本科教学版

清华计算机图书·译丛

The Algorithm Design Manual

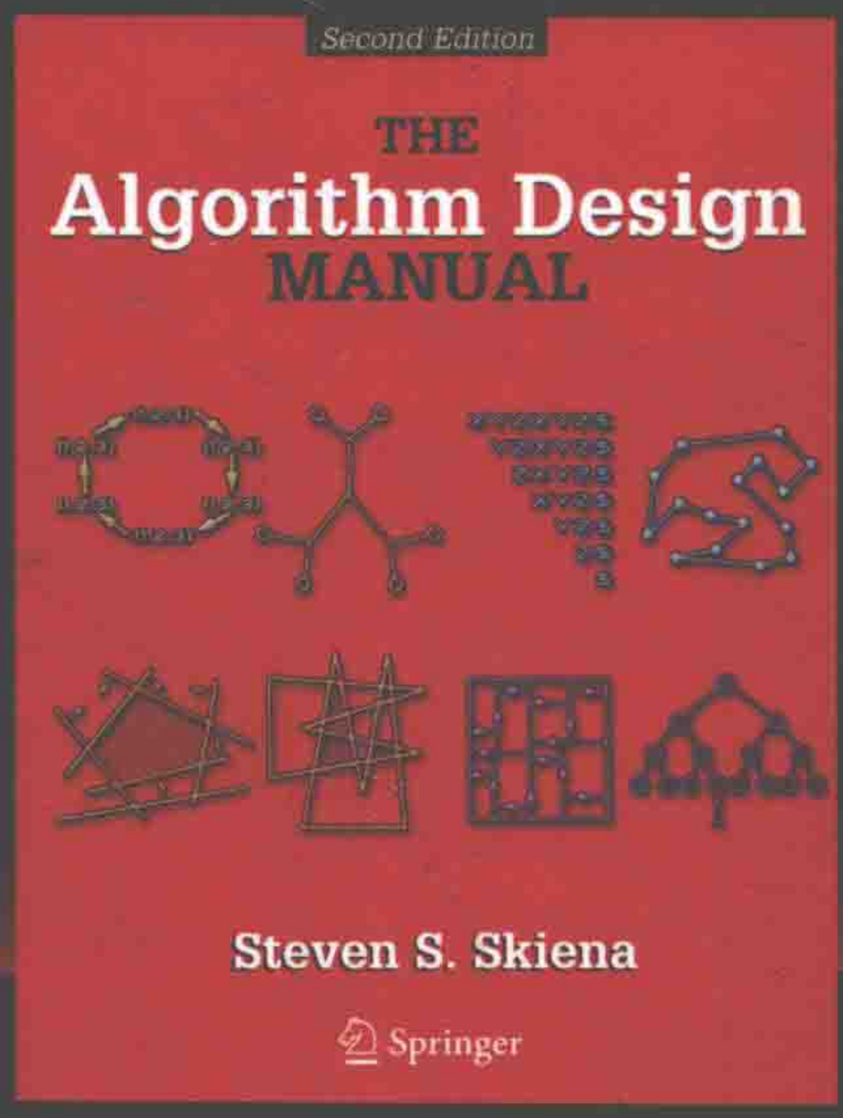
Second Edition

算法设计指南

（第2版）

Steven S. Skiena 著

谢腮 译



清华大学出版社

清华计算机图书译丛

The Algorithm Design Manual

Second Edition

算法设计指南

（第2版）

Steven S. Skiena 著

谢腮 译

清华大学出版社

北京

图书在版编目（CIP）数据

算法设计指南（第 2 版）/（德）斯蒂文·斯金纳（Steven S. Skiena）著；谢勰译. —北京：清华大学出版社, 2017

（清华计算机图书译丛）

ISBN 978-7-302-45734-3

Ⅰ.①算…　Ⅱ. ①斯…　②谢…　Ⅲ. ①算法设计—指南　Ⅳ. ①TP301.6-62

中国版本图书馆 CIP 数据核字(2016)第 286940 号

责任编辑： 龙启铭
封面设计： 傅瑞学
责任校对： 梁　毅
责任印制： 李红英

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： 清华大学印刷厂
经　　销： 全国新华书店
开　　本： 185mm×260mm　　**印　　张：** 23.75　　**字　　数：** 574 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： 69.00 元

产品编号：031141-01

中文版序

I have been gratified by the favorable reception which "The Algorithm Design Manual" has received to, and very excited that Chinese readers will get a new exposure to it through this translation.

