TURING 图灵计算机科学丛书

Pearson

计算机程序设计艺术
卷4A：组合算法（一）

[美] 高德纳（Donald E. Knuth）◎著
李伯民 贾洪峰 ◎译

The Art of Computer Programming
Vol 4A: Combinatorial Algorithms
Part I

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

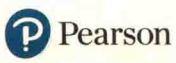
“计算机科学既壮观又幽美，我尝试尽自己所能，以最恰当的方式来解释我所了解的某些片断。很显然，我自己并没有任何超自然能力，但的确很喜欢讲述那些似乎静静地等待着人们去讲出来的故事。写书跟讲故事十分类似。”

——— **图灵访谈之专访 Donald E. Knuth**

《计算机程序设计艺术》系列著作被公认为是对经典计算机科学的权威论述，数十年来，一直是广大学生、研究人员和业内人士学习程序设计理论和实践的无价之宝。这一宏伟浩大的工程始于 1962 年，计划出版 7 卷，目前已经出版了 4 卷。

《计算机程序设计艺术》堪称计算机科学领域的瑰宝。从事研究的人惊艳于其精美优雅的分析，而普通程序员则一直在卓有成效地利用书中提供的各种方案解决日常问题。这些书展现了作者的博观、清晰、精确和幽默，所有的人都钦佩不已。

回复“TAOCP”查看相关书单



www.pearson.com

图灵社区：iTuring.cn
微　　博：@图灵教育　@图灵社区

分类建议 计算机 / 程序设计

人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-51287-1

定价：228.00元

TURING 图灵计算机科学丛书

计算机程序设计艺术
卷4A：组合算法（一）

[美] 高德纳（Donald E. Knuth）◎著
李伯民 贾洪峰 ◎译

The Art of Computer Programming

Vol 4A: Combinatorial Algorithms
Part I

人民邮电出版社
北京

图书在版编目（CIP）数据

计算机程序设计艺术. 卷4. A, 组合算法. 一 /
(美) 高德纳著 ; 李伯民, 贾洪峰译. -- 北京 : 人民邮
电出版社, 2019.6
（图灵计算机科学丛书）
ISBN 978-7-115-51287-1

Ⅰ. ①计… Ⅱ. ①高… ②李… ③贾… Ⅲ. ①程序设
计②电子计算机－算法分析 Ⅳ. ①TP311.1②TP301.6

中国版本图书馆CIP数据核字(2019)第095378号

内 容 提 要

《计算机程序设计艺术》系列被公认为计算机科学领域的权威之作，深入阐述了程序设计理论，对计算机领域的发展有着极为深远的影响。本书是该系列的第4卷A，书中主要介绍了组合算法，内容涉及布尔函数、按位操作技巧、元组和排列、组合和分区以及所有的树等。

本书适合从事计算机科学、计算数学等各方面工作的人员阅读，也适合高等院校相关专业的师生作为教学参考书，对于想深入理解计算机算法的读者，是一份必不可少的珍品。

◆ 著　[美] 高德纳（Donald E. Knuth）
译　李伯民　贾洪峰
责任编辑　傅志红
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
天津翔远印刷有限公司印刷
◆ 开本：787×1092　1/16
印张：46.5
字数：1371千字　2019年6月第1版
印数：1－4 000册　2019年6月天津第1次印刷
著作权合同登记号　图字：01-2011-2961号

定价：228.00元

读者服务热线：(010)51095183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

版权声明

前　言

欲把一切好的内容都装进一本书中，显然
是不可能的，哪怕只是想比较全面地
涉猎主题的某些方面，多半也会导致篇幅急剧增长.
——杰拉尔德·弗兰德，“编者之角”（2005）

卷 4 的书名是组合算法，我在拟书名时特别想给它加个副标题：我最喜爱的程序设计类型. 但是，编辑决定淡化这种个人感情色彩，因此没有这么做. 不过事实上，具有组合风格的程序始终是我所偏爱的.

另一方面，我经常惊奇地发现，“组合”一词在许多人的头脑中意味着计算有难度. 其实，词典学家塞缪尔·约翰逊在其著名的英语词典（1775）中已经说明，“组合”的名词之意“现在普遍用法不当”. 同事们对我讲述种种不利事件时，总会说“事态的组合使我们无功而返”. 对我而言，组合唤起的是纯粹喜悦之情，它却给其他许多人带来一片惊恐，原因何在?

的确，组合问题经常同非常巨大的数字联系在一起. 约翰逊的英语词典中还引述了百科全书编撰者伊弗雷姆·钱伯斯的一段话：用 24 个字母的字母表构成长度小于等于 24 的词，其总数高达 1 391 724 288 887 252 999 425 128 493 402 200. 在钱伯斯的叙述中用 10 代替 24，相应的数量减少到 11 111 111 110；当这个参数减少到 5 时，相应的数量仅有 3905. 所以，如果问题的规模从 5 增加到 10，再增加到 24 甚至更大，就必然出现“组合爆炸”.

