TURING 图灵计算机科学丛书

Pearson

计算机程序设计艺术

卷3：排序与查找

（第2版）

[美] 高德纳（Donald E. Knuth）◎著

贾洪峰 ◎译

The Art of Computer Programming

Vol 3: Sorting and Searching

Second Edition

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵计算机科学丛书

计算机程序设计艺术
卷3：排序与查找
（第2版）

[美] 高德纳（Donald E. Knuth）◎著
贾洪峰 ◎译

The Art of Computer Programming

Vol 3: Sorting and Searching
Third Edition

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3, 排序与查找 : 第2版 / (美) 高德纳 (Donald E. Knuth) 著 ; 贾洪峰译. -- 北京 : 人民邮电出版社, 2017.1
（图灵计算机科学丛书）
书名原文: The Art of Computer Programming
ISBN 978-7-115-36065-6

Ⅰ. ①计… Ⅱ. ①高… ②贾… Ⅲ. ①程序设计
Ⅳ. ①TP311.1

中国版本图书馆CIP数据核字(2016)第266934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。本书为该系列的第 3 卷，全面讲述了排序和查找算法。书中扩展了卷 1 中数据结构的处理方法，并对各种算法的效率进行了大量的分析。

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　贾洪峰
责任编辑　傅志红
执行编辑　江志强 黄志斌
责任印制　彭志环
排版指导　刘海洋

◆ 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
固安县铭成印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：39.5　插页：1
字数：1081千字　2017年1月第1版
印数：1－4 000册　2017年1月河北第1次印刷
著作权合同登记号　图字：01-2009-7277号

定价：198.00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：京东工商广字第 8052 号

目　录

第 5 章 排序

没有什么比率先建立事物的新秩序
更难把握、更担风险、更成败莫测的了.
——尼可罗 · 马基亚维利,《君主论》(1513)

"但你不可能及时查对所有这些驾照号码," 德雷克反驳说,
"我们不需要这样做, 保罗. 我们只需要排成一个列表,
指出重复号码就行了."
——佩瑞 · 梅森,《愤怒的哀悼者》中的律师(1951)

"树排序" 计算机——应用这一新的 "计算机方法" 来进行自然研究,
可以快速辨识美国、阿拉斯加和加拿大 260 多个不同树种,
甚至包括棕榈、沙漠里的树, 以及其他外来树种.
要进行排序, 只要插入排序指针就行了.
——埃德蒙科技公司的产品目录(1964)

在本章, 我们将研究一个在程序设计中经常遇到的主题: 按升序或降序重新排列项目. 设想一下, 如果一本词典的单词不是按照字母顺序排列的, 那这本词典的使用将会是何等困难! 我们将会看到, 项目在计算机存储器中的存储顺序, 通常会深远地影响到处理这些项目的算法的速度和简单性.

尽管英语词典中把 sorting 一词定义为按照类别对事物进行分类、整理的过程, 但计算机程序员在使用这个词时通常会采用一种更特定的含义: 按照升序或降序对事物进行排列. 这一过程可能应当称为 ordering, 而不是 sorting; 但由于 ordering 一词附加了太多的不同含义, 所以一旦使用这个词, 马上就会导致混淆. 比如下面这个句子: "由于我们的磁带机中仅有两台处于工作状态(in working order), 所以我受命(was ordered)在短期内(in short order)订购(to order)更多磁带机, 以使(in order to)数据排序(to order the data)速度提高几个数量级(orders)." 而数学术语则为 order 一词赋予了更多的含义: the order of a group (群的阶)、the order of a permutation (排列的阶)、the order of a branch point (分支点的级数)、relations of order (次序关系), 等等, 等等. 所以我们认为: order 一词可能会导致混乱.

有人曾经提出, sequencing 是排序过程的最佳名字. 但这个词经常显得缺乏适当的内涵, 特别是存在相等元素时; 而且它有时会与其他术语相冲突. 的确, sorting 单词本身的使用也有些过度("I was sort of out of sorts after sorting that sort of data", 在对那类数据进行排序后, 我有点不高兴了), 但它在计算领域的交流中牢固地确定了自己的位置. 因此, 我们在使用 sorting 一词时, 将严格采用其 "排序" 含义, 后文不再解释了.

排序的一些最重要应用包括:

(a) *求解 "归属" 问题.* 也就是将所有具有相同标识的项目集中在一起. 假定有 10 000 个任意顺序的项目, 其中有许多项目的取值相同; 假定我们希望重新排列这些数据, 使所有取值相同的项目都出现在连续位置. 这一分类问题基本上是 sorting 的较早含义(即 "分类"). 只要采用这个单词的新含义(即 "排序")来处理文件, 使这些取值按升序排列 $v_1 \leq v_2 \leq \cdots \leq v_{10000}$, 就可以轻松解决这一问题. 这一过程所能实现的效率, 解释了为什么 sorting 一词的最初含义会发生变化(由 "分类" 变为 "排序").

(b) *匹配两个或更多个文件中的项目.* 如果几个文件按相同顺序排列, 一次顺序扫描就可能找出所有匹配项, 无须反复查找. 佩瑞 · 梅森就是利用这一原理来帮助侦破谋杀案的(见本章

开头的引语）. 在处理一个信息列表时，最快速的处理方法是从头至尾依次遍历该列表，而不是在其中随机跳转，除非整个列表小到可以放到一个高速随机存储器中. 通过排序，有可能对大型文件进行顺序访问，实际取代直接寻址.

(c) *按键值查找信息*. 在第 6 章将会看到，排序还可以帮助查找，因此有助于使计算机输出结果更便于人类使用. 事实上，一份按字母顺序排序的列表，即使与其相关联的数值信息存在计算错误，往往也会显得更权威一些.

尽管排序在传统上主要用于商务数据处理，但它实际上是一种基本工具，每个程序员都应当牢记在心，在各种情况下广泛运用. 我们已经在习题 2.3.2–17 中讨论了它在代数式化简中的应用. 后面的习题将展示各种典型应用.