Seeing my book in translation into a distant language is always a very humbling experience: I know that I wrote it, yet realize that I will never be able to read it. I thank Xie Xie, the translators, for doing a particularly diligent job with my book. I know he worked extra hard to translate the American-specific references I made throughout the book, which apparently made it more like translating a novel than a textbook.

I hope this book encourages greater study of algorithms in China. Indeed I hope that readers who work their way through it consider graduate study at Stony Brook University, where I teach. I hope you enjoy entering the world of algorithms, and find good luck with what you discover there.

Steven Skiena
Distinguished Teaching Professor
Stony Brook University

对于本书所获得的肯定我感到非常荣幸, 而中文读者现在可以通过这部译本来研读它, 也让我非常激动.

每当看到我的书被译成一种完全陌生的语言, 我总是感到十分汗颜: 我知道这是我写的书, 但更知道我永远没法读懂它. 感谢本书的译者谢勰非常尽心尽力地去翻译我的书. 这本书用了很多面向美国本土读者的典故, 我知道译者用了很多时间来诠释它们, 他为此所付出的辛勤工作更像是在翻译一本小说而不是单纯的教材.

我期望这本书能更深入地促进算法这门学科在中国的研究和发展. 另外我衷心希望那些学完本书的读者能够关注我所任教的石溪大学的研究生教育. 期盼本书读者能愉快地走进算法世界并有所收获, 祝你们好运!

Steven Skiena
杰出教学教授
石溪大学

译者的话

一名之立, 旬月踟蹰.
——严复

引介

算法世界中的故事由作者娓娓道来, 各种设计技术非常自然地穿插其中, 读者更像是在阅读一本小说, 你没看错, 这就是Steven Skiena教授的名作*The Algorithm Design Manual*. 要是在美国亚马逊网站上输入"Algorithms", 可以发现这本书长期排名居于前五, 而且还一度是计算机算法类的最畅销书籍(Best Seller). 你可随手翻开一页往下读去, 肯定觉得妙趣横生不忍释卷, 仿佛就像走进了Skiena教授的课堂一样, 这也许是它广受欢迎的主要原因吧. 此外, Skiena教授的主页上还提供了在线课程视频, 若是身临其境更能深入体会其妙.

考虑到国内算法课程教学情况, 我们认为本书的卷I(讨论"实用算法设计")单独成书比较适合讲授, 而且携带这本"算法小说"也更为方便. 为此我们征求了作者本人的同意, 特将这本书的精简版(仅包含卷I)译出以飨读者. 不过, 阅读是一个可能随时中断的过程, 而原书中许多段落只能在课堂讲授的语境中才能理解, 为此我们补充了一些起承转合的词句. 此外, 作者对很多掌故信手拈来, 而且经常还有弦外之音, 妙则妙哉, 可是对于不能一窥堂奥的读者实为遗憾, 中译本尽量对它们给出注释.

说明

- 每章中的*War Story*是本书的一大特色. *War Story*的原意是士兵脑海中难忘的战争故事, 而引申意则是每个人心中值得回忆的经历, 一般都带着些惊险或艰难. 尽管英勇故事(取自《英汉大词典》)、艰难历程、算海拾贝、算法轶事都能在一定程度上表达*War Story*这个词, 但它们都难以精确地等同于"作者Skiena个人在算法方面的*War Story*". 因此书中的*War Story*仍按原文列出, 读者可细细品味它的含义.
- 大部分人名和书名按原文形式给出, 少数历史人物和流传甚广的名著除外.
- 我们将"efficient algorithm"译为"高效算法", 但请注意这只是渐近意义上的"高效".
- 第3章中"抽象数据类型"和"数据结构"这两个概念略有混淆, 译文已作统一修改.
- 以log统一表示以2为底的对数. 原书中较为混乱, 有log, lg, $\log_2$等多种表示形式.
- 倍数问题. 例如"$g(n)$ is a million times larger than $f(n)$", 作者通常想表达的意思是$g(n)$是$f(n)$的一百万倍, 这是美国人的一种常用表述方式(尽管多有诟病).
- "和/或"(and/or)表述. 如果我们说A和/或B, 则意味着三种情况: 可能是A; 也可能是B; 还可能是A和B.