在我这一生中，计算机一直以惊人的速度变成越来越强大的工具. 及至我写下这些字句时，我知道所用笔记本电脑的运算速度，比我当初着手编写这套书使用的 IBM Type 650 计算机快 10 万倍以上，而且现在这台电脑的存储容量也比那时的计算机大 10 万倍以上. 至于明天的计算机，运算速度还会更快，存储容量也会更大. 然而，这些惊人的进展并没有降低人们对于解答组合问题的渴望，情况恰好相反. 我们从前无法想象的如此快速的计算能力，如今提高了人们的期待，并且激起了我们更强的欲望——事实上，因为只要组合元素的数量 n 增加 1，组合问题的规模就可能增加 10 万倍以上.

可以把组合算法非正式地定义为诸如对排列或图这些对象进行组合计算的高速处理方法. 一直以来，我们都在试图寻找特定约束条件下的组合模式或排列的最佳方案. 这类问题的数量十分巨大，编写这种程序的技巧也特别重要，而且富有吸引力，因为只要有一个好主意，有时就可能节省几年乃至几百年的计算机时间.

毫无疑问，组合问题的优异算法可以带来巨大回报，这个事实引领技术水平突飞猛进. 以往认为很难处理的问题，而今可以迎刃而解；过去许多以精巧著称的算法，现在锦上添花. 大约从 1970 年开始，计算机科学家们经历了所谓的“Floyd 引理”现象：看似需用 n^3 次运算处理的问题，实际上用 $O(n^2)$ 次运算就能求解；看似需用 n^2 次运算处理的问题，实际上用 $O(n \log n)$ 次运算就能求解，而且通常 $n \log n$ 还可以减少到 $O(n)$. 至于那些难度更大的问题，求解时间则可以从 $O(2^n)$ 减少到 $O(1.5^n)$，进而减少到 $O(1.3^n)$，等等. 一般说来，剩下的问题依然是很难的，但是我们已经发现它们有一些非常简单的重要特例. 有许多组合问题，我曾经以为在我有生之年不会看到它们的答案，如今却已经得到解决，而且那些突破主要归功于算法的改进，而不是计算机处理器速度的提高.

截至 1975 年，这方面的研究工作进展神速，在主要计算机学术期刊上发表的大量论文竟然都是有关组合算法的. 同时，这些进展不仅是由计算机核心领域的研究人员取得的，而且大量

成果也来自电子工程、人工智能、运筹学、数学、物理学、统计学等其他学科的研究人员. 我曾想尽快完成《计算机程序设计艺术，卷 4》的写作，但是我仿佛是坐在沸腾的冒着气泡的水壶旁边，不得不面对另一类组合爆炸——层出不穷的新主意的大爆炸!

这几卷书诞生于 1962 年之初，当时我天真地拟好一共 12 章的提纲. 未经深思熟虑，我便决定用简短的一章来讨论组合算法，心想:“嘿，看看! 大多数人在利用计算机处理数字，而我能够编写处理模式的程序!”当时，对于已知的每个组合算法都能很容易给出一个非常完备的描述. 即使到了 1966 年，当我把这本已经过度膨胀的书写就约 3000 页初稿的时候，第 7 章的内容还不到 100 页. 我绝对不会想到，当初预计作为“沙拉”的小菜最终竟然升格成了一道主菜.

自 1975 年兴起的组合学热潮，在越来越多人的推动下一浪高过一浪. 新思想不断改善着旧思想，但是很少取而代之，或者使之过时. 所以，我自然不得不抛弃昔日怀抱的希望，我已经不可能围绕这个领域撰写一部一劳永逸的书，把一切题材组织得井然有序，让它成为每个需要解决组合问题的人手中的“灵丹妙药”. 而今，各种各样可用的技术如雨后春笋般层出不穷，以致对于任何的支节问题，我几乎都不能宣称“这是最后的解决方案: 故事终结了”. 相反，我必须把自己严格限制在仅仅阐明一些最重要的原理上，而这些原理大体是迄今我见过的所有有效组合方法的基础. 现在，我为卷 4 积累的原始资料已经超过卷 1 至卷 3 全部资料的两倍.

面对这些堆积如山的资料，不言而喻，我必须把计划编写的“卷 4”变成若干卷. 现在呈现在读者面前的是卷 4A. 倘若我的健康状况能够保持下去，日后陆续会有卷 4B 和卷 4C，或许还会有卷 4D、卷 4E……;(天知道?)当然肯定不会出现卷 4Z.

现在的计划是尽我所能，通过自 1962 年以来积累的文档，系统地讲述我深信仍然有待进一步讨论的组合方法. 我无意追求完美，但是一定要把应有的荣誉归于所有那些提出种种关键思想的先驱们; 所以关于历史情节，我不吝惜笔墨. 除此之外，凡是我认为在今后 50 年内仍然具有重要性的材料，以及某些能够用一两段文字简要叙述的内容，我都不会割爱. 反过来，我没有把那些需要长篇累牍证明的艰深话题收录书中，除非它们确实是基础性的.