可以展示排序通用性的一个早期大规模软件系统是“LARC 科学编译器”，它是由乔尔 · 埃德温 、戴维 · 弗格森和他们在“计算机科学公司”的同事们于 1960 年开发的. 这一优化编译器是针对扩展 FORTRAN 语言开发的，其中大量使用了排序，以适当顺序向各种编译算法提供源程序的相关部分. 第一遍扫描是词法扫描，将 FORTRAN 源代码划分为各个标记，每个标记表示一个标识符、一个常量或一个运算符，等等. 每个标记都被赋予几个序列号；在按照名称和适当的序列号进行排序时，会将所有使用某一给定标识符的内容都集中在一起. 用户用“定义项”来指定一个标识符是代表函数名、参数，还是数组变量，由于为定义项指定的序列号较小，所以它们会在拥有给定标识符的标记中排在前列，这样既便于检查其使用是否存在冲突，又可根据 EQUIVALENCE 声明来分配存储. 针对每个标识符收集的信息现在被附加到每个标记，这样就不再需要在高速存储器中维护标识符的“符号表”. 随后，根据另一序列号对经过更新的标记进行排序，这一操作基本上会使源程序恢复最初顺序，但精心设计的编号机制将算术表达式转换为一种更为方便的“波兰前缀”形式. 在后续编译阶段也会用到排序，以便于循环优化、将错误消息合并到清单中等. 简而言之，采用这一编译器的设计方式，在针对辅助磁鼓存储器上的存储文件执行操作时，几乎都可以顺序完成，这是因为在这些数据上附加了适当的序列号，从而可以采用各种便利的排列方式来存储数据.

20 世纪 60 年代的计算机制造商估计：将所有用户考虑在内时，他们的计算机有超过 25% 的运行时间花费在排序上. 事实上，在许多安装实例中，排序任务占用了一半以上的计算时间. 根据这些统计数字，也许能够得出以下结论之一：(1) 排序有许多重要应用，(2) 许多人在不应当排序的时候进行了排序，(3) 低效排序算法的使用非常普遍. 实际情景可能涉及所有这三种可能性，但无论是哪种情况，都可以看出：排序值得作为实际问题加以认真研究.

即使排序几乎没有什么用处，也有充足的理由对其进行研究. 已经发现的一些天才算法表明：排序本身就是一个极有意义、值得探讨的主题. 在这一领域还有许多富有魅力、悬而未决的问题，当然也有大量已经解决的问题.

从更宽广的角度来看，我们还会发现，排序算法提供了一个很有价值的*案例研究*：如何从整体上攻克计算机编程问题. 在这一章将会介绍许多重要的数据结构操作原理. 我们将研究各种排序技巧的演进过程，尝试描述这些思想最初是如何发现的. 通过这一案例研究的外推，可以学习到大量策略，用于帮助我们为其他计算机问题设计出优秀算法.

排序技巧还出色地展示了*算法分析*中涉及的一般思想——这些思想用于判断各种算法的性能特征，以在相互竞争的方法之间做出明智选择. 在这一章，偏爱数学的读者将会找到许多具有指导意义的技巧，用于评估计算机算法的速度、化解复杂的递推关系. 另一方面，这些材料的组织方式也会使不喜欢数学的读者能够放心地略过这些计算.

在继续讨论之前，应当更清楚地描述一下我们的问题，并介绍一些术语. 有 N 个需要排序的项目

$$R_1, R_2, \ldots, R_N,$$

它们被称为记录，由 N 条记录组成的整个集合称为文件. 每条记录 R_j 有一个键 K_j，它左右着排序过程. 除了键之外，通常还存在其他数据，这一额外的“附属信息”对于排序过程没有什么影响，只是必须将其作为每条记录的组成部分加以携带.

针对这些键指定了一种顺序关系“$<$”，使得对于任意键值 a, b, c 满足以下条件：

(i) $a < b$，$a = b$，$b < a$ 中有且仅有一种关系为真.（这一条件称为三分律.）

(ii) 如果 $a < b$，$b < c$，则 $a < c$.（这一条件是广为人知的传递律.）

性质 (i) 与性质 (ii) 刻画了线性次序的数学概念，也称为全序. 任何满足这两条性质的关系“$<$”都可以使用本章即将介绍的大多数方法进行排序，不过其中一些排序方法专门设计用来处理具有通常次序的数值键或字母键.

排序的目的是确定下标 $\{1, 2, \ldots, N\}$ 的一种排列 $p(1)\,p(2)\ldots p(N)$，使这些键值按非递减顺序排列：

$$K_{p(1)} \le K_{p(2)} \le \cdots \le K_{p(N)}. \tag{1}$$

如果再进一步要求：具有相同键值的记录仍然保持原来的相对顺序，则说这一排序是稳定的. 换句话说，稳定排序具有以下补充特性：

$$\text{如果} \quad K_{p(i)} = K_{p(j)} \quad \text{且} \quad i < j, \quad \text{则} \quad p(i) < p(j). \tag{2}$$

在某些情况下，我们希望重新安排这些记录在存储器中的物理位置，使它们的键值处于有序状态. 但在另外一些情况下，只需一个辅助表就足够了，这个辅助表以某种方式指定排列方式，从而可以按照键序来访问这些记录.

本章的一些排序方法假定同时存在 ∞ 和 $-\infty$ 值，或存在其中之一，分别将其定义为大于或小于所有键：

$$-\infty < K_j < \infty, \qquad \text{对于 } 1 \le j \le N. \tag{3}$$

这些极端值偶尔用作虚拟键或标识指示器. 性质 (3) 中排除了相等的情况. 如果出现相等情况，可以对这些算法进行修改，使它们仍能正常工作，但通常会损失一些效率和简洁性.

排序通常可以划分为内部排序和外部排序. 在内部排序中，记录完全保存在计算机的高速随机存储器中，而当记录数目超出了存储器能够同时容纳的数量时，则采用外部排序. 内部排序在数据的结构安排与存取访问上具有更大的灵活性，而外部排序则向我们展示如何应对相当严格的存取限制.

