经典计算机科学著作最新修订版

计算机程序设计艺术

第3卷

排序与查找

（第2版）

[美] Donald E. Knuth 著
苏运霖 译

国防工业出版社
·北京·

著作权合同登记　图字：军-2001-020 号

图书在版编目(CIP)数据

计算机程序设计艺术．第 3 卷，排序与查找：第 2 版/（美）克努特（Knuth，D.E.）著；苏运霖译．—北京：国防工业出版社，2002.9
书名原文：The Art of Computer Programming
ISBN 7-118-02812-6

Ⅰ.计…　Ⅱ.①克…②苏…　Ⅲ.程序设计
Ⅳ.TP311

中国版本图书馆 CIP 数据核字(2002)第 004601 号

互联网网页 http://www-cs-faculty.stanford.edu/~knuth/taocp.html 包含本书及相关著作的最新信息。

国防工业出版社出版发行
（北京市海淀区紫竹院南路 23 号）
（邮政编码　100044）
北京奥隆印刷厂印刷
新华书店经营
*
开本 787×960　1/16　印张 50½　插页 1　1041 千字
2002 年 9 月第 1 版　2002 年 9 月北京第 1 次印刷
印数：1—4000 册　定价：98.00 元

前　言

Cookery is become an art,
a noble science;
cooks are gentlemen.
烹调法成了一种艺术,
一门高深的科学;
厨师是有教养的人。
— TITUS LIVIUS, *Ab Urbe Condita* XXXIX. vi
(Robert Burton, *Anatomy of Melancholy* 1.2.2.2)

本书形成了第1卷第2章关于信息结构内容的自然续篇,因为它为第1卷的基本结构化思想增加了线性有序数据的概念。

本书书名"排序与查找"可能给人以印象,仿佛它仅仅是为那些关心通用排序程序编制或信息检索应用的系统程序员编写的。然而,事实上排序与查找的领域还提供了一个理想的园地,以讨论范围广泛而且很重要的一般性问题:

·如何发现好的算法?
·如何改进给定的算法或程序?
·如何从数学上分析算法的效率?
·对于相同的任务,人们如何在不同的算法之间进行合理的选择?
·在什么意义之下,可以证明某个算法是"最好"的?
·计算理论同实际考虑如何相互影响?
·像磁带、磁鼓或磁盘这样的外存,如何能有效地用于大型数据库?

其实,我确信,程序设计的每一个重要方面,实际上都离不开排序或查找!

本卷由全套书的第5章和第6章组成。第5章讨论的是排序;这是一个很大的课题,已经把它主要划分成两大部分,即内部排序和外部排序。还有补充的几节,提出了关于排列(5.1节)和最优排序算法(5.3节)的辅助理论。第6章讨论了在表或文件中查找特定项目的问题;这又细分成顺序查找、通过键码(key)比较查找、按数字性质查找、"散列"查找等几种方法,而后考虑了比较困难的辅键码(secondary key)查找问题。在这两章之间存在着令人惊奇的相互影响以及强烈的类似。除了在第2章中所讨论的信息结构以外,这里还讨论了信息结构的两种重要的变形,即优先队列(5.2.3小节)和表示成平衡树的线性表(6.2.3小节)。

和第1卷及第2卷一样,本书包括了大量在其它出版物上从未出现过的许多内

容。许多人善意地写信或口头向我表达了他们的看法，我希望，当我以我自己的文字来表现这些内容的时候，我没有很糟糕地曲解它们。

我未曾有时间系统地查找专利文献；确实，我蔑视当前寻求算法专利的倾向（参见 5.4.5 小节）。如果某人把在本书中未曾引用的有关专利的一个副本寄给我，那我将在未来的版本中恭敬地引用它。然而我要鼓励人们继续发扬悠久的数学传统，把最新发现的算法投进公众领域。与其防范他人从某个人对计算机科学的贡献中获益，不如寻求其它更好的谋生手段。因为它与防范自己的计算机成果被他人利用相比会给人们带来更好的生活。

