

## 人物专访

# 问高德纳的二十个问题\*

作者: 高德纳(Donald Ervin Knuth)

译者: 周 明

关键词: 《计算机程序设计艺术》 ACM 图灵奖获得者

编者按: 高德纳 (Donald Ervin Knuth)<sup>1</sup>, 美国计算机科学家、数学家, 1974 年图灵奖获得者。他是计算机算法和程序设计技术的先驱, 计算机排版系统 TeX 和 METAFONT 的发明者。他的经典巨著《计算机程序设计艺术》(*The Art of Computer Programming, TAOCP*), 洋洋数百万言, 3000 页的多卷本, 堪称计算机科学理论与技术的经典巨著。他誉满全球, 被称为算法分析之父。

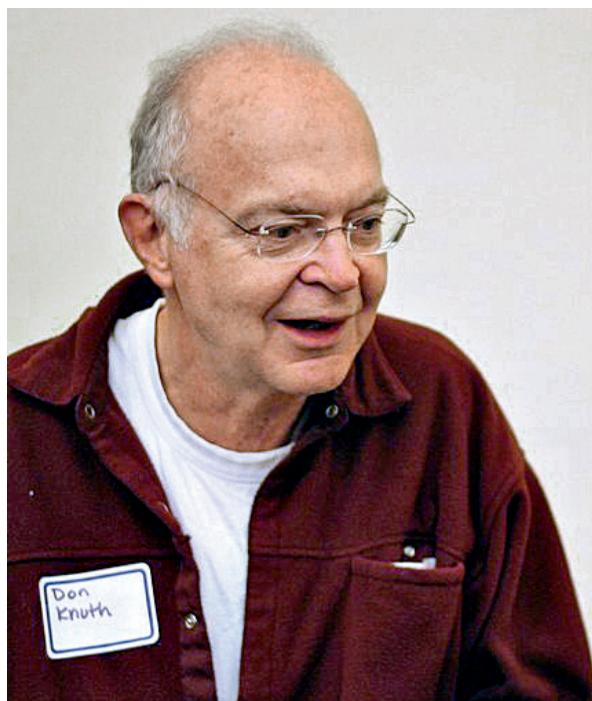
高德纳在收到《计算机程序设计艺术》第二卷校样时, 发现出版社把书中公式排得难看了, 他随即发明了排版系统 TeX 和字形设计系统 METAFONT。他还发明了 Web 和 CWEB 计算机编程系统, 设计了 MIX/MMIX 指令系统。作为学者, 高德纳提倡软件共享, 强烈反对软件专利的授权, 他曾对美国专利和商标局以及欧洲专利局表达过他的反对意见。

《计算机程序设计艺术》的撰写, 始于他攻读博士学位期间。原计划出版七卷, 第一卷《基本算法》于 1968 年出版, 第二卷《半数值算法》于 1969 年出版, 第三卷《排序与查找》于 1973 年出版, 第四卷《组合算法》尚未全部完成。《计算机程序设计艺术》一书, 以其内容的丰富和深刻被人们喻为经典, 称之为“计算机圣经”。该书现已被翻译成多种文字在世界各国广泛流传, 发行量创造了计算机类图书的最高记录, 成为 Addison-Wesley 出版社成立以来销路最好的图书。我国也由苏运霖翻译并由多家出版社出版。最近《计算机程序设计艺术》一书发行了电子书, 可以在平板电脑上直接阅读。高德纳已于 1992 年在斯坦福大学荣誉退休, 以便集中精力完成这部巨著。

为庆祝《计算机程序设计艺术》电子书的出版, 主办方特意邀请计算机科学家、高德纳的同事、粉丝等, 他们每个人可以向高德纳先生提出一个问题, 一共 20 个问题。每一个提问者都感到非常荣幸, 他们认真、精心地准备了要问的问题。在回答这些问题时, 高德纳回顾了他过去 50 年精彩的研究和著述生涯, 并对未来计算机发展提出了精辟见解。他的访谈体现了他对计算机科学的执着精神, 以及对青年计算机工作者的殷切期望。本刊经过高德纳先生同意, 翻译了这篇访谈录, 同时根据中国读者的兴趣特点做了适当的删节和补充。兹刊登如下, 以飨读者。

\* 本文译自 <http://www.informit.com/articles/article.aspx?p=2213858>。

<sup>1</sup> Donald Ervin Knuth 翻译为唐纳德·克努特, 也译为克努思, 1977 年 Knuth 访问中国前夕, 姚期智教授的夫人储枫教授为他取的中文名字为高德纳。——编者注



1. 乔恩·本特利 (Jon Bentley), 研究者<sup>2</sup>: 可真难得呀! 上次我有这样的机会还是在 1974 年 6 月, 您在斯坦福大学上“数据结构”课的时候。在最后一天的课堂上, 您向我们敞开心扉, 允许我们向您提任何问题 (除了政治和宗教以外)。我还清楚地记得您那天被问及的一个问题: “在您编写的所有程序中, 哪一个最让您引以为豪?”