- 原书有许多网址已经失效, 我们直接替换为最新地址但不再一一说明.
- 读者可在http://www3.cs.stonybrook.edu/~skiena/algorist/book/errata查阅原书的最新勘误.
- 卷I中经常引用卷II的章节, 精简版的译文未作删减, 有兴趣的读者可查阅原书相关内容. 另外, http://www3.cs.stonybrook.edu/~skiena/algorist/book/中的"By Problem"分类亦收录了卷II中的所有问题(对应第12章到第18章).
- 本书由LaTeX排版, 书中的插图文件和原书基本一致(少量略作润色或重绘).

致谢

感谢原书作者Steven Skiena教授对中译本的大力支持, 特别是他无私提供了书中所有插图, 并在百忙之中为中译本作序. 此外, 与Skiena教授的多次讨论交流让我深深感受到他的谦和、容忍和智慧. 感谢Vaughan Pratt教授对于"Pattern"一词的详细解释, 他甚至查阅了OED并详尽地叙述了这个词的源流. 感谢清华大学出版社龙启铭编辑的不断鞭策和激励, 让我能够将这本书最终翻译完成. 感谢李树钧博士在LaTeX排版上的建议.

互动

由于译者学识所限, 书中必然有诸多疏漏之处, 还望读者朋友们不吝赐教, 您的宝贵意见和建议可反馈至DSAD2015@163.com, 勘误和资源将在https://github.com/xiexiexx/Translations上发布. 另外, 我们还会在新浪微博/微信公众号"算法时空"上展开对中译本的持续讨论与更新, 敬请关注.

谢 鄂

2017年1月

于古城西安

前言

我遇到的大多数专业程序员都不太愿意去解决算法设计问题. 这点很遗憾, 因为算法设计已成长为计算机科学的核心实用技术之一. 若想设计出正确、高效和易于实现的算法去求解真实世界的问题, 需要了解两种不同的知识体系:

- 技术——优秀的算法设计师懂好几种基本算法设计技术, 包括数据结构、动态规划、深度优先搜索、回溯以及启发式方法. 也许最最重要的设计技术应该就是建模了, 它能将杂乱现实世界中的应用问题提炼精化以便于用算法攻破, 这可称得上是门艺术.
- 资源——优秀的算法设计师都站在巨人的肩膀上. 他们不是每次都从一张白纸开始费尽心思最后创造出新算法来解决问题, 他们会先弄清楚这个问题目前的研究现状. 他们不是从零开始重新实现那些广为流传的算法, 他们会去寻找现有的程序实现并以此作为起点. 他们对许多经典算法问题都非常熟悉, 这些问题为大多数应用问题的建模提供了充足的素材.

本书意在作为算法设计的一本指南, 从而让学生和计算机从业人员能走进组合算法技术的殿堂. 全书分为两卷: 技术和资源. 前者是对计算机算法设计和分析技术的一般性指引. 后者则可让你进行查阅和参考, 它是由多条简介构成的算法问题便览,[1] 其中每一条都包含了算法资料、程序实现以及大量的参考书目.

致读者

自从1997年本书第1版初次面世以来, 它所受到的热烈欢迎让我一直倍感欣慰. 这本书被视为一本独一无二的指南, 它能教你用算法技术解决实际中的许多常见问题. 不过, 从本书十年前问世至今, 这个世界有了许多改变. 实际上, 如果我们将现代算法设计和分析的起源定在1970年左右, 那么从这本书诞生到现在的这个时间段在整个现代算法的发展历史中占了大约30%之多.

这本书有三个方面尤为受人钟爱: (1) 算法问题便览, (2) *War Story*, (3) 本书的电子资源部分. 这些特色在新版中得到了保留并有所加强:

- 算法问题便览——由于找出一个算法问题的现有进展可能会是一项艰巨的任务, 因此我提供了在实际中最重要的75个问题的简要介绍并汇集成“算法问题便览”. 通过查阅这份便览, 学生或从业人员可以很快地确定他们的问题名称和该问题的研究现状, 以及如何利用现有工作的基础去解决它. 为了帮你找到并确认问题, 每条问题简介都包含了“处理前”和“处理后”的对比示意图,[2] 它们描述了输入和输出应符合

---

[1] 译者注: 在本书中, 如无特殊说明, 便览通常指卷II中的算法问题便览, 每条简介都是一个关于算法问题的词条.