利用一个相当出色的通用排序算法，对 N 条记录进行排序所需要的时间大约与 $N \log N$ 成正比，大约要对数据进行 $\log N$ 次“扫描”. 在 5.3.1 节将会看到，如果这些记录的顺序随机，并且排序过程是通过键值的两两比较来完成的，这就是最短排序时间. 如果将记录数量加倍，其他条件不变，排序时间将略高于原来的两倍.（实际上，当 N 趋近于无穷大时，如果键值不同，更准确的排序时间应当是 $N(\log N)^2$，这是因为键大小的增长速度至少和 $\log N$ 一样快. 但从实际应用角度来说，N 永远不可能真正趋近于无穷大.）

另一方面，对于某一连续数值分布来说，如果已知这些键是随机分布的，将会看到，排序过程一般可以在 $O(N)$ 个步骤中完成.

习题（第一组）

1. [*M20*] 由三分律和传递律证明：如果假定排序稳定，则排列 $p(1)\,p(2)\ldots p(N)$ 是唯一确定的.

2. [*21*] 假定一个特定文件中的每条记录 R_j 包含两个键，一个“主键” K_j 和一个“次键” k_j，在每一组键上定义了线性次序 $<$. 然后可以按照通常方式在键对 (K,k) 之间定义字典序：

$$\text{当}\quad K_i<K_j\quad\text{或}\quad\text{当}\quad K_i=K_j\quad\text{且}\quad k_i<k_j\quad\text{时，}\quad (K_i,k_i)<(K_j,k_j).$$

Alice 取得这样一个文件，首先按照主键对其排序，得到 n 组记录，每组记录中的主键相等，

$$K_{p(1)}=\cdots=K_{p(i_1)}<K_{p(i_1+1)}=\cdots=K_{p(i_2)}<\cdots<K_{p(i_{n-1}+1)}=\cdots=K_{p(i_n)},$$

式中 $i_n=N$. 随后，她对 n 组中的每一组记录 $R_{p(i_{j-1}+1)},\ldots,R_{p(i_j)}$ 按照次键进行排序.

Bill 取得同一原始文件，首先按照次键对其排序，然后按照主键对前面得到的文件进行排序.

Chris 取得同一原始文件，根据主键和次键 (K_j,k_j) 的字典序对其进行一次排序操作.

每个人得到的结果是否相同?

3. [*M25*] 设 $<$ 是 $K_1,\ldots,K_N$ 的一个关系，满足三分律，但不满足传递律. 证明：即使不满足传递律，也可能以稳定形式对这些记录进行排序，使其满足条件 (1) 和条件 (2). 事实上，至少有三种排列满足这些条件!

▶ **4.** [*21*] 词典编纂人员在编纂词典时，必须把大写和小写字母整合在一起，所以并没有在词典中实际使用严格的字典序. 他们希望按照如下方式进行排序：

$$\mathrm{a}<\mathrm{A}<\mathrm{aa}<\mathrm{AA}<\mathrm{AAA}<\mathrm{Aachen}<\mathrm{aah}<\cdots<\mathrm{zzz}<\mathrm{ZZZ}.$$

说明如何实现上述字典序.

▶ **5.** [*M28*] 为所有非负整数设计一种满足以下条件的二进制编码：如果把 n 编码为字符串 $\rho(n)$，当且仅当 $\rho(m)$ 根据字典序小于 $\rho(n)$ 时，$m<n$. 此外，对于任意 $m\neq n$，$\rho(m)$ 都不应当是 $\rho(n)$ 的前缀. 如果可能，对于所有大的 n 值，$\rho(n)$ 的长度应当不超过 $\lg n+O(\log\log n)$.（如果希望对混有单词与数字的文本进行排序，或者希望将任何大型字母表映射为二进制字符串，这种编码都是很有用的.）

6. [*15*] 阿呆先生是一位 MIX 程序员，他希望知道存储在单元 A 的数值是大于、小于还是等于存储在单元 B 的数值. 于是，他编写了代码 ‘LDA A; SUB B’，并测试寄存器 A 是正，是负，还是零. 他犯了什么严重错误? 应当怎么做?

7. [*17*] 根据以下规范编写一段 MIX 子例程，进行键值的多精度对比.

调用序列： JMP COMPARE

进入条件： rI1 $=n$；对于 $1\le k\le n$，CONTENTS(A $+k$) $=a_k$ 且 CONTENTS(B $+k$) $=b_k$；
假定 $n\ge 1$.

退出条件： CI = GREATER，如果 $(a_n,\ldots,a_1)>(b_n,\ldots,b_1)$；
CI = EQUAL，如果 $(a_n,\ldots,a_1)=(b_n,\ldots,b_1)$；
CI = LESS，如果 $(a_n,\ldots,a_1)<(b_n,\ldots,b_1)$；
rX 和 rI1 可能受到影响.

其中，关系 $(a_n,\ldots,a_1)<(b_n,\ldots,b_1)$ 表示由左向右的字典序，也就是说，存在一个下标 j，对于 $n\ge k>j$，$a_k=b_k$，而 $a_j<b_j$.

▶ **8.** [*30*] 单元 A 和单元 B 分别包含两个数值 a 与 b. 证明：有可能编写一个 MIX 程序，计算 $\min(a,b)$ 并将其存储在单元 C，但不使用任何跳转操作符.（小心：由于不能判断是否发生了运算溢出，所以明智的做法是确保无论 a、b 取什么值都不会发生溢出.）

9. [*M27*] 以非递减顺序存储 N 个介于 0 和 1 之间的独立均匀分布随机变量，这些数值中第 r 个最小数 $\le x$ 的概率是多少?

习题（第二组）

早年间，计算机没有太多的随机存储器. 以下每个习题都描述了计算机程序员在当时必须解决的一个问题. 假定可供使用的内部存储器容量只有几千个字，用大约六条磁带（这些磁带对于排序来说足够了）作为补充，请给出一种解决这一问题的“好”方法. 事实证明，在这些约束条件下能够出色工作的算法，在现代计算机上也能高效运行.