诚然，组合算法这个领域非常广阔，我无法顾及它的所有方面. 那么，我忽视了什么最重要的东西? 我以为自己最大的盲点是在几何学方面，因为我所擅长的始终是表现和推演代数公式而非处理空间对象. 所以，我在这几本书中不准备讨论同计算几何有关的组合问题，如球的密堆积，或者 n 维欧几里得空间中数据点的聚类，甚至也不讨论涉及平面斯泰纳树的问题. 更重要的是，我力求避开多面体组合学，以及主要基于线性规划、整数规划或半定规划的各种方法. 对于那些题目，在有关这个主题的很多其他书籍中已有充分论述，而且它们依赖于几何直观知识. 对我而言，单纯从组合问题展开讨论更容易理解.

我还必须承认，对于仅在渐近意义下有效的那些组合算法，以及算法的优异性能在问题规模超乎想象时才开始显现的组合算法，我是不太以为然的. 现在有讨论这类算法的大量出版物. 有些人喜欢思考极限问题，认为它是一种智力挑战并且能够带来学术声誉，这是我能理解的，但是对于我自己在实际程序中不予考虑的任何方法，这本书一般只作轻描淡写.(这条规则的运用自然也有例外，特别是对于那些处在主题核心的基本概念，就另当别论. 某些实用价值虽说不大，但是确实非常优美或者包含深刻见解的方法，令我难以割舍; 另外还有一些方法，我则把它们作为反例引用.)

此外，在这套书的前几卷中，我特意几乎完全专注于*顺序算法*，尽管计算机的并行计算能力日益增强. 关于并行计算，我无法断定哪些思想可能在今后 5 年或者 10 年内会很有用，更不用说今后 50 年内了，所以我乐于把这样的问题留给那些比我聪颖的人. 来日，具有才华的程序员们究竟应该掌握怎样的顺序算法知识，单是弄清这个问题就足以检验我自身的能力了.

在安排如何陈述这些材料时，我需要做个重要决定：是按问题还是按方法组织它们. 例如，卷 3 的第 5 章专门讨论了一个问题，即数据排序；对于这个问题，我从不同方面应用了二十几种方法. 相比之下，组合算法涉及许多不同的问题，而解决问题的方法则少之又少. 我最终决定，采用混合策略能够比任何一种单纯的方法更好地组织材料. 于是，这几本书中就采纳了兼收并蓄的原则，例如，7.3 节处理求最短路径问题，7.4.1 节处理连通性问题. 其他很多节则专门讨论基本方法，如布尔代数的应用（7.1 节）、回溯（7.2.2 节）、拟阵理论（7.6 节）或者动态规划（7.7 节）. 至于著名的流动推销员问题，以及同覆盖、着色和填充有关的其他经典组合问题，没有单辟小节讨论，但是当用不同方法处理这些问题时，它们多次出现在书中不同的地方.

我已经提到过组合计算技术的巨大进步，但是我并非暗示人们已经解决了所有的组合问题. 正值计算机程序的运行时间不断攀升之际，程序员们不能指望从本书中找到无坚不摧的“银弹”. 这里描述的方法通常会比一个程序员早先尝试的方法快得多，但是，我们不得不面对这样一个现实：组合问题正在迅速演变成为巨大的问题. 我们甚至可以严格证明，就连一个自然的细小问题也不存在一种现实的可行解，尽管原则上它是可解的（见 7.1.2 节的拉里 · 斯托克迈尔和阿尔伯特 · 迈耶定理）. 在另外一些情况下，我们甚至不能证明一个给定问题不存在适合的算法，知道的只是可能没有这样的算法，因为任何有效的算法将产生一种令人满意的方法，足以解决一大堆难倒世上无数杰出专家的问题（见 7.9 节关于NP 完全性的讨论）.

经验表明，新的组合算法将会不断涌现，用以解决组合学中新出现的问题以及处理老问题的变形或特例；同时，人们对于这种算法的期望也会与日俱增. 当程序员们面对这样一些挑战时，计算机程序设计艺术将会达到一个又一个新高度，不过，今天的方法多半仍旧是不可或缺的.

本书中大部分内容是相对独立的，但是时常同卷 1 至卷 3 讨论的主题关联. 在前面几卷中，对机器语言程序设计的底层细节作过广泛深入的讨论，所以本卷介绍的算法通常是在抽象级别上说明，与任何具体机器无关. 然而，组合程序设计的某些方面的确同过去从未出现过的底层指令有着密切联系. 在出现这种情况的地方，书中相应的所有例子都是基于 MMIX 计算机的，这款计算机取代了卷 1 前几版中定义的 MIX 计算机. 关于 MMIX 的详细材料在《计算机程序设计艺术：MMIX 的增补》（*The Art of Computer Programming*, Volume 1, Fascicle 1）中介绍，其中包含 1.3.1 节，1.3.2 节，等等. 此外，这些材料也可以从互联网上获取，配套的汇编程序和模拟程序同样可以从网上下载.