您的回答 (40 年后我只记得大概了) 是为一个小型计算机编写的编译器。该小型计算机只有 1024 字节的可用内存。您第一个版本的程序有 1029 字节长, 超了 5 个字节, 但您最终还是设法把它缩小到 1023 个字节。您说您特别骄傲的是可以把这么多的功能塞进这么小的内存里。

我今天的问题与上面这个问题类似, 就是在您所写的程序中, 哪些程序您觉得最引以为豪? 为什么?

高德纳: 我也想问你同样的问题呢! 这个问题就像是问有好几个孩子的父母, 最喜欢的孩子是哪一个小, 很不容易回答。

当然, 我对 TeX 和 METAFONT 都感到十分骄

傲, 因为它们似乎帮助我们改变了世界, 也因为它们给我带来了许多友谊。由于这两项技术, 也使得我今天的电子书成为可能。我感到非常高兴的是, 我 30 年前做的工作, 经历了这么多年的技术变迁仍然还有用。像这本 3000 页的《计算机程序设计艺术》, 在小小的平板电脑上, 看起来如此美妙, 即使经过缩放, 也仍然很棒。

当我在 20 世纪 90 年代为《计算机程序设计艺术》第四卷做准备的时候, 我用你和我都知道的自然语言编程 (literate programming) 写了几十个简短的程序例程。那些工作已被收录到 *Stanford Graph Base* (1994 版) 一书中。我仍然喜欢使用和修改它们。我最喜欢的是塔扬 (Tarjan) 求解强连通分量 (strong component) 的漂亮的算法实现。如有兴趣, 大家可参看 *Stanford Graph Base* (1994 版) 的第 512~519 页。

不得不承认, 我也为我实现的 IEEE 浮点算术的程序感到特别骄傲。请参见我的 *MMIXware* 论著 (1999 版), 以及那本书里对 MMIX 的元模拟器 (meta-simulator) 的描写。在此书中, 我首次全面解释了先进的流水线计算机的很多原理。

自然语言编程一直是我人生最大的乐趣之一。事实上, 我发现我每周大约会写两个程序, 大小都有, 也是在为下一卷书做准备。

2. 戴夫·瓦尔登 (Dave Walden), TeX 用户组: 您愿意公开您 3000 页巨著《计算机程序设计艺术》的手稿吗? 作为您在 1965 年前后对计算机算法和分析看法的一个有历史价值的文物, 我想很多人都想看看。

高德纳: 学者们可以到斯坦福大学博物馆, 在那里可以看到我的第一卷到第三卷的手稿。今后其他的手稿也会存放在那儿。我看没有什么必要非要保留它们让更多人看到, 尽管我认为那里面关于棒球的、后来我又决定不用的材料都挺不错的。我觉得应该先公布计算机科学先驱著于 20 世纪 40~50 年代的手稿资料。

我在今天写的书稿里也试图保留我早期稿件年

<sup>2</sup> 世界著名计算机科学家, 被誉为实践探索先锋, 影响算法发展的十位大师之一。——编者注

轻的风格,当然不是指那些天真或者过时的东西。我的英语也得到了提高,比如我也学会了何时说“that”而不是“which”,这要部分归功于盖·斯蒂尔(Guy Steele)的指点。

**3. 查尔斯·雷泽森(Charles Leiserson),麻省理工学院:**《计算机程序设计艺术》一书体现了您对计算机科学的大爱,特别是对算法和离散数学的爱。但爱并不容易。在写这本书时,您感觉哪些时候必须尽最大努力去克服阻碍您前行的困难?

**高德纳:**查尔斯,我也想问你同样的问题呢!

对我来说,最难的事始终是如何削减篇幅。现在看来不是很成功,尽管有很多地方重新写了。

最艰难的技术挑战是为MMIX编写模拟器。我需要在幕后制作,以便形成书上的样子。我觉得这是我经历的最困难的编程。要是没有自然语言编程方法,我认为我可能无法成功地完成这项任务。

许多“担任主角”的数学章节也很费劲。不过,总体而言,在撰写TAOCP的50多年以后,我认为通过做这件事学到的东西是对我艰苦努力的最大补偿。

**4. 丹尼斯·沙沙(Dennis Shasha),纽约大学:**一个漂亮的算法和一个漂亮的理论应该怎样比较才算好?换句话说,对算法和理论,您的审美标准是什么?

**高德纳:**漂亮有很多方面,当然,都是因人而异的。对我来说,漂亮的定理和算法出自不同的原因。有些是因为它们有许多不同的应用;有些是因为在有限的资源下,它们表现仍然优秀;有些是因为它们有赏心悦目的模式;有些则是因为它们有着充满诗意的概念。

例如,我提到了强连通分量的塔扬算法。塔扬设计的这个问题有几种数据结构,这些结构以一种令人惊异的美丽方式组合在一起。因此,在探索一个有向图需要搜索的量时,会发现总是在你可控的范围之内。另外,他的算法同时也可做拓扑排序。