10. [*15*] 有一个包含 100 万个数据字的磁带. 如何判断这条磁带上有多少个互不相同的数据字?

11. [*18*] 你在美国国家税务局工作. 你会从各家单位收到数百万个“信息”表，报告这些单位向别人支付了多少报酬；还会从人们那里收到数百万个“税”表，报告这些人收到了多少报酬. 如何找出那些没有申报全部收入的人?

12. [*M25*] （矩阵转置）有一条磁带，其中包含 100 万个字，表示按行存储的 1000×1000 矩阵的元素：$a_{1,1}\, a_{1,2} \ldots a_{1,1000}\, a_{2,1} \ldots a_{2,1000} \ldots a_{1000,1000}$. 如何创建一条磁带，在其中按列存储这些元素：$a_{1,1}\, a_{2,1} \ldots a_{1000,1}\, a_{1,2} \ldots a_{1000,2} \ldots a_{1000,1000}$?（争取使数据扫描次数不超过 12 次.）

13. [*M26*] 如何将一个包括 N 个字的大型文件“打乱”为一种随机排列方式?

14. [*20*] 你正在使用两个计算机系统，它们对于用来确定数字字母顺序的“排序序列”采用不同约定. 如何将一个计算机的有序数字字母文件调整为另一计算机使用的顺序?

15. [*18*] 有一个姓名清单，其中列出了大量在美国出生的人的名字，还有他们出生的州名. 如何计算在每个州出生的人数?（假定清单中每个人只出现一次.）

16. [*20*] 为便于修改大型 FORTRAN 程序，希望设计一个“交叉引用”例程；这一例程以 FORTRAN 程序为输入，在输出时添加一个索引，给出程序中对每个标识符（即每个名字）的每次应用. 如何设计这样一个例程?

▸ **17.** [*33*] （图书馆卡片排序）在计算机数据库出现之前，每个图书馆都维护一个卡片目录，使用户能够找到自己想要的书籍. 但是随着馆藏的增加，要将目录卡片排成便于人类使用的顺序，是一件非常复杂的任务. 下面的“字母顺序”清单展示了在《美国图书馆协会排片规则》（芝加哥：1942）推荐的许多步骤：

卡片文字	注释
R. Accademia nazionale dei Lincei, Rome	Ignore foreign royalty (except British)
1812; ein historischer Roman.	Achtzehnhundertzwölf
Bibliothèque d'histoire révolutionnaire.	Treat apostrophe as space in French
Bibliothèque des curiosités.	Ignore accents on letters
Brown, Mrs. J. Crosby	Ignore designation of rank
Brown, John	Names with dates follow those without
Brown, John, mathematician	... and the latter are subarranged
Brown, John, of Boston	by descriptive words
Brown, John, 1715–1766	Arrange identical names by birthdate
BROWN, JOHN, 1715–1766	Works “about” follow works “by”
Brown, John, d. 1811	Sometimes birthdate must be estimated
Brown, Dr. John, 1810–1882	Ignore designation of rank
Brown-Williams, Reginald Makepeace	Treat hyphen as space
Brown America.	Book titles follow compound names
Brown & Dallison's Nevada directory.	& in English becomes “and”
Brownjohn, Alan	
Den', Vladimir Éduardovich, 1867–	Ignore apostrophe in names
The den.	Ignore an initial article
Den lieben langen Tag.	... provided it's in nominative case
Dix, Morgan, 1827–1908	Names precede words

1812 ouverture.	Dix-huit cent douze
Le XIXe siècle français.	Dix-neuvième
The 1847 issue of U. S. stamps.	Eighteen forty-seven
1812 overture.	Eighteen twelve
I am a mathematician.	(a book by Norbert Wiener)
IBM journal of research and development.	Initials are like one-letter words
ha-I ha-ehad.	Ignore initial article
Ia; a love story.	Ignore punctuation in titles
International Business Machines Corporation	
al-Khuwārizmī, Muḥammad ibn Mūsā, *fl.* 813–846	Ignore initial "al-" in Arabic names
Labour. A magazine for all workers.	Respell it "Labor"
Labor research association	
Labour, *see* Labor	Cross-reference card
McCall's cookbook	Ignore apostrophe in English
McCarthy, John, 1927–	Mc = Mac
Machine-independent computer programming.	Treat hyphen as space
MacMahon, Maj. Percy Alexander, 1854–1929	Ignore designation of rank
Mrs. Dalloway.	"Mrs." = "Mistress"
Mistress of mistresses.	
Royal society of London	Don't ignore British royalty
St. Petersburger Zeitung.	"St." = "Saint", even in German
Saint-Saëns, Camille, 1835–1921	Treat hyphen as space
Ste-Marie, Gaston P	Sainte
Seminumerical algorithms.	(a book by Donald Ervin Knuth)
Uncle Tom's cabin.	(a book by Harriet Beecher Stowe)
U. S. bureau of the census.	"U. S." = "United States"
Vandermonde, Alexandre Théophile, 1735–1796	
Van Valkenburg, Mac Elwyn, 1921–	Ignore space after prefix in surnames
Von Neumann, John, 1903–1957	
The whole art of legerdemain.	Ignore initial article
Who's afraid of Virginia Woolf?	Ignore apostrophe in English
Wijngaarden, Adriaan van, 1916–	Surname begins with uppercase letter

（这些规则大多有一些特定的例外，还有其他许多规则，这里没有列出.）

如果你接到一项任务：用计算机对大量目录卡进行排序，最终维护由这些卡片组成的一个超大型文件，如果你没有机会改变卡片编排的长期策略，如何安排这些数据，以方便进行排序和合并操作?