[2] 译者注: 这种示意图在算法问题便览中已经明确地标为“输入”和“输出”.

的规格. 有位颇具洞察力的评论家因为这份便览非常强大的缘故, 将我的这本书称作"算法世界搭车客手册"(*The Hitchhiker's Guide to Algorithms*).[1]
算法问题便览是本书最重要的部分. 为了在这一版中更好地改进它, 我对每个问题都向相关的世界级权威专家征求了反馈. 而且新版尤为关注对每个问题的现有软件实现讨论部分的更新.

- *War Story*——在实际中, 算法问题不会出现在大项目的初始阶段. 相反地, 它们通常会以子问题形式出现, 而此时你肯定能看到如下情景: 要么程序员不知道该如何继续下去, 要么目前的设计方案不足以完成该任务.
  为了让你更好地去观察了解算法问题在实际中究竟是如何出现的, 我们在书中纳入了一系列*War Story*, 或者可称作"我们与真实问题交战所经历的故事". 这些故事的寓意是, 算法设计和分析不仅仅是理论, 它更是在你真有需要的时候可以派上用场的一个重要工具.
  这一版保留了上一版中的所有*War Story*(适当进行了更新), 新增的*War Story*覆盖了外部排序、图算法和模拟退火以及一些其他主题.
- 电子资源部分——由于从事实际工作的人通常想找到一个程序而不光是算法, 我们提供了链接让你可以获取那些可靠的算法实现(只要原站点仍有效). 同时我们也已将这些实现汇集在网站(`http://www.cs.sunysb.edu/~algorith`)上, 读者可以很方便地检索. 本书初版刚问世之后没多久, 我们这个站点就一直稳居Google上的"算法"网站排名首位.
  此外, 我们所提供的推荐建议能让你更容易地找出最适合于某项任务的代码. 有了这些程序实现之后, 算法设计的关键问题就变成了对你的应用问题正确地建模, 而不是花时间去熟悉某个实际算法的细节. 这个重要观点渗透于整本书之中.

我们在本书中没有讲到的内容也同样重要. 我们不强调数学形式的算法分析, 所以大多数的分析论证以较为直观的方式给出. 你在这本书中找不到一个定理. 如果需要了解更多的细节, 读者应该去研究本书中所提到的程序和参考文献. 一言以蔽之, 这本指南的目标是让你尽可能快地走上正确的方向.

致教师

本书所涵盖的素材足以开设一门标准的"算法导论"课程. 我们假设读者已学完相当于中级编程的课程(通常名叫"数据结构"或"计算机科学(II)").

你可在`http://www.algorist.com`下载用于讲授这门课程的一整套课程幻灯片. 此外, 我还基于上述幻灯片录制了在线音频和视频讲座, 可供一个完整学期的算法课程使用. 通过神奇的互联网, 让我来帮你教这门课!

本书强调的是设计而非分析. 它既适用于传统的课堂授课, 也适应于新兴的"主动式学习"方法, 也就是教授不再讲课而是去引导学生分组来解决实际问题. 实际上, 本书的*War Story*就是主动式学习方法的鲜活实例.

我在全书中做了很多教学法方面的改进. 体现"教材导向"教学法的特征包括:

---

[1] 译者注: 脱胎于科幻名著《银河系搭车客指南》(*The Hitchhiker's Guide to the Galaxy*).

