processPageOpenRequest ()
{
    if (current_count < MAX_COUNT)
    {
        sendResponse ();
        current_count++;
    }
    else
    {
        sendErrorMessage ();
    }
}
```

```
processPageCloseRequest ()  
{  
    current_count--;  
}
```

浏览器请求新网页时，会调用 `processPageOpenRequest` 方法；而接收到关闭网页或跳转到其他网页的信息时，则调用 `processPageCloseRequest` 方法。虽然包括多线程在内的很多部分都需要完善，但编写上述算法框架后，我还是一边享受着美味的咖啡，一边想着“在限期内完成开发任务并不困难”。但事后证明，我的这种想法是错觉。

要想发现此算法中隐藏的 Bug，需要了解网页浏览器的运行方式。如果是编程高手，那么即使不了解浏览器运行方式，也能指出此算法的内在问题。希望各位把自己想象是指挥整个项目的总设计师，然后找出其中问题。此处可以给出一个小小的提示：问题与两个方法调用的时刻，即 `current_count` 值发生变化的微小时刻有关。

大部分情况下，此算法看似运行正常。`MAX_COUNT` 的值设置为比较小的数（例如 10）之后，连续打开浏览器窗口到第 11 个时，出现错误信息。接着关闭一个已打开的窗口，使打开的网页数量变为 9，然后又能顺利打开第 10 个窗口。打开服务器端日志即可查看 `current_count` 的值已经增加到 10，这表明算法执行正确。

这种测试称为单元测试，是程序员为了确认软件是否满足要求而自行进行的测试过程。通常，为了查看新编程序代码或组件是否在整

个软件中运行正常，专门的测试部门会耗费大量的时间和精力进行测试。因此，单元测试是进行正式测试之前，为了检测基本功能而进行的“预测试”过程。

从某种角度讲，进行单元测试的认真程度是衡量程序员真正实力的标尺。往往有些实力不足（或工作态度不认真）的程序员，对自己编写的程序连最基本的测试都不做，而直接提交给测试部门（甚至是客户！）。程序员不容许有任何“失误”，所以细致而周密的测试工作是“必须”的，而不是“可选”环节。利用简单的测试就能提前发现 Bug，而若是让用户在使用过程中自己发现，那么不仅要损失开发成本，更是程序员的耻辱。

编写能够执行上述功能的类之后，我进行了编译，并安装到测试环境进行测试。测试过程比较顺利，没有发现什么问题，所以我打算将代码保存到源代码仓库。正要保存的瞬间，我无意中看到已经打开的第 10 个窗口。这时突然产生了要点击 Refresh 按钮的想法。目前为止的测试中，打开第 11 个窗口时，我只考虑了在新窗口中打开新网页的情况。

Refresh（刷新）的主要功能是，在同一网页窗口关闭第 10 个网页的同时，打开第 11 个网页（即相同网页）。按理说，此操作也应当能够正常完成。第 10 个网页关闭时，Java Applet 的 `stop` 方法会向服务器端发出 `current_count` 减 1 的请求，然后再发出对第 11 个网页的请求。按照这种请求方式，第 11 个网页应该顺利打开。网页被 Refresh 时，按照如下形式执行方法。