有时甚至可以通过展示一个漂亮的算法来证明

一个漂亮的定理。比如在TAOCP中,定理5.1.4D和推论7H就是如此。

**5. 马克·陶伯(Mark Taub),培生集团:**现在的App(特点是小、功能单一、网络程序)已经成为今天主流的编程模式,它的出现是否对您的TAOCP计划产生影响呢?

**高德纳:**人们在写App的时候已经使用了本书第一卷的思想和模式。而且App使用了不断增长的程序库,这些方式与TAOCP密切相关。这些库的用户应该知道库的具体实现方法。

未来的几卷将可能是让人更喜欢的App类型,因为我一直在收集很多有趣的游戏和难题,试图引人入胜地展示这些程序的设计技术。

**6. 拉迪亚·帕尔曼(Radia Perlman),英特尔:**(1)什么是您曾经希望但是最终却没有收录到TAOCP里的内容?(2)如果您出生在200年以前,您会从事哪一个职业呢?

**高德纳:**对于问题(1),我想包括的内容差不多都已收入,要么出现在现有的几卷,要么会出现在将来发表的几卷中。卷四B一开始会用几十页介绍某些新颖的数学方法,这些都是我在写卷一时并不知道要写的内容(现在这部分通过我的网站可以看到,名为“数学预备”)。我计划未来出版新卷需要引用最新发明的材料时,发行一些补充的“分册”,这些材料将归入第三卷。

问题(2)是一个多么令人着迷的问题。以前还从没有人问过我呢。

如果我出生在1814年,我受到的教育肯定非常有限,而且几乎没有办法接触到知识。我的祖辈都是劳动者,在不属于自己的农场里干活,居住于现在称之为德国北部的地方。

但我想,你心里其实想问:如果我是当时少数可以受到很好教育的人之一,而且可以自由选择职业的话,我会怎么办。

我一直想成为一名教师。事实上,我上一年级

<sup>3</sup> 极客是美国俚语“geek”的音译。随着互联网文化的兴起,这个词含有智力超群和努力的语意,又被用于形容对计算机和网络技术有狂热兴趣并投入大量时间钻研的人。——编者注

的时候，就想教一年级；上二年级时，我想教二年级；等等。我最终成为了一名大学教师。因此我想，如果可能的话，我会是一名教师。

继续这种推测，我要解释的是极客<sup>3</sup>。弗雷德·格林贝格尔很久以前告诉我，在他的经验里，大约2%的大学生对计算机着迷。这就像他和我曾经经历的一样。这个数字一直在我的脑海里，多年来我多次确认了他的这个观察经验。例如我了解到，1977年伊利诺伊大学有1.1万名学生，其中220人学计算机科学！

因此我相信，世界人口的一小部分已经获取了一种特殊的思维方式，我碰巧也在其中。这些人在计算机科学领域有了自己的学科名字后，碰巧发现了彼此的存在。

为简单起见，让我说像我这样的人是“极客”，他们约占世界人口的2%。我不知道如何解释计算机科学系为何得以迅速崛起——从1965年开始到1975年，在10年时间里，几乎所有的学院和大学从无到有，都设立了计算机系。但我知道，这些计算机系为计算机极客们提供了一个可以一起工作的“家”。同样，我也不知道有什么可以更好地解释这么多年来我所见证的很多软件项目最后失败的原因，除了一个假设，那就是这些项目并没有交给真正的计算机极客。

那么，谁是19世纪早期的极客？在1814年之前的一段时间，我也许会说，亚伯（1802年），他一般被认为是数学家。还有雅各比（1804年）、汉密尔顿（1805年）、柯克曼（1806年）、摩根（1806年）、刘维（1809年）、库默尔（1810年）和中国的李善兰（1811年）。我列出的这些“数学家”的著作令我着迷。之后到了1814年，加泰罗（1814年）、西尔维斯特（1814年）、布尔（1815年）、维伊尔斯特拉（1815年）和博查特（1817年）。我会喜欢与这些人为伍，如果幸运的话，我可能也会做着与他们类似的事情。

顺便说一句，我认为历史上第一个归类为“100%极客”的人是阿兰·图灵。他的许多前辈有很强的极客症状，但图灵确实是完全地“极客”化了。

**7. 托尼·加迪斯 (Tony Gardis), 作家:** 您还记得您是何时发现了编程的乐趣，因此决定一生为之奋斗？

**高德纳:** 1957年夏，在克利夫兰 Case Tech 的大一和大二暑假期间，我被允许与 IBM 650<sup>4</sup> 待了一夜，那时，我完全被它迷住了。

虽然我确信可以把计算机看作是一生的工作，尽管那时我还没有听说过有这样的职业。确实，如上面所说，我一直想做一名教师。我曾在1958年写过一个编译器的手册，后来由于机缘巧合，我把它用在了我的教科书里面，并在我的课堂上使用（1959年）。不过，编程对我而言起初只是一个爱好。在此之后，我在读研究生时，编程成为我依赖的一种手段。

我没有看到计算机编程和我想做的数学教师的职业有什么联系，直到我遇见了鲍勃·弗洛伊德。那是在1962年底。我也没有预见到计算机科学成为一门学科，直到1964年我遇到了乔治·弗思。

**8. 罗伯特·赛哲维克 (Robert Sedgewick), 普林斯顿大学:** 我记得几年前，您的立场是，您不想试图让所有的人都去阅读您的书——因为您知道，只有对那些有兴趣和天赋的，并且喜欢编程，愿意探索编程和数学关系的人，您的书才有特别的价值。我想知道您现在的想法有没有变化？我们的社会花了很长时间才认识到普及教育的好处。现在的问题是，是不是所有的人都应该学习编程，您是怎么看待这个问题的？