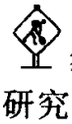
在我从教学工作上退休下来之前，我曾把这本书用作大学三年级至研究生层次数据结构第二门课的教材，并省略掉了大部分数学内容。我也曾使用本书的数学部分作为研究生层次算法分析课程的基础，并特别强调了 5.1，5.2.2，6.3 和 6.4 诸节。关于具体的计算复杂性方面的研究生课程也可以以 5.3 节和 5.4.4 小节，以及第 2 卷的 4.3.3，4.6.3 和 4.6.4 诸小节为基础。

除了偶尔讨论第 1 卷中说明的 MIX 计算机外，本书的绝大部分内容都是独立、自成体系的。附录 B 包含有对于所使用的数学记号的概述，其中的一些记号同传统的数学书中找到的记号有些差别。

第 2 版前言

这个新版本是同第 1 卷及第 2 卷的第 3 版相匹配的。那两卷的出版用上了 TEX 和 METAFONT 系统，使我得以庆祝这两个专门为本套书开发的出版系统的完成。

本书转换成电子格式使我有机会重新检查正文的每一个字和每一个标点符号。在添加或许某些更成熟的记述的同时，我试图保留原来文字的朝气。已经增加了数十个新的习题，对数十个老习题也给出了新的改进了的答案。变动随处可见，但最主要的是在 5.1.4 小节（关于排列和图表），5.3 节（关于最优排序），5.4.9 小节（关于磁盘排序），6.2.2 小节（关于熵），6.4 节（关于通用散列法）以及 6.5 节（关于多维树和检索结构）。



然而，《计算机程序设计艺术》的写作仍然是进行中的工作。关于排序与查找的研究仍然以非凡的速度发展着。因此本书的某些部分加上了“正在施工”的图标以对该部分内容还不是最新表示歉意。例如，如果我今天仍向本科生讲授数据结构，我肯定将花些时间讨论像树积（treap）这样的随机化结构；但在本书中，我只能引用有关这个课题的主要文章，告诉大家将来的 6.2.5 小节的写作计划。我的文件夹里已充满了许多重要的素材，我打算从现在起用大约 17 年时间，把这些素材包含在

第3卷最后的辉煌的第3版中！但是在此之前我必须首先完成第4卷和第5卷，而且除非绝对必要，否则我不想使它们的问世再有任何拖延了。

我对于过去35年来帮助我搜集和改进本书内容的数百位人们表示衷心的感谢。准备这一新版本的大部分艰苦的工作是由Phyllis Winkler（他把第1版的文本转换成TEX的形式）和Silvio Levy（他编辑文字并绘制了数十张插图），以及Jeffrey Oldham（他把原来250张以上的插图转换成METAPOST格式）完成的。Addison-Wesley生产部的同仁们也一如既往地提供了极大的帮助。

我已经改正了细心的读者们在第1版中发现的，以及天啊，竟无人发觉的那些错误，而且，我已竭尽全力试图避免在新的内容中再引进新的错误。然而，我设想，某些欠缺之处仍在所难免，我想尽早地改正它们。因此我将很乐意地支付2.56美元给每一个技术、印刷或历史错误的头一个发现者*。在版权页上所引的网页中包含了已向我报告过的所有错误的更正。

Stanford, California　　D. E. K.
1998年2月

There are certain common Privileges of a Writer,
the Benefit whereof, I hope, there will be no Reason to doubt;
Particularly, that where I am not understood, it shall be concluded,
that something very useful and profound is concht underneath.
一位作者享有某些公认的特权，
我认为，这件事的好处没有理由加以怀疑；
特别是，在我还不被人们理解的地方，
背后蕴涵着某种非常有用和意义深远的东西。
—JONATHAN SWIFT, *Tale of a Tub*, Preface (1704)

* 中文版已按2001年10月底的最新勘误表更正了原书的错误。网页 http://www.ndip.com.cn/computer/taocp 上也将列出对中文版的勘误表及从作者处获得的最新信息。——本书责任编辑

关于习题的说明

本套书的习题既可用于自学，也可用于课堂练习。无论是谁，如果想纯粹地通过阅读，而不将所阅读的信息应用到特定问题上，并由此牵引思考先前阅读的内容，就想学到一门学问，纵然可能，那也是很困难的。其次，对于我们自己的发现，我们总是领会得最透。因此，习题形成了这一套书的一个重要部分；我着意使这些习题含有丰富的信息，而且也尽量选择既有趣又有启发性的习题。