- 更轻松的讨论——本书卷I的讲解材料是上一版的两倍. 新增部分注重对基础材料给出更细致全面的阐述, 而并不是去增加更多的专题.
- 以错误开始——在算法教材中重要的算法通常会以既定事实的面貌出现, 而这样会掩盖其中的设计理念以及为何其他方案会失败的微妙原因. *War Story*通过一些应用问题来讲解藏于解决方案背后的风起云涌, 不过我还把这种讲授模式扩展到了经典算法设计的内容中.
- 停下来想想——这里会举例讲解我自己求解某个具体作业题时的思路演进——从错误开始一直到最后解决的全过程. 我把这类问题模块散布于整本教材中以提升读者对解题活动的积极性. 每个问题之后马上给出了解答.
- 更多且更精良的课后习题——这一版的习题是前一版的两倍. 那些在上一版中已被发现不易理解或模棱两可的习题已作改进或替换. 书中所有问题都标有难度等级(从1到10).
- 自我激励式考试设计——在我的算法课中, 我向学生保证所有的期中和期末考试题都直接取自本书的习题. 这样规定了一种"促学型考试"(student-motivated exam), 因此学生完全明白应该如何学习才能在考试中有好的表现. 习题由我精心挑选而得, 可保证总数量、广度及难度都能满足这种考试的需求, 此外备选问题不会过少也不会过多.
- 所得教益——我们对这部分内容给出显著的阴影标记, 以强调这是你需要从对应章节中所汲取的大局观.
- 编程挑战赛的相关链接——每章的习题都包含三到五个"编程挑战赛"问题的链接, 它们来自`http://www.programming-challenges.com`. 这些链接可用于补充纸笔算法课程所缺乏的实际编程教学环节.
- 更多真代码, 更少伪代码——本书中的算法更多是以代码(用C语言写成)而不是伪代码形式出现. 我认为对于较简单的算法而言, 经过测试的真实程序实现不但正确而且可靠, 这可比那些形式化稍弱的表述[1]强多了. 完整实现可于`http://www.algorist.com`下载, 你可以深入学习研究这些程序.
- 章节注释——卷I的每章都包含一个简短的章节注释, 它们为读者指明了基本书目和补充参考文献.

致谢

十年后新书的题献会主要关注时间流逝中的世事变幻. 自第1版问世后, Renee已经成了我的妻子, 然后又变成两个孩子(Bonnie和Abby)的母亲. 我的父亲离开了人世, 但我的母亲和兄弟(Len与Rob)仍然与我同在, 并且在我的生命中占据了非常重要的地位. 我将这本书献给我的家人, 不管是新成员还是老成员, 无论尚在或是离去.

[1] 译者注: 指的是以伪代码或普通语言形式描述的算法.

我要感谢以下人士, 他们为新版做出了卓越的贡献: Andrew Gaun和Betson Thomas提供了很多方面的帮助, 尤其是在构建新网站`http://www.cs.sunysb.edu/~algorith`的基础架构和有关手稿准备的各类事项上. David Gries热心无私地为本书提供了很有价值的反馈. Himanshu Gupta和Bin Tang非常大胆地用这版的手稿在课堂讲授. 此外, 我还要感谢我在Springer出版社的编辑Wayne Wheeler和Allan Wylde.

我邀请了一批相关领域的算法大师审查了"算法世界搭车客手册"部分(即卷II)的那些章节, 他们与我分享了聪明才智, 并督促我不断进步. 感谢他们:

Ami Amir, Herve Bronnimann, Bernard Chazelle, Chris Chu, Scott Cotton, Yefim Dinitz, Komei Fukuda, Michael Goodrich, Lenny Heath, Cihat Imamoglu, Tao Jiang, David Karger, Giuseppe Liotta, Albert Mao, Silvano Martello, Catherine McGeoch, Kurt Mehlhorn, Scott A. Mitchell, Naceur Meskini, Gene Myers, Gonzalo Navarro, Stephen North, Joe O'Rourke, Mike Paterson, Theo Pavlidis, Seth Pettie, Michel Pocchiola, Bart Preneel, Tomasz Radzik, Edward Reingold, Frank Ruskey, Peter Sanders, Joao Setubal, Jonathan Shewchuk, Robert Skeel, Jens Stoye, Torsten Suel, Bruce Watson, and Uri Zwick.

很多习题都来源于我的同事或是受到其他教材的启发. 多年后要想再弄清它们来自哪里则是一项挑战, 不过每个问题的来源(我已尽全力重新搜集)都已发布在网站上.

若是不感谢为本书初版做出重要贡献的人也是很无礼的. Ricky Bradley和Dario Vlah建立了网站所需的坚实基础架构, 不但逻辑清晰而且易于扩展. Zhong Li用xfig绘制了卷II中的大部分算法示意图. Richard Crandall, Ron Danielson, Takis Metaxas, Dave Miller, Giri Narasimhan和Joe Zachary都审查过第1版的初稿——他们的反馈都是经过深思熟虑的, 也正是这些反馈使得本书不断完善, 最终呈现在读者面前.