**高德纳:** 编程，当然不仅仅是一门普通的课程，本质上它可以强化并适用于许多不同类型的学科。我知道你在普林斯顿大学和利用网上教学来教授学生先进的编程知识，取得了很大的成功。

但你的问题涉及的是每一个人。我还是觉得要

<sup>4</sup> 20世纪50年代初期，IBM 开始迅速进入电子数据处理的新领域。IBM于1953年宣布，IBM 650 磁鼓数据处理机的电子计算可靠性达到一个新水平。每当发生机器处理错误，IBM 650 可以自动修复，这在当时是很大的改善。——编者注

过很多年，我才会建议自己非常聪明的妻子、儿子和女儿学习编程，更不用说其他人了。

几年前，尼克·特雷费森告诉我，他刚刚参观了他儿子所在牛津就读的高中（是世界上最好的中学之一），了解到居然没有一个学生知道如何编程！英国现在开始改变，而且比美国的变化速度更快。这种革命几乎可以肯定的是，需要一代人甚至更长的时间来完成。可是教师将来自哪里？

我的经验是启用大学生，他们对编程感兴趣，希望成为他们生活的一部分。TAOCP本质上是供专业人士使用的，主要是为极客编写的，不是为普通读者。

另一方面，在我的书里，我尝试为非专业读者来解释一切，我尽可能不用术语，也尽可能避免高等数学术语。因为这些会把普通程序员吓跑。只要有可能，我尽量把理论文献的艰涩内容转换成一个高中生可以理解的语言。

我知道，即使这样，我的书还是很不容易理解，即使极客也有难度。但我要是不这么做，我的书就可能就更难理解了。

**9. 芭芭拉·斯蒂尔 (Barbara Steele) :** 您使用什么工具，把您的书转换为电子书？这个过程是怎样的？

**高德纳 :** 我知道这些书不太容易做成电子书，必须要请专家转换来完成。幸运的是，我在2011年获得了一笔奖金，我把它用来请专业人士帮助我进行了书的转换，达到了我所期待的质量，而且也没有耽误我的写作计划，这要感谢伯克利数学科学出版社 (MSP) 的帮助。

我的主要目标是让书籍容易被检索。这是一件相当具有挑战性的事情。其次我想让读者点击任何练习题、方程式、插图、表、算法的标号，就可以轻松跳转到练习题，也可以在练习题和答案之间轻松跳转。

伯克利数学科学出版社的工作人员写了一个特殊的软件，把我的 TeX 源文本转换成合适的输入以创建 pdf 文件。我不知道细节，我只知道他们使用了类似于用于 WEB 和 CWEB 的“更改文件”。我

已经仔细地检查了转换结果。效果不错，我很高兴。此外，他们还设计了一个方法，使得我针对读者发现的问题做任何改动都不会有困难。

**10. 斯尔维欧·列维 (Silvio Levy), MSP :** 您能否讲讲 pdf, ePUB 等电子书与 TAOCP 纸质书的不同。各自有什么优缺点？

**高德纳 :** 纸质书更重，但是他们不需要电池或插电。他们总是在那里，我不需要启动，就可把几本书同时打开。我可以在页边空白处做注解，也可以用不同颜色做标记。我希望10年后电子书也有类似的功能。我习惯翻动书页，徜徉其中，远远超过在一个电子书中的感觉，但是也许我的孙子会有截然不同的看法。电子书的优势是有搜索能力。

印刷书中的索引，好处是可以突出重点。但该索引也可出现在电子书中，单击索引就可以跳到索引所指向的页面。今天的电子书不方便设置书签，不方便返回几分钟前你阅读的页码，特别是在你点击一个网络链接后，想再回到原先阅读的地方。但功能必定会不断完善，今天的电子设备也会越来越好。

我期待未来的电子书有专家们的评注。就像电影 DVD 的导演评价。

**11. 彼得·歌顿 (Peter Gordon), 艾迪生-韦斯利出版公司 (Addison-Wesley) (已退休) :** 如果在当初 TAOCP 首次出版时，如今的电子书的特性和功能已全部具备，您会用不同的方式写书吗？

**高德纳 :** 嗯，我想不会有太大差别。我将不得不考虑用颜色做每件事，考虑交互式的图表、表格、方程和练习。一个人无法使用电子书的所有功能。

过去只能用黑白字体，只能用一定尺寸的打印纸，不过我很庆幸在过去50年里完成了3000页的书稿。

**12. 尤迪·曼波 (Udi Manber), 谷歌 :** 早期的 TAOCP 把计算机编程建成了计算机科学。它们引入了必要的严谨。今天的大多数应用程序都与人类有关——社会交互、搜索、娱乐等。请问，关于如何把严谨引入到这些新的应用中，您有什么建议吗？

**高德纳 :** 数值计算在计算机诞生时是核心应用，

当然，现在所占的比例越来越小了，但我也不赞成人们过多地集中到那些大的应用上。

我的 METAFONT 工作使得我“进入”一些应用，在这些应用里，无法定义出“正确性”。我怎么知道我的字母 A 程序会不会产生一个正确的图像？我不会知道，我已经学会接受不确定性。另一方面，当我执行一个程序来解释 METAFONT 规范并绘制关联的位图<sup>5</sup>，有很多严谨的空间。字体渲染的算法是我见过的最有趣的程序之一。