在许多书里，容易的习题被随机地混杂于极端困难的问题当中。有时这是不合适的，因为读者预先想要知道一道习题花多少时间——否则他们可能就越过这些习题了事。这一情况的典型例子是由 Richard Bellman 所著的*Dynamic Programming* 一书。这是一本重要的先驱性的论著，其中在某些章的末尾，在"习题和研究题"的标题下，把一组问题收集在一起，而且一些极其平凡的问题出现在一些深入的、还未被解决的问题当中。据说，有人曾问过 Bellman 博士，怎样区分习题和研究题。他回答说："如果你能解决它，那它就是一道习题，否则它就是一个研究题。"

但在这一类书里把研究问题和很容易的习题都包括进来确有其道理；因此，为使读者免于陷入确定哪是习题哪是研究题的困境，我给习题注明了等级分，以指出困难的程度。这些分数大体具有下列意义：

分数	解释
00	一个极其容易的习题。如果你已理解正文的内容，就可能立即做出回答。这样一道题几乎总是可以"眉头一皱"就把它做出。
10	一个简单的问题，它要求你去思考刚刚学过的内容，但绝不意味着是困难的。你应当有能力在顶多一分钟之内就把它做出。在获得解答的过程中可能要用到笔和纸。
20	一个普通的问题。它检查你对正文内容的基本理解，但你可能需要 15 或 20 分钟才能完整地回答它。
30	一个中等难度和/或复杂的问题。这个题目可能需要两个小时以上的工作才能令人满意地解决。或者甚至更长时间，如果电视机在开着的话。
40	确实是一个十分困难或冗长的问题。在学校里，它将适合于作为一个学期的课程设计。一个学生应当有能力在相当长的时间里来解决它，但这个解不会是平凡的。
50	就作者在编写本书时所知，这是一个还未被令人满意地解决的问题，尽管已经有很多人做了尝试。如果你已经找到了这样一个问题的答

案，你应当把它写出来发表。其次，本书的作者将乐意尽快地听到关于解答的消息（当然，假定它是正确的）。

通过在这个“对数”尺上的内插，其它分数的意义也就清楚了。例如 17 分将表示比普通的题更为简单的一道习题。一个随后被某个读者解决了的 50 分的题在本书后来的版本中将以 45 的分数出现，而且会在互联网上的勘误表中刊出（参见版权页）。

分数除以 5 的余数表示所要求的详细工作量。因此分数是 24 的习题可能要比分数是 25 的题花费更长的时间来求解，但后者将要求更多的创造性。

作者已经认真地试图指定精确的分数，但是提出问题的人要想确切地知道对于求解的另一个人，它的难度如何，肯定是困难的；而且每个人都会有较其他人更为适应的问题类型。只能希望这些分数较好地反映了习题的困难程度，希望读者把它们当做是一般的导引，而不应作为绝对的指标。

本书是为具有不同程度的数学训练和素养的读者写的；因此有些习题是特意为有更多的数学基础的读者提供的。如果一道题的出题动机或涉及的数学概念超越了主要兴趣仅仅是算法编程本身的读者应掌握的程度，则在分数前边加上 *M*。如果一道习题的答案必须涉及在本书中未予提供的微积分或其它高等数学知识，则对这道题标以“*HM*”的字母。“*HM*”标记并不必定意味着困难。

某些习题的前边标有三角符“▶”；这一标志说明是特别有启发性的问题，因而特别予以推荐。当然，并不期望读者/学生来求解所有的习题。所以标出了那些看起来最有价值的部分（这并不意味着贬低其它习题！）。每个读者至少应该尝试去解分数是 10 或者更低的所有问题；箭头可以帮助指出应该对具有较高分数的哪些问题予以优先考虑。

在答案部分中已经列出大多数习题的解答，请明智地使用它们。在你已真正地做出努力亲自解决问题之前，不要去翻答案，除非你确实没有时间来做这一特定的问题。在获得了你自己的解答或者对该题做了郑重的尝试之后，你可能会感到答案是有启发和有帮助的。给出的解答通常十分简短，作者认为你已经认真地通过自己的方法做过求解尝试因而略去了其细节。有时解答所提供的信息比所要求的为少，但通常都是更多的。十分可能，你有比这里给的更好的答案，或者在答案中你发现一个错误。在这样的情况下，作者将乐意知道详细的情况。在本书以后的版本中将在适当地方刊登出这些改进了的答案以及提供这些解的人的姓名。