18. [*M25*] （欧内斯特·帕克）莱昂哈德·欧拉曾经猜测 [*Nova Acta Acad. Sci. Petropolitanæ* **13** (1795), 45–63, §3; 写于 1778 年]，对于正整数 u, v, w, x, y, z，以下方程无解：

$$u^6 + v^6 + w^6 + x^6 + y^6 = z^6.$$

同时，他还猜测，对于所有 $n \geq 3$，以下方程没有正整数解：

$$x_1^n + \cdots + x_{n-1}^n = x_n^n,$$

但计算机发现了恒等式 $27^5+84^5+110^5+133^5=144^5$，证明这个更具一般性的猜想是错误的，见利昂·兰德、托马斯·帕金和约翰·塞尔弗里奇，*Math. Comp.* **21** (1967), 446–459. 诺姆·埃尔基斯后来找到了当 $n=4$ 时的无数个反例 [*Math. Comp.* **51** (1988), 825–835]. 你能否想出一种方法，通过排序来帮助查找欧拉猜想在 $n=6$ 时的反例?

▶ **19.** [*24*] 给定一个文件，其中包含大约 100 万个不同的 30 位二进制字 $x_1,\ldots,x_N$，有什么好办法可以找出全部*互补*对 $\{x_i,x_j\}$? （如果一个字中为 0 的位置，另一个字中为 1，反之亦然，则说这两个字是互补的. 因此，如果将两个字看作二进制数，当且仅当它们的和为 $(11\ldots1)_2$ 时，这两个字互补.）

▶ **20.** [*25*] 给定一个文件，其中包含 1000 个 30 位二进制字 $x_1,\ldots,x_{1000}$，怎样准备一个字对列表，使其中的所有字对 (x_i,x_j) 都满足 $x_i=x_j$，最多有两个数位例外?

21. [*22*] 如何寻找由五个字母组成的变位词 (anagram)，比如 CARET, CARTE, CATER, CRATE, REACT, RECTA, TRACE; CRUEL, LUCRE, ULCER; DOWRY, ROWDY, WORDY? [人们可能想知道，除了下面这一组五字母英语变位词之外，是否还有其他包括 10 个或更多个五字母英语变位词的集合：

APERS, ASPER, PARES, PARSE, PEARS, PRASE, PRESA, RAPES, REAPS, SPAER, SPARE, SPEAR,

也许还可以向这一集合中添加法语单词APRÈS.]

22. [*M28*] 给定大量有向图的技术说明，可以用什么方法将*同构*图划分在一起?（如果有向图的顶点之间存在一一对应关系、有向弧之间存在一一对应关系，其中顶点与有向弧之间的对应关系保持关联性，则说这些有向图是同构的.）

23. [*30*] 在一个包括 4096 人的人群中，每个人有大约 100 个熟人. 已经准备了一个文件，其中列出了所有互相熟识的人员对.（这一关系是对称的：如果 x 熟悉 y，那 y 也熟悉 x. 因此，这个文件包括大约 200 000 个项目.）给定一个 k 值，如何设计一种算法，列出这一人群中所有包含 k 个人的*小团体*?（小团体是指一组互相熟悉的人：小团体中的每个人都与所有其他人熟悉.）假定不存在人数达到 25 的小团体，因此小团体的总数不会过多.

▶ **24.** [*30*] 姓名不同的 300 万人，头脚相挨着躺下来，可以从纽约排到加利福尼亚. 每个参与者都拿到一个纸片，在上面写下自己的名字和紧排在他西侧的人名. 在最西端的那个人不知道要做什么，所以他把自己的纸片扔了. 然后把剩下的 2 999 999 个纸片放在一个巨大的篮子里，运到华盛顿特区的国家档案馆. 在那里，将篮中的纸片完全打乱，传送到磁带上.

这时，一位信息科学家观察到：磁带上有足够的信息可以按原始顺序重建出人员列表. 一位计算机科学家发现了一种方法，只需对数据磁带进行不到 1000 次扫描就能完成这一重建过程，而且仅对磁带文件进行顺序访问，并使用少量随机存储器. 怎样才可能做到呢?

[换句话说，对于 $1\le i<N$，以随机顺序给出数对 (x_i,x_{i+1})，其中 x_i 互不相同. 如何获得序列 $x_1x_2\ldots x_N$? 要求所采取的全部操作仅限于能够在磁带上使用的串行技术. 这个问题就是：在没有一种简单方法来判断两个给定键的前后顺序时，如何完成排序. 我们已经在习题 2.2.3–25 中提出了这一问题.]

25. [*M21*] （*离散对数*）已知 p 是一个（相当大的）素数，a 是一个模 p 的*原根*. 因此，对于 $1\le b<p$ 范围内的所有 b 值，存在唯一的 n 值，使得 $a^n \bmod p=b$, $1\le n<p$.（这个 n 称为 b 关于 a 对 p 求模的*指数*.）说明在给定 b 时，如何在不超过 $\Omega(n)$ 个步骤中找到 n. [*提示*：设 $m=\lceil\sqrt{p}\rceil$，试针对 $0\le n_1,n_2<m$ 求解 $a^{mn_1}\equiv ba^{-n_2}$ (modulo p).]

*5.1 排列的组合性质

有限集的一个“排列”是将其元素列成一行的一种排法. 排列对于研究排序算法特别重要，这是因为它们代表了未排序输入数据. 一些排列会使排序过程的某一特定步骤被执行特定的次数，为了研究不同排序方法的效率，我们希望能够计算此类排列的数目.

当然，我们在前面几章中已经频繁遇到排列. 例如，在 1.2.5 节讨论了两种基本的理论方法，用于构建 n 个对象的 $n!$ 个排列；在 1.3.3 节分析了一些涉及排列的循环结构及乘法性质的算法. 3.3.2 节研究了它们的“上运行”和“下运行”. 本节的目的是研究排列的其他几个性质，并考虑允许出现相等元素的一般情景. 在这一研究过程中，我们将会学习大量有关组合数学的知识.

排列的性质本身就足以值得关注，集中地系统介绍这些内容，要比把这些材料分散在本章各个位置更方便一些. 不过，对于不太喜欢数学的读者，或者急于钻研排序方法的读者，建议直接跳到 5.2 节，因为这一节实际上与排序没有多少直接联系.