另外一种可以从网上下载的资源是称为《斯坦福图库》（Stanford GraphBase）的一套程序和数据，它在本书的例子中经常被引用. 我鼓励读者多利用它，对于学习组合算法，我以为这不失为一种非常有效和愉快的途径.

顺便说一句，我很高兴在写这篇前言时能说明一下，书中自然提到了我的博士论文导师小马歇尔 · 霍尔（1910—1990）的一些成果，以及霍尔的论文导师奥斯丁 · 欧尔（1899—1968）的一些成果，还有欧尔的论文导师托拉尔夫 · 斯科伦（1887—1963）的一些成果. 至于斯科伦的论文导师阿克塞尔 · 图厄（1863—1922）的一些成果，我在第 6 章已经介绍过了.

有几百位读者帮助我查出这卷书几份初稿（它们原先公布在互联网上，后来又印成几册平装书）中遗留的大量错误，谨对他们表示衷心的感谢. 特别是索斯藤 · 达尔海姆尔、马库斯 · 范莱文和乌多 · 维尔穆特三人提出的大量建议，对本书具有特殊的影响. 但是在汇集成书后的字里行间，我担心仍然隐藏着其他错误，而且希望能够尽快予以更正. 因此，我很乐意向首先发现任一处技术性错误、印刷错误或历史知识错误的人支付 2.56 美元. 下面的网页上列出了所有已反馈给我的最新勘误：https://www-cs-faculty.stanford.edu/~knuth/taocp.html.

高德纳，加利福尼亚州斯坦福市，2010 年 10 月

在第 1 版的前言中，我曾经请求读者不要专门注意辞典中的错误.
如今，我反倒希望自己未曾这样说过，而且还要感谢对我的请求置之不理的那些读者.
——诺尔曼 · 萨瑟兰，《国际心理学辞典》（1996）

诚然，我应该对遗留的错误负责. 不过
在我看来，我的朋友们理应发现更多一些错误.
——赫里斯托斯 · 帕帕季米特里乌，《计算复杂性》（1994）

我愿意从事种种不同领域的工作，以便使我的错误更稀疏地分散开来.
——小维克多 · 克利（1999）

关于参考文献的注释. 经常引用的若干学术期刊和会议刊物有特别的代码名称，它们出现在书后的索引中. 但是，在各种 IEEE 会刊的引用中包含一个代表会刊类别的字母代码，置于卷号前面，用粗体表示. 例如“*IEEE Trans.* **C-35**”是指 *IEEE Transactions on Computer*, Volume 35（《IEEE 计算机会刊》，第 35 卷）. IEEE 现在不再使用原来那些简便的字母代码，其实它们并不难辨认：“**EC**”曾经代表“电子计算机”，“**IT**”代表“信息论”，“**SE**”代表“软件工程”，“**SP**”代表“信号处理”，等等；“**CAD**”是指“集成电路和系统的计算机辅助设计”.

如“习题 7.10-00”这样的写法，表示 7.10 节中一道还不知道题号的习题.

关于记号的注释. 对于数学概念的代数表示，简单而直观的约定始终有利于促进科学的发展，尤其是当世界上多数研究人员都使用一种共同符号语言的时候. 可惜在这方面，组合数学当前的事态多少有些混乱，因为同样一些符号在不同的人群中有时完全代表不同的意义；在比较狭窄的分支领域从事研究工作的某些专家，他们也会在无意中引入彼此冲突的符号表示. 计算机科学——它与数学中的许多主题相互影响——应当尽可能地采用内部一致的记号来避开这种危险. 所以，我经常不得不在若干对立的方案中做出选择，虽然明知结果不会令人人满意. 凭借多年的经验以及与同事之间的讨论，以及经常在不同方案之间反复试验，我尽力找出适用的记号，我相信这些将是未来最好的记号. 在其他人尚未认同对立方案时，通常有可能找到可以接受的共同约定.

附录 B 给出了本书使用的所有主要记号，其中不可避免地会包含一些还不够标准的记号. 读者如果偶然遇见一个有些奇怪或者不好理解的公式，通过附录 B 大体可以找到说明我的意图的章节和段落. 不过，我仍然应该在这里举出几个例子，以期引起初次阅读本书的读者的注意.

- 十六进制常数前面冠有一个$^{\#}$符号. 例如，$^{\#}123$ 是指 $(123)_{16}$.
- “非亏减”运算 $x \dot{-} y$ 有时称为点减或饱和减，结果是 $\max(0, x - y)$.
- 三个数 $\{x, y, z\}$ 的中位数用 $\langle xyz \rangle$ 表示.
- 像 $\{x\}$ 这样含单个元素的集合，在书中通常简单地用 x 表示，例如 $X \cup x$ 或 $X \setminus x$.
- 如果 n 是一个非负整数，在 n 的二进制表示中，取 1 的位数记为 νn. 此外，如果 $n > 0$，n 最左边的 1 和最右边的 1 分别用 $2^{\lambda n}$ 和 $2^{\rho n}$ 表示. 例如，$\nu 10 = 2$，$\lambda 10 = 3$，$\rho 10 = 1$.
- 图 G 和图 H 的笛卡儿积用 $G \square H$ 表示. 例如，$C_m \square C_n$ 表示一个 $m \times n$ 环面，因为 C_n 表示 n 个顶点的一个环.