当做一道习题时，一般地你可以使用前边习题的答案，除非明确地禁止这样做。在对习题打分时，已经考虑到这一点了，因此很可能第 $n+1$ 题的分数比第 n 题还要低，尽管它把第 n 题的结果作为一个特殊情况包括进来。

代码含义		
	00	立即可解的
	10	简单的（--分钟）
	20	一般水平（一刻钟）
▶特别推荐的	*30*	难度适中
M 面向数学的	*40*	学期设计
HM 要求“高等数学”的	*50*	研究题

习　题

▶1.[*00*] 分数"*M20*"意味着什么?

2.[*10*] 一本教科书中的习题对读者有什么价值?

3.[*HM45*] 证明当 n 是一个整数且 $n>2$ 时,方程 $x^n+y^n=z^n$ 无正整数 x,y,z 的解。

Two hours' daily exercise . . . will be enough
to keep a hack fit for his work.
每天两小时的训练……将足以
令一匹马很好地工作。
—M. H. MAHON, *The Handy Horse Book* (1865)

目　录

第5章　排　序

第6章　查　找

第5章　排　　序

There is nothing more difficult to take in hand, more perilous to conduct, or more uncertain in its success, than to take the lead in the introduction of a new order of things.

没有什么比带头建立事物的新秩序更难掌握，更担风险，更高深莫测的了。

—NICCOLÒ MACHIAVELLI, *The Prince* (1951)

"But you can't look up all those license numbers in time," Drake objected.

"We don't have to, Paul. We merely arrange a list and look for duplications."

"但是你不能及时查出全部那些执照号"，Drake 争辩说。

"没有必要，Paul。我们只要列一张表，并把重复的找出来。"

—PERRY MASON, in *The Case of the Angry Mourner* (1951)

"Treesort" Computer—With this new 'computer-approach' to nature study you can quickly identify over 260 different trees of U. S., Alaska, and Canada, even palms, desert trees, and other exotics. To sort, you simply insert the needle.

"树排序"计算机——应用这种新的"计算机方法"来进行树种研究，你能快速地确认260种以上美国、阿拉斯加和加拿大的不同树种，甚至棕榈树、沙漠里的树，以及其它外来树种。要想排序，你只须插入这颗针就行了。

—Catalog of Edmund Scientific Company (1964)

在这一章里，我们将研究在程序设计中经常出现的一个课题：以递增或递减的次序重新排列项目。我们可以设想一下，倘若字典中的词不是以字母的顺序排列，

那么使用这样的字典将是何等困难！同样，存在于计算机存储器中的各项的次序，对于处理这些项目的算法的速度及简便性来说，也有着重要的影响。

尽管英语词典里把“sorting”一词定义为按照种类或类型，分开或安排事物的过程，但是计算机程序员习惯于在更为特殊的意义下来使用它，即把事物排成递增或递降的次序。这个过程也许应称做 ordering，而不是 sorting；但任何试图把它称为“ordering”的人很快就会导致混乱，因为这个词已被附加了许多不同的意义。例如，考虑下边的句子：“由于我们只有两台磁带机处于工作状态(working order)，我被要求(was ordered)去以快速订货的形式(in short order)订购(order)更多的磁带机，以便(in order)以快若干数量级(order)的速度对数据进行排序(order)”。在数学术语中，“order”的意义太多了(群的阶(order)，置换的阶(order)，分支点的级数(order)，次序(order)的关系，等等)。所以我们可以得出结论，“order”一词容易导致混乱。

某些人建议用“sequencing”作为排序过程的适当名称，但是这个词通常缺乏恰当的涵义，特别是当出现相等的元素时，而且，它有时同其它术语冲突。实际上，“sorting”本身也是一个被滥用了的词(如：在对那一类(sort)数据进行排序(sorting)以后，我有几分(sort of)不快(out of sorts))，但它已经在计算的用语中被确认并使用了。因此，我们将把“排序”(sorting)一词用于“排成有序”的严格意义之下，而不去作进一步的辩解。