*5.1.1 反序

设 $a_1\,a_2\ldots a_n$ 是集合 $\{1,2,\ldots,n\}$ 的一个排列. 如果 $i<j$ 且 $a_i>a_j$，则元素对 (a_i,a_j) 称为该排列的一个*反序*. 例如，排列 3 1 4 2 有三个反序：$(3,1)$、$(3,2)$ 和 $(4,2)$. 每个反序都是一对乱序元素，所以只有那个没有反序的排列才是有序排列 $1\,2\ldots n$. 我们之所以对反序感兴趣，主要原因就是它与排序方法的这一联系，尽管我们已经使用过这一概念来分析动态存储分配算法（见习题 2.2.2–9）.

反序的概念是由加布里埃尔·克拉默在 1750 年结合其著名的线性方程组求解规则提出的 [*Intr. à l'Analyse des Lignes Courbes Algébriques* (Geneva: 1750), 657–659, 见托马斯·缪尔, *Theory of Determinants* **1** (1906), 11–14]. 大体上，克拉默采用以下方式定义一个 $n\times n$ 矩阵的行列式：

$$\det\begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1n}\\ \vdots & \vdots & & \vdots\\ x_{n1} & x_{n2} & \ldots & x_{nn}\end{pmatrix}=\sum(-1)^{\operatorname{inv}(a_1a_2\ldots a_n)}x_{1a_1}x_{2a_2}\ldots x_{na_n},$$

也就是对 $\{1,2,\ldots,n\}$ 的所有排列 $a_1\,a_2\ldots a_n$ 求和，其中 $\operatorname{inv}(a_1\,a_2\ldots a_n)$ 是该排列的反序个数.

设 b_j 表示在 j 的左侧有多少个元素大于 j，从而可以得到排列 $a_1\,a_2\ldots a_n$ 的*反序表* $b_1\,b_2\ldots b_n$. 换句话说，b_j 是指有多少个反序的第二个元素为 j. 例如，排列

$$5\;9\;1\;8\;2\;6\;4\;7\;3, \tag{1}$$

其反序表为

$$2\;3\;6\;4\;0\;2\;2\;1\;0, \tag{2}$$

这是因为，5 和 9 位于 1 的左侧，5 9 8 位于 2 的左侧，等等. 这一排列共有 20 个反序. 根据定义，数值 b_j 总是满足

$$0\le b_1\le n-1,\quad 0\le b_2\le n-2,\quad \ldots,\quad 0\le b_{n-1}\le 1,\quad b_n=0. \tag{3}$$

通过简单的观察就可能看出有关反序的一个最重要事实：*反序表唯一地确定了对应的排列*. 只需要连续确定元素 $n,n-1,\ldots,1$ 的相对位置（按照所列顺序），就可以由任意满足 (3) 式的反序表 $b_1\,b_2\ldots b_n$ 得到生成这一反序表的唯一排列. 例如，我们可以通过以下方法构造一个与 (2) 式相对应的排列：写下数 9；由于 $b_8=1$，所以将 8 放在 9 的后面；与此类似，由于

$b_7 = 2$，所以将 7 放在 8 和 9 的后面；接下来，由于 $b_6 = 2$，所以将 6 排在已有的头两个数字之后．到目前所得到的部分结果为

$$9\ 8\ 6\ 7.$$

继续进行，由于 $b_5 = 0$，所以将 5 放在最左侧；将 4 放在四个数之后；将 3 放在六个数之后（也就是最右侧），得

$$5\ 9\ 8\ 6\ 4\ 7\ 3.$$

以类似方式插入 2 和 1，将得到 (1).

因为我们经常可以将一个用排列表述的问题转换为用反序表表述的等价问题，而后者更容易解决，所以上述对应特性非常重要．例如，考虑最简单的问题：$\{1, 2, \ldots, n\}$ 可能有多少种排列？其答案必然是所有反序表的总数，而这一数目是很好计算的：b_1 有 n 种选择，b_2 有 $n-1$ 种独立选择，……，b_n 有 1 种选择，所以总共有 $n(n-1)\ldots 1 = n!$ 种选择．因为诸 b 完全相互独立，所以反序的数目很容易计算．另一方面，诸 a 必须彼此不同．

1.2.10 节分析了从右向左读取一个排列时出现的局部最大值数目．换句话说，我们计算了有多少元素大于其所有后续元素．（例如，在 (1) 中，从右向左的最大值是 3, 7, 8, 9.）这一数目就是有多少个 j 使 b_j 取其最大值 $n-j$．因为 b_1 以 $1/n$ 的概率等于 $n-1$，而 b_2（独立地）以 $1/(n-1)$ 的概率等于 $n-2$，等等．因此，通过考虑反序可以很清楚地知道从右向左的最大值的平均个数是

$$\frac{1}{n} + \frac{1}{n-1} + \cdots + \frac{1}{1} = H_n.$$

以类似方法可以很轻松地推导出相应的生成函数．

如果交换一个排列的两个相邻元素，容易看到反序总数将会增大或减少单位 1．图 1 给出 $\{1, 2, 3, 4\}$ 的 24 个排列，用直线将交换一次相邻元素所得到的排列连接在一起．沿任何一条直线向下移动，都恰好生成一个新的反序．因此，一个排列 π 的反序数目是图 1 中从 1234 到 π 这一向下路径的长度．所有此类路径的长度必然相同．