习题说明

这套书的习题既可用于自学，也可用于课堂练习．任何人单凭阅读而不运用获得的知识解决具体问题，进而激励自己思考所阅读的内容，就想学会一门学科，即便可能，也很困难．再者，人们大凡对亲身发现的事物才有透彻的了解．因此，习题是这套书的一个重要组成部分．我力求习题的信息尽可能丰富，并且兼具趣味性和启发性．

很多书会把容易的题和很难的题随意混杂在一起．这样做有些不合适，因为读者在做题前想知道需要花多少时间，不然他们可能会跳过所有习题．理查德·贝尔曼的《动态规划》（*Dynamic Programming*）一书就是个典型的例子．这是一本很重要的开创性著作，在书中某些章后“习题和研究题”的标题下，极为平常的问题与深奥的未解难题掺杂在一起．据说有人问过贝尔曼博士，如何区分习题和研究题，他回答说：“若你能求解，它就是一道习题；否则，它就是一道研究题．”

在我们这种类型的书中，有足够理由同时收录研究题和非常容易的习题．因此，为了避免读者陷入区分的困境，我用等级编号来说明习题的难易程度．这些编号的意义如下所示．

等级　说明

00　极为容易的习题，只要理解了文中内容就能立即解答．这样的习题差不多都可以“在脑子中”形成答案．

10　简单问题，它让你思考刚阅读的材料，决非难题．你至多花一分钟就能做完，可考虑借助笔和纸求解．

20　普通问题，检验你对正文内容的基本理解，完全解答可能需要 15 到 20 分钟．也许 25 分钟．

30　具有中等难度的较复杂问题．为了找到满意的答案，可能需要两小时以上．要是开着电视机，时间甚至更长．

40　非常困难或者耗时很长的问题，适合作为课堂教学中一个学期的设计项目．学生应当有能力在一段相当长的时间内解决这个问题，但解答不简单．

50　研究题，尽管有许多人尝试，但直到我写书时尚未有满意的解答．你若找到这类问题的答案，应该写文章发表．而且，我乐于尽快获知这个问题的解答（只要它是正确的）．

依据上述尺度，其他等级的意义便清楚了．例如，一道等级为 *17* 的习题就比普通问题略微简单点．等级为 *50* 的问题，若是将来被某个读者解决了，可能会在本书以后的版本中标记为 *40*，并发布在互联网上的本书勘误表（网址见第 vi 页）中．

等级编号除以 5 得到的余数，表示完成这道习题的具体工作量．因此，等级为 *24* 的习题，比等级为 *25* 的习题可能花更长的时间，不过做后一种习题需要更多的创造性．等级为 *46* 及以上的习题是开放式问题，有待进一步研究，其难度等级由尝试解决该问题的人数而定．

我力求为习题指定精确的等级编号，但这很困难，因为出题人无法确切知道别人在求解时会有多大难度；同时，每个人都会更擅长解决某些类型的问题．希望等级编号能合理地反映习题的难度，读者应把它们看成一般的指导而非绝对的指标．

本书的读者具有不同程度数学教育和素养，因此某些习题仅供喜欢数学的读者使用．如果习题涉及的数学背景大大超过了仅对算法编程感兴趣的读者的接受能力，那么等级编号前会有

一个字母 M. 如果习题的求解必须用到本书中没有详细讨论的微积分等高等数学知识，那么用两个字母 HM标记. HM 记号并不一定意味着习题很难.

某些习题前有个箭头 ▸，这表示问题极具启示性，特别向读者推荐. 当然，不能期待读者或者学生做全部习题，所以我挑选出了看起来最有价值的习题.（这并非要贬低其他习题！）读者至少应该试着解答等级 10 以下的所有习题，再去优先考虑箭头标出的那些较高等级的习题.

书后给出了多数习题的答案. 请读者慎用答案，还未认真求解之前不要求助于答案，除非你确实没有时间做某道习题. 在你得出自己的答案或者做了应有的尝试之后，再看习题答案是有教益和帮助的. 书中给出的解答通常非常简短，因为我假定你已经用自己的方法做了认真的尝试，所以只概述其细节. 有时解答给出的信息比较少，不过通常会给较多信息. 很可能你得出的答案比书后答案更好，你也可能发现书中答案的错误，对此，我愿闻其详. 本书的后续版本会给出改进后的答案，在适当情况下也会列出解答者的姓名.

你做一道习题时，可以利用前面习题的答案，除非明确禁止这样做. 我在标注习题等级时已经考虑到了这一点，因此，习题 $n+1$ 的等级可能低于习题 n 的等级，尽管习题 n 的结果只是它的特例.

编号摘要：			
		00	立即回答
		10	简单（一分钟）
		20	普通（一刻钟）
▸	推荐的	*30*	中等难度
M	面向数学的	*40*	学期设计
HM	需要“高等数学”	*50*	研究题

习题

▸ **1.** [*00*] 等级“*M15*”的含义是什么?