排序的某些最重要的应用是：

a) *解决“一起性”的问题*，即把具有相同标志的所有项目连在一起。假设我们有随机次序的 10 000 个项目，其中许多有相等的值；而且假设我们要重新排列数据，使得具有相等值的所有项出现在连续的位置上。这实际上是该词传统意义下的“sorting”问题；通过在该词的新意义下对文件排序，可以容易地解决这个问题，使得这些值按递增的次序 $v_1 \leqslant v_2 \leqslant \cdots \leqslant v_{10000}$ 排好。在这个过程中可能达到的效率，说明了为什么“sorting”原来的意义已经改变。

b) *匹配在两个或更多个文件中的项*。如果若干个文件已经按次序排好，则通过顺序地扫描一趟它们，就可能从中找出所有匹配的项，而无须返回去查。这是 Perry Mason 在解决一个谋杀案件时用到的原理(见本章开头的引用语)。从开始到末尾顺序地遍历一个信息表，而不是在表中随机地来回跳动，通常可以最快速地处理一个信息表，除非这个表足够小，可以存入到高速随机存取的存储器中。排序使得有可能对大型文件使用顺序存取，从而有可能取代直接寻址。

c) *通过键码值查找信息*。如同我们将在第 6 章中要见到的，排序也有助于查找，因此它也有助于使计算机的输出更适合人们的需要。事实上，按字母顺序排序的一份清单，往往看上去十分可信，即使有关的数字信息计算得不正确也是如此。

尽管排序传统上多用于商业数据处理，但它实际上是程序员应当掌握的运用广泛的一项基本工具。在习题 2.3.2-17 中，我们已经讨论了它在简化代数公式中的用途。以下的习题说明各种各样的典型应用。

展示排序多样性最初的大型软件系统之一，是由 J. Erdwinn 和 D. E. Ferguson，以及他们在 Computer Sciences Corporation(计算机科学公司)的助手们于 1960 年开

发的 LARC Scientific Compiler(拉克科学编译语言)程序。这个用于扩充的 FORTRAN 语言的优化编译程序,大量使用了排序,使得各种编译算法同源程序的有关部分都以适当的顺序出现。第一遍是词法扫描,它把 FORTRAN 源代码分成单个的记号,每个表示一个标志符,或者一个常数,或者一个操作符等等。每个记号被赋以若干顺序号;当对名字和适当的顺序号进行排序时,一个给定标志符的所有使用就联系在一起了。一个用户将用之来确定标志符是否代表一个函数名,一个参数或一个数组变量的"定义项",被给予一个低的顺序号,以使它们在具有一个给定标志符的诸记号当中首先出现;这样,既便于校验一个标志符的使用是否有冲突,又便于相对于 EQUIVALENCE 说明符来分配存储。以这种方式收集在一起的每个标志符的信息现在附加在每个记号上;这样在高速存储器中就不需要保留标志符的"符号表"了。更新过的记号然后按另一个序列号排序,它基本上是把源程序恢复到它原来的顺序,只是该序列是特殊设计的,它把算术表达式变成为更方便的"波兰前缀"形式。排序也用于编译的稍后阶段,以便于进行循环优化,把错误信息合并到清单当中,等等。简言之,编译程序被设计成,使得实际上可以通过顺序地处理存于辅助磁鼓内的文件而完成工作,因为适当的顺序号,是以这样一种方式附加在数据上的,即数据能被排成各种方便的序列。

20 世纪 60 年代的计算机厂家估计,当把他们的所有顾客都考虑在内时,在他们的计算机上,将有超过 25% 的运行时间花在排序上。事实上,有许多计算机装置,仅其中的排序,就用去计算时间的一半以上。由这些统计,可以得出结论:(i)排序有许多重要的应用;(ii)当不应当进行排序时,许多人却进行了;(iii)低效的排序算法正被普遍地使用着。实际情形可能包括所有这 3 种可能性。但无论如何,排序都值得作为一个实际问题而认真地加以研究。

即使排序毫无用处,也仍然有许多理由对它进行研究! 业已发现的巧妙算法表明,排序本身就是一个值得剖析的极其有趣的课题。在这方面,还有很多有魅力的悬而未决的问题,当然也有不少已经解决了。