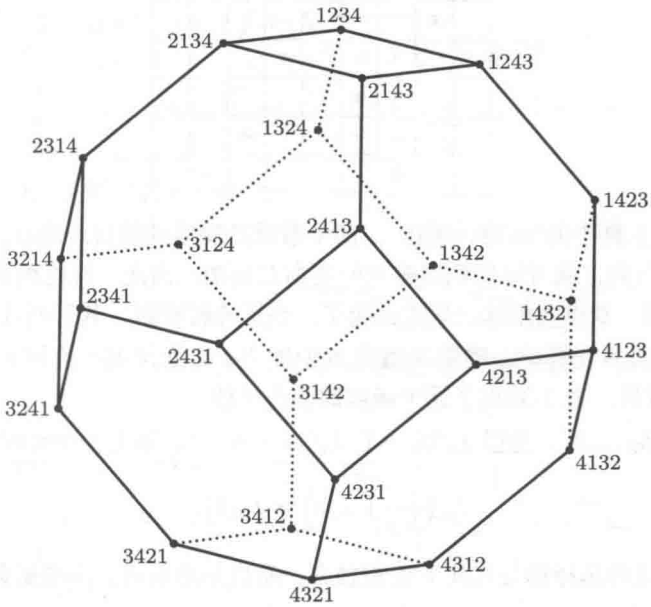


图 1 截八面体，说明在交换一个排列的相邻元素时的反序变化

顺便指出，图 1 中的图形可以看作一个三维立体“截八面体”，它有 8 个正六边形表面和 6 个正方形表面. 这是阿基米德提出的经典均匀多面体之一（见习题 10）.

读者不要把排列的反序与排列的*逆*相混淆. 回想一下，我们可以将一个排列写成两行：

$$\begin{pmatrix} 1 & 2 & 3 & \dots & n \\ a_1 & a_2 & a_3 & \dots & a_n \end{pmatrix}; \tag{4}$$

这个排列的逆 $a'_1\,a'_2\,a'_3\dots a'_n$ 是通过以下变换所得到的排列：交换这两行的位置，然后对各列进行排序，使新的上面一行满足递增顺序：

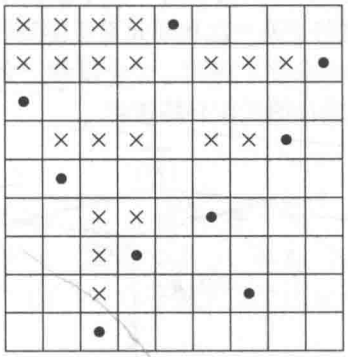
$$\begin{pmatrix} a_1 & a_2 & a_3 & \dots & a_n \\ 1 & 2 & 3 & \dots & n \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & \dots & n \\ a'_1 & a'_2 & a'_3 & \dots & a'_n \end{pmatrix}. \tag{5}$$

例如，591826473 的逆为359716842，这是因为

$$\begin{pmatrix} 5&9&1&8&2&6&4&7&3 \\ 1&2&3&4&5&6&7&8&9 \end{pmatrix} = \begin{pmatrix} 1&2&3&4&5&6&7&8&9 \\ 3&5&9&7&1&6&8&4&2 \end{pmatrix}.$$

逆的另一定义方式为：当且仅当 $a_k = j$ 时，我们说 $a'_j = k$.

排列的逆是由海因里希 · 罗思最早定义的［在 *Sammlung combinatorisch-analytischer Abhandlungen* 中定义，由卡尔 · 兴登堡编，**2** (Leipzig: 1800), 263–305］，他注意到逆与反序之间的重要联系：*排列的逆与排列本身恰好有同样多的反序数目*. 罗思对这一事实的证明可能不是最简单的一个，但它具有指导意义，而且非常漂亮. 构造一个 $n \times n$ 的棋盘，只要 $a_i = j$，就在第 i 行第 j 列放一个黑点. 然后，如果一个方格的下方（同一列内）和右侧（同一行内）都有黑点，则在该方格放入一个 $\times$. 例如，591826473 的图形为：



容易看到，b_j 就是 j 列中的 $\times$ 数，所以 $\times$ 的个数就是反序的数目. 现在，如果对这个图形进行转置——交换行与列，就可以得到原排列的逆的对应图. 因此，在这两种情况下，$\times$ 的个数（反序的数目）相同. 罗思利用这一事实证明了：*当矩阵转置时，其行列式不变*.

在几种排序算法的分析中，需要知道有多少种“n 元素排列”恰好有 k 个反序. 我们用 $I_n(k)$ 来表示这一数目. 表 1 列出了这个函数的前几个值.

考虑反序表 $b_1\,b_2\dots b_n$，易得 $I_n(0) = 1$, $I_n(1) = n-1$，还有一个对称性质：

$$I_n\left(\binom{n}{2} - k\right) = I_n(k). \tag{6}$$

此外，由于每个 b 值的选择都与其他 b 值相独立，所以不难看出，生成函数

$$G_n(z) = I_n(0) + I_n(1)z + I_n(2)z^2 + \cdots \tag{7}$$

表 1 具有 k 个反序的排列

n	$I_n(0)$	$I_n(1)$	$I_n(2)$	$I_n(3)$	$I_n(4)$	$I_n(5)$	$I_n(6)$	$I_n(7)$	$I_n(8)$	$I_n(9)$	$I_n(10)$	$I_n(11)$
1	1	0	0	0	0	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0	0	0
3	1	2	2	1	0	0	0	0	0	0	0	0
4	1	3	5	6	5	3	1	0	0	0	0	0
5	1	4	9	15	20	22	20	15	9	4	1	0
6	1	5	14	29	49	71	90	101	101	90	71	49

满足 $G_n(z) = (1+z+\cdots+z^{n-1})G_{n-1}(z)$. 因此它拥有邦雅曼·罗德里格斯注意到的较简单形式 [*J. de Math.* **4** (1839), 236–240]:

$$(1+z+\cdots+z^{n-1})\ldots(1+z)(1) = (1-z^n)\ldots(1-z^2)(1-z)/(1-z)^n. \tag{8}$$

根据这一生成函数，可以很轻松地扩展表 1，还可以验证：表中锯齿线下方的数满足

$$I_n(k) = I_n(k-1) + I_{n-1}(k), \quad k < n. \tag{9}$$

（这一关系对于锯齿线上方的数不成立.）一种更复杂的论证过程表明（见习题 14）：事实上，有以下公式

$$\begin{aligned}
I_n(2) &= \binom{n}{2} - 1, && n \ge 2;\\
I_n(3) &= \binom{n+1}{3} - \binom{n}{1}, && n \ge 3;\\
I_n(4) &= \binom{n+2}{4} - \binom{n+1}{2}, && n \ge 4;\\
I_n(5) &= \binom{n+3}{5} - \binom{n+2}{3} + 1, && n \ge 5;
\end{aligned}$$