2. [*10*] 教科书中的习题对于读者具有什么价值?

3. [*H45*] 证明每个简单联通的闭合三维流形都拓扑等价于一个三维球体.

相当一部分有益的艺术尝试
来自于天马行空的想象.
——亨利 · 詹姆斯，“小说的艺术”（1884）

我要感谢我所有的朋友，此时此刻尤其对那些朋友深表谢忱，
他们在经年累月之后终于不再问我："这本书的进展如何？"

——彼得 · 戈梅斯，《拜读圣经》（1996）

我终于向世人交付了一直以来许诺要写的一本书.
关于此书，我担心读者寄予的期望太高.
它的推迟出版，在很大程度上不得不归咎于那些对它表现出
异乎寻常热忱的杰出人士，他们从四面八方向我提供补充资料.

——詹姆斯 · 鲍斯威尔，《约翰逊博士传》（1791）

作者特别感谢 Addison-Wesley 出版公司的忍耐，
从约稿合同签订之日算起，公司对这份手稿等待了整整十年.

——弗兰克 · 哈拉里，《图论》（1969）

厌恶平方根或者代数的普通孩子
从包含相似数学原理的智力游戏中寻找乐趣，
这也许可以成为一个研究课题，即开发出
足以让家族骨相学家多少感到惊奇的数学天赋和创造能力.

——萨姆 · 劳埃德，《谜题的世界》（1896）

来一杯"碧特伯格"！

——碧特伯格啤酒的广告语（1951）

Hommage à Bach.

ASCII 字 符 表

	#0	#1	#2	#3	#4	#5	#6	#7	#8	#9	#a	#b	#c	#d	#e	#f	
#2x		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	#2x
#3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	#3x
#4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	#4x
#5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	#5x
#6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	#6x
#7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	▒	#7x
	#0	#1	#2	#3	#4	#5	#6	#7	#8	#9	#a	#b	#c	#d	#e	#f	

MMIX 操 作 码 表

	#0	#1	#2	#3	#4	#5	#6	#7	
#0x	TRAP $5v$	FCMP v	FUN v	FEQL v	FADD $4v$	FIX $4v$	FSUB $4v$	FIXU $4v$	#0x
	FLOT[I] $4v$		FLOTU[I] $4v$		SFLOT[I] $4v$		SFLOTU[I] $4v$		
#1x	FMUL $4v$	FCMPE $4v$	FUNE v	FEQLE $4v$	FDIV $40v$	FSQRT $40v$	FREM $4v$	FINT $4v$	#1x
	MUL[I] $10v$		MULU[I] $10v$		DIV[I] $60v$		DIVU[I] $60v$		
#2x	ADD[I] v		ADDU[I] v		SUB[I] v		SUBU[I] v		#2x
	2ADDU[I] v		4ADDU[I] v		8ADDU[I] v		16ADDU[I] v		
#3x	CMP[I] v		CMPU[I] v		NEG[I] v		NEGU[I] v		#3x
	SL[I] v		SLU[I] v		SR[I] v		SRU[I] v		
#4x	BN[B] $v+\pi$		BZ[B] $v+\pi$		BP[B] $v+\pi$		BOD[B] $v+\pi$		#4x
	BNN[B] $v+\pi$		BNZ[B] $v+\pi$		BNP[B] $v+\pi$		BEV[B] $v+\pi$		
#5x	PBN[B] $3v-\pi$		PBZ[B] $3v-\pi$		PBP[B] $3v-\pi$		PBOD[B] $3v-\pi$		#5x
	PBNN[B] $3v-\pi$		PBNZ[B] $3v-\pi$		PBNP[B] $3v-\pi$		PBEV[B] $3v-\pi$		
#6x	CSN[I] v		CSZ[I] v		CSP[I] v		CSOD[I] v		#6x
	CSNN[I] v		CSNZ[I] v		CSNP[I] v		CSEV[I] v		
#7x	ZSN[I] v		ZSZ[I] v		ZSP[I] v		ZSOD[I] v		#7x
	ZSNN[I] v		ZSNZ[I] v		ZSNP[I] v		ZSEV[I] v		
#8x	LDB[I] $\mu+v$		LDBU[I] $\mu+v$		LDW[I] $\mu+v$		LDWU[I] $\mu+v$		#8x
	LDT[I] $\mu+v$		LDTU[I] $\mu+v$		LDO[I] $\mu+v$		LDOU[I] $\mu+v$		
#9x	LDSF[I] $\mu+v$		LDHT[I] $\mu+v$		CSWAP[I] $2\mu+2v$		LDUNC[I] $\mu+v$		#9x
	LDVTS[I] v		PRELD[I] v		PREGO[I] v		GO[I] $3v$		
#Ax	STB[I] $\mu+v$		STBU[I] $\mu+v$		STW[I] $\mu+v$		STWU[I] $\mu+v$		#Ax
	STT[I] $\mu+v$		STTU[I] $\mu+v$		STO[I] $\mu+v$		STOU[I] $\mu+v$		
#Bx	STSF[I] $\mu+v$		STHT[I] $\mu+v$		STCO[I] $\mu+v$		STUNC[I] $\mu+v$		#Bx
	SYNCD[I] v		PREST[I] v		SYNCID[I] v		PUSHGO[I] $3v$		
#Cx	OR[I] v		ORN[I] v		NOR[I] v		XOR[I] v		#Cx
	AND[I] v		ANDN[I] v		NAND[I] v		NXOR[I] v		
#Dx	BDIF[I] v		WDIF[I] v		TDIF[I] v		ODIF[I] v		#Dx
	MUX[I] v		SADD[I] v		MOR[I] v		MXOR[I] v		
#Ex	SETH v	SETMH v	SETML v	SETL v	INCH v	INCMH v	INCML v	INCL v	#Ex
	ORH v	ORMH v	ORML v	ORL v	ANDNH v	ANDNMH v	ANDNML v	ANDNL v	
#Fx	JMP[B] v		PUSHJ[B] v		GETA[B] v		PUT[I] v		#Fx
	POP $3v$	RESUME $5v$	[UN]SAVE $20\mu+v$		SYNC v	SWYM v	GET v	TRIP $5v$	
	#8	#9	#A	#B	#C	#D	#E	#F	