从一个更广泛的方面着眼,我们也会发现,对于一般情况下如何着手解决计算机程序设计的问题,排序算法即是一个有价值的实例分析。在这一章我们将说明数据结构操作的许多重要原理。我们将考察各种排序技术的演化,以期向读者指出,这些思想首先是如何被发现的。通过外推这一实例分析,我们可以学到许多好的策略,来帮助我们就其它计算机问题设计好的算法。

排序技术也为包含于算法分析中的一般思想提供了极好的说明——这些思想通常用来确定算法的性能特征,以便在不同的方法之间进行明智的选择。专长数学的读者,将在这一章中找到用于评价计算机算法的速度和解决复杂的递归关系的,许多有启发的技术。另一方面,由于已对内容做了适当的编排,使得那些并不很擅长数学的读者也能跳过这些计算而不受影响。

在往下进行之前,应该更清楚地来定义我们的问题,并且介绍一些术语。给定有待排序的 N 个项目

$$R_1, R_2, \cdots, R_N$$

我们称这些项目为记录，并称 N 个记录的整个集合为一个文件。每个记录 R_j 有一个键码 K_j，它支配着排序的过程。在一个记录中，除这个键码外，还可能有其它的信息；这些额外信息，除非必须作为记录的一部分来一起处理，否则对于排序过程没有影响。

在键码上确定一个次序关系“$<$”，使得对于任意 3 个键码值 a, b, c，下列条件成立：

i) $a<b, a=b, b<a$ 三个可能性中恰有一个可能性成立（这是三分律）。

ii) 如果 $a<b$，而且 $b<c$，则 $a<c$（这是熟知的传递律）。

性质 i) 和 ii) 表征了线性次序的数学概念，也称做全序。任何满足 i) 和 ii) 的关系“$<$”都可以按本章中即将介绍的大多数方法进行排序，尽管有些排序技术被设计成只对具有通常次序的数字或字母键码有效。

排序的目标，是确定下标为 $\{1,2,\cdots,N\}$ 的一个排列 $p(1)p(2)\cdots p(N)$，它以非递降的次序来放置所有键码：

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

如果我们提出进一步的要求，即具有相同键码的记录保留它们原来的相对次序，则称这个排序是稳定的。换句话说，稳定排序有下列附加的性质，即

$$p(i) < p(j) \quad \text{每当 } K_{p(i)} = K_{p(j)} \text{ 且 } i < j \text{ 时} \tag{2}$$

在某些情况下我们可能要求在存储器中对记录进行物理上的重新排列，使其键码成为有序的。而在其它一些情况下，可能仅仅需要一张辅助表以某种方式确定这个排列，以便这些记录能根据其键码的次序来加以存取。

本章中有一些排序方法假定了值“∞”或“$-\infty$”，它们分别地定义为大于或小于所有键码：

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

这些极值有时被用做人为的键码，也用做标志（sentinel）指示器。(3) 中排除了相等的情况，如果出现相等，可以对这些算法进行修改，使得它们仍然有效，但通常要损失一些简洁性和有效性。

排序一般可分为内部排序和外部排序。在内部排序中，记录被整个地保留在计算机的高速随机存取存储器中；当记录比内存一次所能容纳的还多时，则使用外部排序。内部排序在构造和存取数据方面具有更多的灵活性，而外部排序则告诉我们如何应对更为严格的存取约束。

利用一个较好的通用排序算法，对 N 个记录进行排序所需要的时间，大约同 $N \log N$ 成比例；我们对数据大约扫描 $\log N$ “趟”。如同在 5.3.1 节将看到的，如果所有记录的次序随机而且排序是通过键码的两两比较进行的，则这是极小的时间。于是，如果把记录的数目加倍，则将花略多于两倍的时间对它们进行排序，而所有其它方面不变（实际上，当 N 趋于无穷时，如果键码不同的话，则排序所需时间的一个更好表示是 $N(\log N)^2$，因为键码的大小必须至少以 $\log N$ 的速度增长；但对于实际使用来说，N 事实上绝不会趋于无穷）。