一般来说，$I_n(k)$ 的公式包括大约 $1.6\sqrt{k}$ 项：

$$\begin{aligned}
I_n(k) = {} & \binom{n+k-2}{k} - \binom{n+k-3}{k-2} + \binom{n+k-6}{k-5} + \binom{n+k-8}{k-7} - \cdots \\
& + (-1)^j \left(\binom{n+k-u_j-1}{k-u_j} + \binom{n+k-u_j-j-1}{k-u_j-j} \right) + \cdots, \quad n \ge k,
\end{aligned} \tag{10}$$

其中 $u_j = (3j^2-j)/2$，是所谓的“五角数”.

如果将 $G_n(z)$ 除以 $n!$，可以得到给出 n 元素随机排列的反序个数的概率分布的生成函数 $g_n(z)$. 它是乘积

$$g_n(z) = h_1(z)h_2(z)\ldots h_n(z), \tag{11}$$

式中 $h_k(z) = (1+z+\cdots+z^{k-1})/k$ 是一个小于 k 的随机非负整数的一致分布的生成函数. 可以得到

$$\begin{aligned}
\operatorname{mean}(g_n) &= \operatorname{mean}(h_1) + \operatorname{mean}(h_2) + \cdots + \operatorname{mean}(h_n)\\
&= 0 + \frac{1}{2} + \cdots + \frac{n-1}{2} = \frac{n(n-1)}{4}; \qquad (12)\\
\operatorname{var}(g_n) &= \operatorname{var}(h_1) + \operatorname{var}(h_2) + \cdots + \operatorname{var}(h_n)\\
&= 0 + \frac{1}{4} + \cdots + \frac{n^2-1}{12} = \frac{n(2n+5)(n-1)}{72}. \qquad (13)
\end{aligned}$$

所以反序的平均数相当大，大约为 $\frac{1}{4}n^2$. 标准差也相当大，大约为$\frac{1}{6}n^{3/2}$.

珀西·麦克马洪发现了有关反序分布的一个重要事实 [*Amer. J. Math.* **35** (1913), 281–322]. 将排列 $a_1\,a_2\ldots a_n$ 的指数定义为所有满足 $a_j > a_{j+1}$, $1 \le j < n$ 的下标 j 之和. 例如，5 9 1 8 2 6 4 7 3 的指数为 $2+4+6+8=20$. 在本例中，这一指数碰巧与反序的数目相同. 如果列出 $\{1,2,3,4\}$ 的 24 个排列，即：

排列	指数	反序数目	排列	指数	反序数目
1 2 3 4	0	0	3|1 2 4	1	2
1 2 4|3	3	1	3|1 4|2	4	3
1 3|2 4	2	1	3|2|1 4	3	3
1 3 4|2	3	2	3|2 4|1	4	4
1 4|2 3	2	2	3 4|1 2	2	4
1 4|3|2	5	3	3 4|2|1	5	5
2|1 3 4	1	1	4|1 2 3	1	3
2|1 4|3	4	2	4|1 3|2	4	4
2 3|1 4	2	2	4|2|1 3	3	4
2 3 4|1	3	3	4|2 3|1	4	5
2 4|1 3	2	3	4|3|1 2	3	5
2 4|3|1	5	4	4|3|2|1	6	6

我们看到，拥有指数 k 的排列数目与拥有 k 个反序的排列数目相同.

初看起来，这一事实似乎是显而易见的，但进一步仔细研究后会发现它非常难以理解. 麦克马洪给出了一种巧妙的间接证明：设 $\operatorname{ind}(a_1\,a_2\ldots a_n)$ 表示排列 $a_1\,a_2\ldots a_n$ 的指数，并设

$$H_n(z) = \sum z^{\operatorname{ind}(a_1\,a_2\ldots a_n)} \tag{14}$$

为相应的生成函数. (14) 的右侧是针对 $\{1,2,\ldots,n\}$ 的所有排列求和. 我们希望证明 $H_n(z) = G_n(z)$. 为此，我们将定义一种一一对应关系，一端是由非负整数组成的 n 元组 $(q_1,q_2,\ldots,q_n)$，另一端是 n 元组的有序对

$$\big((a_1,a_2,\ldots,a_n),\,(p_1,p_2,\ldots,p_n)\big),$$

其中 $a_1\,a_2\ldots a_n$ 为下标 $\{1,2,\ldots,n\}$ 的一个排列，且 $p_1 \ge p_2 \ge \cdots \ge p_n \ge 0$. 这一对应关系满足条件

$$q_1+q_2+\cdots+q_n = \operatorname{ind}(a_1\,a_2\ldots a_n) + (p_1+p_2+\cdots+p_n). \tag{15}$$

针对所有非负整数 n 元组 $(q_1,q_2,\ldots,q_n)$ 求和的生成函数 $\sum z^{q_1+q_2+\cdots+q_n}$ 为 $Q_n(z) = 1/(1-z)^n$. 针对所有整数 n 元组 $(p_1,p_2,\ldots,p_n)$（其中 $p_1 \ge p_2 \ge \cdots \ge p_n \ge 0$）求和的生成函数 $\sum z^{p_1+p_2+\cdots+p_n}$ 为（如习题 15 所示）

$$P_n(z) = 1/(1-z)(1-z^2)\ldots(1-z^n). \tag{16}$$

依据式 (15)，我们将要建立的一一对应关系将会证明 $Q_n(z) = H_n(z)P_n(z)$，即

$$H_n(z) = Q_n(z)/P_n(z). \tag{17}$$

但根据 (8)，$Q_n(z)/P_n(z)$ 为 $G_n(z)$.