作为谷歌、Adobe 和其他公司产品用户，我知道严谨在地图数据的操作、交通数据、像素数据、语言数据及元数据等中都非常需要。此外，很多处理是用分布式、分散算法来完成的，它比 20 世纪 60 年代任何人想象得更加严谨。

所以我不能说，严谨已从计算机科学的场景消失了。可是，我真的希望，谷歌、Adobe 和苹果的程序员好好钻研一下，当我不用 Linux 的时候，防止他们的系统把我家用的计算机弄瘫痪了。

总的来说，我赞同你说的不要减少严谨，相反，应该增加严谨，这很重要。不能在“基线”上定义正确性，不应让人们认为在一个正确性极为重要的大系统里面不存在中间的层次。系统的鲁棒性和质量由每一个薄弱的环节决定。

另一方面，我当然不认为什么事情都应该数学化，也不赞成凡是涉及计算机的东西都是计算机科学的一个分支。许多重要的软件系统的部件并不一定需要极客人才；相反，在理想情况下，多学科应该相互协作，因为各种各样东西的互补，生活才会如此快乐。差异性万岁！

事实上，我自己在严谨性方面也不彻底。我很少给出一个正式过程来证明我的程序是正确的。我一旦构建了一个非正式的证明，只要说服了我自己，我就没有更多的兴趣了。例如，我没有兴趣定义 TeX 的正确意味着什么，或者用形式化方法验证我的 550 页程序确实没有故障。我知道出现异常的结果是可能的。

比如用户指定了 1 英里宽的页宽，或者用到含有无穷多个零的常数。我已经尽力避免灾难性的崩溃，但我没有检查每个加法运算的可能溢出。

我的数学基础有一个根本性的缺失就是缺乏算法分析。例如，考虑一个数字序列排序的程序。由于弗洛伊德、霍尔和其他人的工作，我们有了正式的语义定义和工具。通过它，我们可以验证排序总能实现。我的工作超越正确性验证，集中到诸如程序运行时间的分析：我写下递推公式 (recurrence)，想表示对任意输入数据程序所做的平均比较次数。我百分之百地确定我的递推公式正确地描述了程序的性能，我所有的同事也都同意我的递推公式“明显”而且有效。但是，我还没有一个正式的工具，通过它来证明我的递推公式是正确的。我一点也不理解我的推理过程！我的学生莱尔·莱姆肖曾在他的论文（1979 年）中试图构建合适的基础，但是这个问题本质上很困难。不过，我想我还是不必过于担忧这种情况。

**13. AI·阿霍 (AI Aho)，哥伦比亚大学：**大家都知道图灵机是顺序计算的通用模型。但让我们考虑一下与周围保持互动的各种反应分布式系统，比如互联网、云计算甚至大脑，请问是否存在一个关于这类系统的通用计算模型？

**高德纳：**我在逻辑问题方面能力不强，所以 TAOCPL 对这类问题仅仅是蜻蜓点水。TAOCPL 的计算模型（比如在卷一的第 4~8 页）考虑到针对单处理器的“反应过程”（即“计算方法”），但在完成了第七章之后，我一直打算讨论第八章的递归程序和其他合作进程。1964 年弗洛伊德在其了不起的综述文章中所介绍的借助半自治代理的漂亮的上下文解析模型，对我的影响很大。

我想把第一卷中计算的集合论模型理论的计算模型扩展到你刚才提到的问题上。它们也许可以对这个问题的启发。

但完全分布式的过程超出了我的书的范围和我

<sup>5</sup> 位图图像(bitmap)，亦称为点阵图像或绘制图像，是由像素的单个点组成的。这些点可以进行不同的排列和染色以构成图样。当放大位图时，可以看见赖以构成整个图像的无数单个方块。——编者注

自己的理解能力。很长一段时间，我一直想了解蚁群是如何执行令人难以置信的有组织的任务的，这也许有助于理解人类的认知。然而，不断入侵我家的蚂蚁搞得我焦头烂额，只好作罢。

**14. 盖·斯蒂尔 (Guy Steele), Oracle 实验室 :**您和我都对程序分析感兴趣，我想问，人们在不执行程序的前提下能知道什么呢？类型理论和霍尔逻辑是这种推理的两种形式，而且您对使用数学工具来分析算法的执行时间已经有很大的贡献。您认为在程序分析领域，目前需要解决的问题是什么？

**高德纳 :**我肯定你不是真的对通过程序执行来分析算法持否定态度。你和我都想知道程序的功能并执行它们。通常执行的结果跟我们认为的假设相左。

寻求更好的方法来验证程序是一个计算机科学的著名难题。正如我跟前面尤迪所说的，我特别支持开发更好的技术以避免程序崩溃。

刚才我正在写卷四 B 部分，讨论了可满足性算法。这是工业上的一个重要问题。几乎没有人知道现代启发式算法为什么能行或者为什么会失败。大多数已经被证明的重要技术起初正是由于错误的原因而造成的。