另一方面，如果已知关于某个连续的数值分布，诸键码是随机分布的，则我们将看到，平均说来，我们可以在 $O(N)$ 步内实现排序。

习　题——第一组

1.[*M20*] 由三分律和传递律证明，当假定排序稳定时，排列 $p(1)p(2)\cdots p(N)$ 是惟一确定的。

2.[*21*] 假定某文件中的每个记录 R_i 包含两个键码，一个"主键码" K_i 和一个"次键码" k_i，而且每一个键码集合中各定义了一个线性次序 $<$。则我们可以用通常的方式在键码对 (K,k) 之间定义一个字典序：

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

Alice 取这个文件，并且首先按主键码对它进行排序，得到 n 组记录，且对于每组记录，有相同的主键码

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

其中 $i_n=N$。然后，她按各组的次键码对 n 个组 $R_{p(i_{j-1}+1)},\cdots,R_{p(i_j)}$ 中的每一个进行排序。

Bill 取同一个原始文件，首先按次键码进行排序，然后对得到的文件按主键码进行排序。

Chris 取同一个原始文件，按主键码和次键码 (K_j,k_j) 的字典序对它进行一次排序操作。

问每个人是否得到相同的结果？

3.[*M25*] 设 $<$ 是 $K_1,\cdots,K_N$ 上的一个关系，它满足三分律，但不满足传递律。证明，即使没有传递律，也有可能以一种稳定的方式对记录进行排序，且满足条件(1)和(2)；事实上，至少有 3 个满足这些条件的排列方式！

▶4.[*21*] 实际上词典中并未使用严格的字典序，因为大小写字母都必须参与排列。它们要有以下这样一个顺序：

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

说明如何来实现字典序。

▶5.[*M28*] 试对所有的非负整数设计一个二进制编码，使得如果把 n 编码为串 $\rho(n)$，则我们有 $m<n$ 当且仅当 $\rho(m)$ 按字典序小于 $\rho(n)$。而且，对于任何 $m\neq n$，$\rho(m)$ 不应是 $\rho(n)$ 的一个前缀。如果可能，对于所有大的 n，$\rho(n)$ 的长度应当至多为 $\lg n+O(\log\log n)$（如果我们要对文字和数字混合的文本进行排序的话，或者如果我们要把任意大的字母表映射成二进制串的话，则这样一个编码是有用的）。

6.[*15*] B. C. Dull 先生（一个 MIX 程序员）要想知道存在于单元 A 中的数是否大于、小于或等于存在于单元 B 中的数，所以，他写语句"LDA A;SUB B"并测试寄存器 A 是正，是负或是 0。他犯了什么严重错误？他应该如何做？

7.[*17*] 按下列说明写出多精度键码比较的 MIX 子程序：

调用序列：JMP COMPARE

入口条件：rI1 = n；对于 $1\leqslant k\leqslant n$，CONTENTS(A + k) = a_k，CONTENTS(B + k) = b_k；

假定 $n\geqslant 1$。

出口条件：CI = GREATER，　如果 $(a_n,\cdots,a_1)>(b_n,\cdots,b_1)$；

CI = EQUAL，　如果 $(a_n,\cdots,a_1)=(b_n,\cdots,b_1)$；

CI = LESS，　如果$(a_n,\cdots,a_1)<(b_n,\cdots,b_1)$；

rX 和 rI1 可能要受影响。

这里，关系$(a_n,\cdots,a_1)<(b_n,\cdots,b_1)$表示从左到右的字典序；即，对于 $n\geqslant k>j$，有一个下标 j，使得 $a_k=b_k$，而 $a_j<b_j$。

▶8.[*30*] 单元 A 和 B 分别存放两个数 a 和 b。证明，有可能写出一个 MIX 程序，它计算并保存 $\min(a,b)$于单元 C 中，而不使用任何转移操作符。（警告：由于你不可能测试是否出现算术溢出，所以确保无论是 a 值还是 b 值都不会出现溢出，是明智的。）

9.[*M27*] 在 n 个独立地一致分布于 0 和 1 之间的随机变量排为非递减顺序之后，这些变量中第 r 个最小者小于等于 x 的概率是多少？

习　题——第二组

下列每一个习题指出了，在以前计算机还只有较少随机存取内存的时候，计算机程序员所要解决的问题。假设仅有几千字的内存可供利用，通过大约半打的磁带机作为补充（对于排序来说这些磁带机足够了），试提出解决这个问题的“好”方法。在这样的限制之下工作得很好的算法，证明在现代机器上也是有效的。