如果发生转移则 $\pi = 2v$；如果没有发生转移则 $\pi = 0$.

目　录

第 7 章　组合查找

在发现它们之前，你会终日搜索，
而在你一旦拥有它们之后，就失去查找的价值了.
——巴萨尼奥，《威尼斯商人》（第一幕，第一场，第 117 句台词）

在如此密集的人类行动和反应中，
事件的所有可能组合都可能会发生，
而且许多细微末节的问题可能以惊人、离奇的方式呈现.
——夏洛克 · 福尔摩斯，《蓝宝石案》（1892）

组合算法的范围过于广阔，
以致于不能把它们写进一篇文章甚至单独一本书中.
——罗伯特 · 塔扬（1976）

当与各式各样的人争辩时，我头脑中总有
这样的印象，成功者是那些对于解决迷题
独具天赋的人. 生活中充满着迷题，当它们
不可避免地降临到我们头上时是在呼唤我们去解决.
——小萨姆 · 劳埃德（1926）

组合学研究的是可以把离散对象排列成种种不同模式的方法. 例如，讨论的对象可能是 $2n$ 个数 $\{1,1,2,2,\ldots,n,n\}$，而我们想把它们排成一行，使得在每个数 k 的两次出现之间恰好存在 k 个数. 当 $n=3$ 时，实际上仅有这样一种排成“兰福德配对”的方法，即 231213（及其左右反转）. 同样，当 $n=4$ 时也只有唯一解. 许多其他类型的组合模式将在下面讨论.

研究组合问题时，一般有五类基本问题，其中一些难度更大.

(i) 存在问题：是否存在遵循模式的排列 X?

(ii) 构造问题：倘若存在，能否快速找到这样的 X?

(iii) 计数问题：存在多少不同的排列 X?

(iv) 生成问题：能否依序访问所有的排列 $X_1, X_2, \ldots$?

(v) 优化问题：给出目标函数 f，何种排列使 $f(X)$ 达到最大值或最小值?

就兰福德配对而言，这里的每一个问题都很有趣.

例如，考虑存在问题. 通过反复试验很快发现，当 $n=5$ 时，我们无法把 $\{1,1,2,2,\ldots,5,5\}$ 恰当地置放在 10 个位置上. 两个 1 必须同时放在两个偶数号位置或者两个奇数号位置；同样，两个 3 和两个 5 必须选择两个偶数号或者两个奇数号的位置；但是，两个 2 和两个 4 使用两种编号的一个位置. 因此，我们不能恰好填满每对奇偶数间的 5 个位置. 这个推理同样证明，当 $n=6$，或者一般地，当 $\{1,2,\ldots,n\}$ 中有奇数个奇数值时，这个问题没有解.

换句话说，对于某个整数 m，兰福德配对只能在 $n=4m-1$ 或者 $n=4m$ 时存在. 反过来，当 n 是具有这种形式的值时，罗伊 · 戴维斯找到了构造符合要求的排列位置的简练方法（见习题 1）.