如果我能选择的话，我希望像你这样的人能把很多精力投入到我现在才意识到的一个问题上：今天的多线程计算机程序员需要新的工具，以使链接数据结构更加有利于高速缓存。人们可以在很多情况下启动一些辅助的并行线程，目的是预测未来的主要计算线程将要做的内存存取，然后把这些数据提前加载到高速缓存上。然而，对一个普通程序员包括我来讲设置好这些事情都太过繁复。

**15. 罗伯特·塔扬 (Robert Tarjan), 普林斯顿大学 :**在算法设计与分析方面，未来的工作最有前途的方向是什么？您看到了哪些有趣并且重要的问题？

**高德纳 :**关于满足性问题，我的手稿中已经提

到了 25 个研究问题，其中大部分还不为理论界所知。因此，很多问题大概可以在卷四 B 完成之前回答。新问题无处不在，经常出现。但你所指的当然是更一般性的问题。

一般来说，我会找一些算法，他们相对于问题的规模来讲，执行的速度很快。而这些问题的规模  $n$  是可控的（译者注，问题的规模  $n$  不是无限大的）。今天的论文大部分致力于越来越大的问题的算法，但它们仅当  $n$  超过了宇宙大小的时候才有用。

在某种意义上，这种论文让我的日子更好过，因为我没有在 TAOCPP 讨论这些方法。我不反对纯理论的研究，它们大大提高了我们处理实际问题的能力，就其本身而言也十分有趣。所以我有时候玩渐进游戏。但我也愿意看到更多的实用算法。

要关心计算机上运行的算法的效率问题。作为斯坦福大学 GraphBase 项目的一部分，我实现了 4 个计算图的最小生成树的算法，其中之一就是你和查理顿河·卡普开发的漂亮的算法。虽然我希望你的方法是赢家，因为它只检查主要数据的次数，比其他的算法次数少一半，但是实际上它比 Kruska 令人敬仰的方法差  $1/3 \sim 1/2$ 。部分原因是糟糕的缓存作用，但主要原因是  $O$  符号里面的常数因子太大了。

**16. 弗兰克·鲁斯科依 (Frank Rusky), 维多利亚大学 :**您能否就做重要问题的重要性给些评论？我的感觉是计算机科学研究、资金和学术招聘等方面越来越关注短期问题，在他们心中有一种经济上的考虑。您同意这种观点吗？这是一个不好的趋势吗？您觉得有办法缓解这种状况吗？

同样，您认为单个研究者会不复存在吗？现在很多论文有多个作者。一篇文章有 5 个作者都很常见。但当我深究其中细节，发现通常似乎只有一个人或两个人贡献了新想法，其他人要么是导师，要么是资金支持者，可能还有其他人员。我很确定欧拉<sup>6</sup>没有跟五个人共同发表过任何论文。是什么原

<sup>6</sup> 莱昂哈德·欧拉(Leonhard Euler)，瑞士数学家、自然科学家，是18世纪数学界最杰出的人物之一，也是最多产的数学家。他平均每年写出800多页的论文，还为课本写了大量的力学、分析学、几何学等内容。他的著作《无穷小分析引论》《微分学原理》《积分学原理》等已成为经典著作。——编者注

因导致这样的趋势，它对历史有何负面作用，我们应该做些什么来改变它？

**高德纳：**我害怕有人问有关经济学的问题，我根本不了解这个问题。如同我不知道人们为什么会花钱买东西。我愿意相信，一些经济学家有足够的智慧使世界运行一段时间，但他们的理由超出了我的理解。

我只是写书。我想讲述的故事似乎是重要的，至少对于极客。我从没费心考虑过营销，或者卖什么，除非我的出版商要求我回答问题，就像我现在做的！

三年前，我发表了《论文选集：乐趣和游戏》，共750页，讲的全是不重要的问题。在许多方面，我能在世界历史的某一个时间段内，能够写出这本书，所获得的满足感比我看到TAOCP目前不错的状况要大得多。

我已到了可以被称为“坏脾气”老人的年龄，也许这就是我为什么对你所提出的这种令人震惊的趋势的担忧有强烈同感。当人们通过衡量我的工作有多大程度上影响了华尔街来评定我的工作质量时，我感到极度不安。

每个人似乎都明白，天文学家们从事天文学研究是因为天文学是有趣的。为什么他们不明白我做计算机科学研究是因为计算机科学是有趣的，而不是考虑它是否让任何人挣到钱？其原因可能是，并不是每个人都是极客。

关于联合署名，你肯定是对的。欧拉在18世纪从不随意署名。事实上，我也想不出那时候有两个作者的数学文章，直到哈蒂和李图伍德在20世纪初开始一起工作。

就我自己而言，我最早的两篇文章是联合署名的，因为其他作者做了理论部分，我写了计算机程序来验证它。其他两篇文章是与ALGOL语言有关，与ACM委员会一起完成的。在其他一些文章里，当时我在加州理工学院，我提出了理论，与我的学生共同编写了计算机程序来验证它。有一篇文章是与迈克·格雷、罗恩·格雷厄姆和大卫·约翰逊一起完成的，他们提出理论，我给予解释。你和我在