我对算法所了解的很多东西都是和我的研究生在一起学到的. 他们中的很多人(Yaw-Ling Lin, Sundaram Gopalakrishnan, Ting Chen, Francine Evans, Harald Rau, Ricky Bradley和Dimitris Margaritis)都是相关*War Story*中的真实英雄. Estie Arkin, Michael Bender, Jie Gao和Joe Mitchell是我在石溪大学的朋友和算法领域的同事, 与他们共事与相处都很开心. 最后, 感谢我的朋友们, Michael Brochstein以及在纽约的其他友人, 在我离开石溪造访他们时, 共渡了许多美好时光.

说明

作者大度地接受对任何不足之处的批评乃是一个传统. 可我有所不同. 本书中所出现的任何错误、缺陷或问题都是他人的过失, 不过你要指出来我会非常感激, 这样我就可以去追究相关人士的责任了.

Steven S. Skiena
石溪大学(Stony Brook University)[1] 计算机科学系
纽约 石溪(Stony Brook, NY) 11794-4400
`http://www.cs.sunysb.edu/~skiena`
2008年4月

[1] 译者注: 以前称为"纽约州立大学石溪分校"(State University of New York at Stony Brook).

目录

卷 I

实用算法设计

第1章

算法设计导引

何谓算法? 算法是完成某项特定任务的具体步骤. 算法更是藏于程序背后的思想, 当然这个程序本身至少要过得去才能谈及思想.

一个算法想要让人关注, 它一定要能解决一个有着清楚而且准确叙述的一般性问题(*problem*). 要想精确地表述清楚一个算法问题, 我们得描述该算法问题所要处理算例(*instance*)[1]的完备集合, 以及处理完任意一个算例之后所想要得到的输出. 分清一个问题和问题的一个算例, 这点极其重要. 例如排序这个算法问题[2]可按如下方式来定义:

问题: 排序

输入: 一个由n个键(key)组成的序列$a_1, \ldots, a_n$.

输出: 输入序列的置换(即重新安排)$a'_1, \ldots, a'_n$, 它满足$a'_1 \leqslant a'_2 \leqslant \cdots \leqslant a'_n$.

排序的算例可以是存储姓名的一个数组, 例如{*Mike*, *Bob*, *Sally*, *Jill*, *Jan*}, 也可以是存储数字的一个列表, 例如{154, 245, 568, 324, 654, 324}. 先搞清楚你是不是在处理一个一般性的问题, 这才是你通向最终解决它的第一步.

接收满足问题要求的输入算例, 再将其转换成我们想要的输出, 算法(*algorithm*)就是完成上述任务的具体步骤. 有许多不同的算法能求解排序问题. 例如插入排序(*insertion sorting*)就是一种排序的方法, 它一开始会从序列中取出一个元素(从而形成一个平凡的有序列表), 然后再将序列中余下元素按输入次序逐个插入到列表中, 并保证插入后列表仍然有序. 以C实现的插入排序算法描述如下:

```
void insertion_sort(item s[], int n)
{
    int i, j;                       /* 计数器 */
    for (i = 1; i < n; i++){
        j = i;
        while ((j > 0) && (s[j] < s[j - 1])){
            swap(&s[j], &s[j - 1]);
            j = j - 1;
        }
    }
}
```

---

[1] 译者注: 为区别书中所举实例, 此处译为"算例", 对算例的描述即规定了对输入数据的要求.

[2] 译者注: 需要特别注意, 问题和算例在本书算法体系中是两个特定的概念, 与我们通常对它们的认识略有不同. 因此作者又进行了强调, 下同.

图1.1给出了插入排序算法处理某个具体算例(单词“INSERTIONSORT”中的字母)时数据逻辑关系变化的情况演示.