那么，究竟有多少本质上不同的兰福德配对 L_n? 当 n 增加时，存在很多这样的配对：

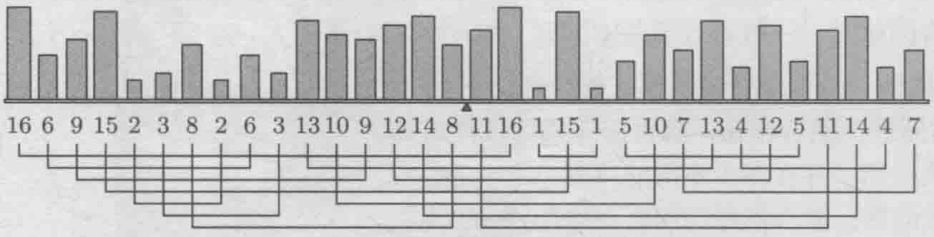
$$\begin{aligned} L_3&=1; & L_4&=1;\\ L_7&=26; & L_8&=150;\\ L_{11}&=17\,792; & L_{12}&=108\,144;\\ L_{15}&=39\,809\,640; & L_{16}&=326\,721\,800;\\ L_{19}&=256\,814\,891\,280; & L_{20}&=2\,636\,337\,861\,200;\\ L_{23}&=3\,799\,455\,942\,515\,488; & L_{24}&=46\,845\,158\,056\,515\,936. \end{aligned} \tag{1}$$

[L_{23} 和 L_{24} 的值是米卡埃尔·卡捷基、克里斯托夫·雅耶和阿兰·布伊在 2004 年和 2005 年确定的，见 *Studia Informatica Universalis* **4** (2005), 151–190.] 一种直觉上的计算显示，当 L_n 不是 0 时，它的数量级粗略为 $(4n/e^3)^{n+1/2}$（见习题 5）. 而且事实上，在所有已知情形下这个预测基本是正确的. 但是，显然不存在简单公式.

兰福德排列问题是一类称为*恰当覆盖问题*的组合难题的简单特例. 在 7.2.2.1 节，我们将要研究一种算法（称为“舞蹈链”），它是产生这类问题所有解的一种便捷方法. 例如，当 $n = 16$ 时，该方法对于它找到的每个兰福德配对排列大约仅需 3500 次内存访问. 因此，简单地生成全部配对并计数，便能在合理的时间内计算 L_{16} 的值.

但需注意，L_{24} 是一个*天文数字*——大约是 5×10^{16}，约为 1500 MIP 年.（回忆一下，一“MIP 年”是每秒执行 100 万条指令的计算机每年执行的指令数量，即 31 556 952 000 000.）所以，L_{24} 的确切值显然是由不涉及生成所有排列的某种方法确定的. 实际上有一种非常快的方法，利用多项式代数计算 L_n. 习题 6 中描述的启发性方法需要进行 $O(4^n n)$ 次运算，其效率看似不高. 但是，它以一个巨大的数量级因子 $\Theta((n/e^3)^{n-1/2})$ 胜过生成所有配对排列并且计数它们的方法，甚至当 $n = 16$ 时它的执行速度大约还要快 20 倍. 另一方面，我们或许永远无法知道 L_{100} 的确切值，即便计算机的速度变得越来越快.

我们还可以从各种不同途径考虑那些*最优的*兰福德配对排列. 例如，可以这样安排砝码 $\{1, 1, 2, 2, \ldots, 16, 16\}$ 的 16 个配对排列，它们满足兰福德条件，并且在将它们按相应次序置于一根平衡杆上时不会使杆倾斜，在这个意义下它们是“完全平衡”的：



(2)

换句话说，我们有 $15.5 \cdot 16 + 14.5 \cdot 6 + \cdots + 0.5 \cdot 8 = 0.5 \cdot 11 + \cdots + 14.5 \cdot 4 + 15.5 \cdot 7$. 同时，其中还有另一种平衡 $16 + 6 + \cdots + 8 = 11 + 16 + \cdots + 7$，因此还有 $16 \cdot 16 + 15 \cdot 6 + \cdots + 1 \cdot 8 = 1 \cdot 11 + \cdots + 15 \cdot 4 + 16 \cdot 7$.

此外，(2) 中的排列在全部 16 阶兰福德配对排列中具有最小宽度：图中底部的连线显示，按照从左到右的顺序，在任何点没有 7 个以上的未完成配对. 我们可以证明，宽度 6 是不够的.（见习题 7.）

在 $\{1, 1, \ldots, 16, 16\}$ 的排列 $a_1 a_2 \ldots a_{32}$ 中，什么排列在 $\sum_{k=1}^{32} k a_k$ 达到最大值的意义下是最小平衡的？可以证明，这个最大的可能值是 5258. 这种配对排列共有 12 016 个，其中一个是

$$2\,3\,4\,2\,1\,3\,1\,4\,16\,13\,15\,5\,14\,7\,9\,6\,11\,5\,12\,10\,8\,7\,6\,13\,9\,16\,15\,14\,11\,8\,10\,12. \tag{3}$$

一个更有趣的问题是求按照字典序的最小和最大兰福德配对排列. 对于 $n = 24$，如果我们用字母 a, b, ..., w, x 代替数 1, 2, ..., 23, 24，答案是

$$\begin{aligned}&\{\texttt{abacbdecfgdoersfpgqtuwxvjklonhmirpsjqkhltiunmwvx},\\ &\ \texttt{xvwsquntkigrdapaodgiknqsvxwutmrpohljcfbecbhmfejl}\}\end{aligned} \tag{4}$$

在这一章的后面几小节，我们将要讨论许多关于组合最优化的方法. 当然，我们的目标是在检查范围不超出全部可能排列空间的很小一部分的情况下求解这类问题.