2004年写过一篇关于递归程序的文章，我们平分了工作。

多作者现象还没有像物理学和生物学那样对计算机科学产生很大的冲击。我了解，汤普森-路透索引200多篇文章就会有1000多位作家。这些文章仅仅是一年内发表的。当我在TAOCP引用一篇文章时，我喜欢提到所有的作者并在索引里面列出他们的名字。如果计算机科学跟其他学科那样有太多的联合作者，也许我这么做就不太可行。

合作工作令人兴奋，并且能够取得单个人无法取得的成果。但正如你所说，作者就是作者，而不是挂名。你提到思想史，我认为发现方法往往比识别发现者更重要。不过，功劳应该分明。

我猜想多作者现象主要是由于贫乏的奖励政策。无知的管理者似乎是基于文章的数目来评定薪资水准和做出提升决定。

我们能做什么？正如我所说，我没有能力处理经济。我经历了拒绝随大溜的生活，对抗我不赞同的趋势。我经常拒绝将我的名字添加到一篇论文中。但我已经有了安逸的生活，而年轻人可能会被迫屈服于同行的压力。

**17. 安德鲁·宾斯托克 (Andrew Binstock), Dr.Dobb's 杂志：**在2012年ACM图灵百年纪念会上，您说您开始相信 $P=NP$ 。您能不能解释您对这个问题的思考，您是怎么想到的，以及您有这些愈来愈强的想法是否令您自己也吃惊？

**高德纳：**像你说的，我相信 $P=NP$ ，即存在一个整数 $M$ 和算法 $A$ ，将用 $n^M$ 基本步骤解决所有属于 $NP$ 的 $n$ 位问题。

我的一些理由是天真的。很难相信 $P \neq NP$ ，还有这么多聪明的人也没有发现为什么。另一方面，如果你设想一个数字 $M$ 是有限的但特别大，就像我的论文处理有限问题中的数字 $10 \uparrow \uparrow \uparrow 3$ 那样，则会有非常多的可能算法，对给定的 $n$ 位，执行 $n^M$ 逐位，或加法，或转移操作。而且真的很难相信，所有这些算法都会失败。

我认为等式 $P=NP$ 不会成为有益的东西，即使它被证明了。因为这样的证据几乎肯定会是非建

设性的。虽然我觉得  $M$  可能存在，但我也认为人类永远不会知道这个值。我甚至怀疑，没人会知道  $M$  的上限。

数学里到处是这样的例子，有些东西被证明是存在的，但并不能告诉我们如何求得这些东西。一个算法存在的知识完全不同于一个实际的算法知识。

更重要的是，罗伯逊和西莫已经证明了图论中的一个著名定理：在  $\text{minor}$  运算下封闭的任何类  $C$  的图，都具有有限数目的  $\text{minor-minimal}$  图（图的  $\text{minor}$  是通过删除节点、边或者缩小一个点的边得到的图。一个  $C$  的  $\text{minor-minimal}$  图  $H$  是一个图，其较小的  $\text{minors}$  都属于  $C$ ，尽管  $H$  不属于  $C$ ）。因此存在一个多项式时间的算法用于决定一个给定的图是否属于  $C$ 。这个算法检查的是  $G$ ，不包括任何  $C$  的  $\text{minor-minimal}$  图作为其  $\text{minor}$ 。

但我们并不知道那是什么算法，因为除少数特殊类  $C$  外， $\text{minor-minimal}$  图通常未知。该算法存在，但目前还不清楚能否在有限时间内发现它。

当我阅读洛瓦斯 (Lovász) 的文章学习罗伯逊和西莫定理时，发现该定理的结果令我惊讶，它打破了平衡，使得我倾向于  $P = NP$  的假设。

这个故事的寓意是，人们应当区分已知（或可知）多项式时间算法和任意多项式时间算法。人们针对可满足性可能永远无法实现的一个多项式时间算法，即使  $P$  恰好等于  $NP$ 。

**18. 杰弗里·沙理特 (Jeffrey O Shallit)，滑铁卢大学：**决策方法、自动定理证明、证明助理已经在许多不同的领域取得成功：比如，组合恒等式的 Wilf-Zeilberger 方法和罗宾斯猜想。您觉得 100 年后，定理发现和证明会是什么样子？会和今天一样，还是更加自动化？

**高德纳：**除了经济，我也怕有人问我关于未来的事情，因为我是一个众所周知的糟糕的先知。不过我可以尝试着回答你的问题。

假设有 100 年的可持续文明，我确信会有很大比例的定理（甚至 38.1966%）由计算机辅助发现。而且还会有不小的百分比 (0.7297%) 由计算机验证得到证明，这可能是凡人无法理解的。

在我的博士论文 (1963 年) 里，我看过由计算机生成的小的有限射影平面的示例，并使用此数据构建了一个不知道其类型的无限的多平面。十年后，通过研究史蒂夫·库克的自动机，我发现高德纳-莫里斯-普拉特算法能够在线性时间内识别连接的回文。

不过，几个月前，我试图做类似的事情，但是失败了。我有一个 5000 步机器发现的证明，想证明  $\text{flower snark}$  图的边缘不能着三色，我想猜透机器是如何想到的。虽然几天后我放弃了，但我还是认为有可能研制新的计算机证明工具，以产生令人惊奇的效果。