10.[*15*] 给你一条含有 100 万个字的磁带。你如何确定该磁带中有多少字不同？

11.[*18*] 你是美国国内税收服务人员，你收到了来自各单位的数百万“信息”表格，报告他们已向人们支付了多少钱；同时，又从人们那里收到了数百万“税收”表格，报告他们已经得到了多少收入。你如何发现没有报告他们所有收入的人？

12.[*M25*]（转置一个矩阵）给你一条含有 100 万个字的磁带，这些字表示按行顺序存储的 1000×1000 矩阵的元素：$a_{1,1}\ a_{1,2}\cdots a_{1,1000}\ a_{2,1}\cdots a_{2,1000}\cdots a_{1000,1000}$。你如何建立一条磁带，其中元素是按列的次序 $a_{1,1}\ a_{2,1}\cdots a_{1000,1}\ a_{1,2}\cdots a_{1000,2}\cdots a_{1000,1000}$ 存储的（试试看，最多只对数据扫描 10 趟）？

13.[*M26*] 你如何把 N 个字的一个大型文件“洗”成随机排列？

14.[*20*] 你正在用两个计算机系统进行工作，它们对字母数字的排列方式有不同的约定。你如何使一台计算机按另一台计算机使用的次序对字母数字文件进行排序？

15.[*18*] 给你一份出生在美国的附有其出生州名的人员名单，名单上的人很多，你如何计算出每个州出生的人数（假设这个名单中每个人只出现一次）？

16.[*20*] 为了易于改动大型 FORTRAN 程序，你需要设计一个“交叉引用”程序；这样一个程序以 FORTRAN 程序作为输入，并连同一个索引一起把它们打印出来，索引说明程序中每个标志符（即每个名字）的各次使用。你如何来设计这个程序？

▶17.[*33*]（图书馆卡片排序）在实现计算机化的数据库之前，每个图书馆都有一个卡片目录，以便使用者可以找到他们所要的书。但随着图书馆藏书的增加，将目录卡片排序以便于人们使用这项工作变得十分复杂。下面的“按字母顺序的”清单指出了*American Library Association Rules for Filing Catalog Cards*（Chicago，1942）中建议的许多规程：

卡片的正文	注　释
R. Accademia nazionale dei Lincei, Rome	Ignore foreign royalty (except British)
1812; ein historischer Roman.	Achtzehnhundert zwölf

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 süssen Mädeln.	…provided it's in nominative case
Dix, Morgan, 1827—1908	Names precede words
1812 ouverture.	Dix-huit cent douze
Le XIXe siècle fran cais.	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, Muhammad 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 upper case letter

这些规则大多数都有若干例外，而且还有许多其它规则这里未予说明。

如果你的工作就是用计算机对大量图书馆的目录卡片进行排序，并最终维护这样一个庞大的卡片文件，而且你也没有机会去改变卡片归档的这些长期有效的规则，那么你应怎样安排数据，以便于进行排序和合并操作？

18. [*M25*] (E. T. Parker) 欧拉曾经猜测[*Nova Acta Acad. Sci. Petropolitanæ* **13**(1795), 45～63, §3; 写于 1778 年]，方程

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

没有正整数的解 u, v, w, x, y, z。同时，他猜测，对于所有的 $n \geqslant 3$

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

没有正整数的解。但是这个猜测已被计算机发现的恒等式 $27^5 + 84^5 + 110^5 + 133^5 = 144^5$ 所否定，参见 L. J. Lander, T. R. Parkin 和 J. L. Selfridge, *Math. Comp.* **21** (1967), 446～459。后来，Noam Elkies 又发现了 $n = 4$ 时无穷多的反例[*Math. Comp.* **51**(1988), 825～835]。你能否想出一个办法，其中的排序将有助于找到当 $n = 6$ 时欧拉猜测的反例？

▶19. [*24*] 给定一个包含大量不同的 30 位二进制字 $x_1, \cdots, x_N$ 的文件，找出其中所有补码对