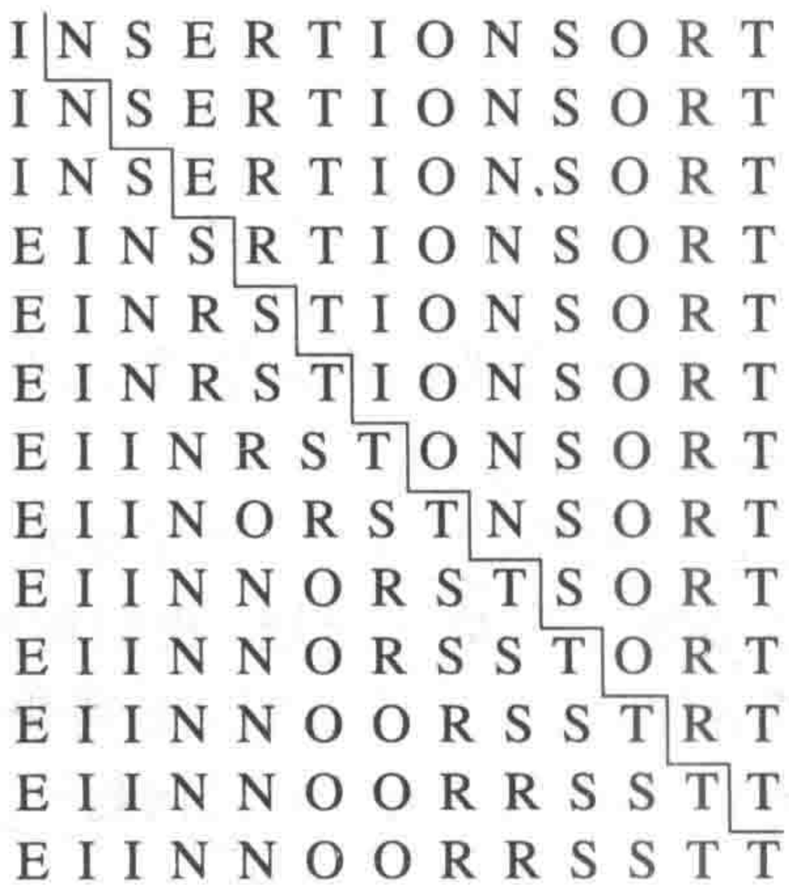


图 1.1 插入排序运行情况演示(时间顺序自上而下)

注意此算法具有通用性. 只要能给出合适的比较操作(<)来测出两个键谁应排在前面, 这个算法就能像处理数字那样将姓名排好次序. 根据我们对排序问题的定义, 可以毫无困难地证明该算法能将所有可能出现的输入算例正确地排序.

一个优秀的算法得具有三个理想的特性. 我们所追求的算法应该——正确(*correct*), 并且高效(*efficient*), 同时易于实现(*easy to implement*). 这些目标也许无法全部实现. 不过对于工业界所用的程序, 只要能对问题给出还算不错的答案并且不会让应用程序速度减慢, 通常都是可以接受的, 我们也不会再去管是否存在更好的算法. 在工业界, 通常只有出现性能上或法律上的严重问题后, 才会引发“寻找最佳答案”或“达到最高性能”这样的议题.

我们在本章中将重点关注算法正确性的问题, 而将效率的相关讨论推迟到第2章. 我们很少能立即判定某个算法能正确无误地解决所给的问题. 大多数情况下, 有了算法的正确性证明才能知晓算法是正确的, 这种证明能解释为什么我们能确信该算法在输入问题的任意算例情况下都肯定能得到所要的结果. 不过, 我们在进行深入讨论之前将先举例说明, 为什么“这是显然的”绝不足以证明正确性, 反而这句话通常完全就是错的.

1.1 机器人巡游优化

让我们考虑一类常在制造业、运输业和测试应用中常会出现的问题. 假定给我们一个机器臂, 它配有一件工具(比如烙铁). 在制造电路板时, 所有的芯片和其他电路元件必须焊接固定到基片上. 更确切地说, 每个芯片都有一组触点(或焊线)要焊到电路板上. 要为机器臂编制程序以完成此作业, 我们首先必须编制出一套触点的访问次序, 这样机器臂就可以先访问(并焊接)第1个触点, 然后是第2个, 第3个……这样继续下去, 直到完成作业为止. 接下来机器臂移回到第1个触点, 为下一块印刷板作准备, 这样使加工轨迹(tool-path)变成一个封闭的巡游(或称为环).