我注意到，计算 Erdős<sup>7</sup> 的差异常数，由于蒂莫西·高尔斯的 Polymath (Tim Gowers' Polymath) 项目而享有盛名。许多数学家通过互联网合作，对可满足性测试算法构建了一个有意义的测试集。我第一次尝试计算它用了 49 个小时。两周后，我减少到了 2 小时以内，但仍需 2000 万个步骤的证明。目前我还看不到有什么办法可以减少到几千个步骤以内。

**19. 斯科特·阿伦森 (Scott Aaronson)，麻省理工学院：**您会建议其他科学家放弃使用电子邮件，就像您那样吗？

**高德纳：**我的情况跟别人不一样，因为当我不被打断时，我可以尽全力做好工作，我吃饭、睡觉、写书，或多或少像隐士那样，每天花大量的时间阅读文档和其他人的代码。正如我在我的主页说明的那样，很多人停留在事情的表面，但是我想深入到底层。

因此，我并不提倡任何愿意交流的人放弃电子邮件。实际上我在这方面经常叨扰别人，向他们提问，麻烦他们。

我欢迎大家用电子邮件报告 TAOCP 中的问题，

<sup>7</sup> Erdős 数是用来衡量其他数学家与已故的发表论文数比欧拉还多的数学家保罗·爱多士合作程度的指数。对爱多士来说，Erdős 数为 0，如果数学家 A 直接与爱多士联手发表过论文，那么 A 的 Erdős 数为 1，如果数学家 B 没有与爱多士合著论文，但与 A 一起发表过论文，则 B 的 Erdős 数为 2，以此类推。——编者注

我会尽快地改正。其他未经请求的消息发送给 bit-bucket, 否则标记为 /dev/null。

**20. J.H 奎克 (J.H. Quicker), 博客:** 为什么这个采访被称为“二十个问题”, 我看只有 19 个问题呀?

**高德纳:** 我被难住了。不, 等等, 我知道了, 拉迪亚问了两个。

巧合的是, TAOCP 也包含 4500 个问题和解答。■

### 术语解释

1. 《计算机程序设计艺术》简称 TAOCP, 是高德纳编著的七卷本著作。作者因此获得 ACM 颁发的 1974 年度图灵奖。该书已有三卷翻译成中文, 分别为《基本算法》《半数值算法》和《排序与查找》(《计算机程序设计艺术》(第三版), 国防工业出版社, 2002)。该书 1999 年底被 *American Scientist* 列为 20 世纪最佳 12 部学术专著之一(与狄拉克的《量子力学》、爱因斯坦的《相对论》、曼德布罗特的《分形论》、鲍林的《化学键》、罗素和怀特海德的《数学原理》、冯·诺依曼和摩根斯坦的《博弈论》、维纳的《控制论》、伍德沃和霍夫曼的《轨道对称性》、费曼的《量子电动力学》等科学史上的经典著作并列)。

此书共 7 卷, 总目录如下:

- 第 1 卷 基本算法  
(Vol 1: Fundamental Algorithms)
- 第 2 卷 半数值算法

(Vol 2: Seminumerical Algorithms)

- 第 3 卷 排序与查找  
(Vol 3: Sorting and Searching)
- 第 4 卷 组合算法  
(Vol 4: Combinatorial Algorithms)
- 第 5 卷 语法算法  
(Vol 5: Syntactic Algorithms)
- 第 6 卷 上下文无关语言理论  
(The Theory of Context-Free Language)(计划中)
- 第 7 卷 编译技术  
(Compiler Techniques)(计划中)

2. 自然语言编程 (literate programing) 是高德纳引入的编程方法。程序就是用自然语言, 比如英语给出程序逻辑的解释, 加上宏的片段和传统的源代码。在此基础上可以生成一个可变异的源代码。这种方式代表了从传统地服从计算机的方式和顺序来写程序, 到可以按照程序员的逻辑和思维顺序来编写程序的转变。

**致谢:** 本文有些信息综合了维基百科和百度百科的相关条目, 在此致谢。



周明

CCF 高级会员、杰出演讲者、本刊编委。微软亚洲研究院首席研究员。主要研究方向为自然语言处理、机器翻译、文本挖掘、信息检索等。  
mingzhou@microsoft.com

## 2014 CCF 青年科学家奖推荐启动

2014 CCF 青年科学家奖推荐启动。该奖是为激励和表彰中国计算机及相关领域在科学研究、技术发明、系统开发及应用推广等方面有突出成就和重大贡献的青年科学家而设。2014 年度提名现已开始受理。

在科学研究、技术发明、系统开发及应用推广等方面有突出成就和重大贡献, 1974 年 1 月 1 日以后出生的 CCF 会员可参加评选。

推荐方式采用(联名)推荐方式。由不少于三名 CCF 理事、监事或高级会员联名推荐(也可混合推荐), 填写推荐表格。推荐者中须有 1 人非常熟悉被推荐人的成就和贡献情况, 称为主推荐人。主推荐人须能对被推荐人情况进行书面或口头陈述, 并回答评奖(分)委员会的问题。每一位推荐人推荐人数不得超过两名。推荐从即日起至 11 月 10 日止。详情和推荐表下载请登录 [www.ccf.org.cn/awards](http://www.ccf.org.cn/awards)。