《编程珠玑(第二版)》

图书基本信息

书名：《编程珠玑(第二版)》

13位ISBN编号：9787508319148

10位ISBN编号：7508319141

出版时间：2004-4

出版社：中国电力出版社

作者：本特利

页数：217

译者：本特利著

《编程珠玑(第二版)》

内容概要

《编程珠玑(第2版)》是计算机科学方面的经典名著。书的内容围绕程序设计人员面对的一系列实际问题展开。作者Jon Bentley 以其独有的洞察力和创造力，引导读者理解这些问题并学会解决方法，而这些正是程序员实际编程生涯中至关重要的。

《编程珠玑(第二版)》

作者简介

Jon Bentley，世界著名计算机科学家，被誉为影响算法发展的十位大师之一。他先后任职于卡内基—梅隆大学(1976—1982)、贝尔实验室(1982—2001)和Avaya实验室(2001年至今)。在卡内基—梅隆大学担任教授期间，他培养了包括Tcl语言设计者John Ousterhout、Java语言设计者James Gosling、《算法导论》作者之一Charles Leiserson在内的许多计算机科学大家。2004年荣获Dr．Dobb's程序设计卓越奖。

《编程珠玑(第二版)》

书籍目录

《编程珠玑(第二版)》

精彩短评

1、我这几年读过最好的程序书籍
2、只记得读过，但很惭愧，已经完全记不清书写的是啥内容了
3、基本功啊基本功，当年看的时候不懂，现在懂了一点，感觉压力好大
4、都说大道至简，这本书主要关注程序的效率，而且有很强的扩展性，要多复习
5、算法实践，真实的程序员而非算法的罗列。
6、这本书是……名符其实的面试宝典。然而书中的许多知识确实如珠玑一样宝贵，这是一本能够提高一个程序能力的书，但它却这么薄！
7、读过一章。。。。。
8、内容很精彩，可惜这翻译，哎
9、写的挺好的，有的复杂的未看懂，温习了一些知识
10、那个站着会出错坐着不会出错的可怜程序真是太搞笑了。而且作者的跋里的精分流的写作风格更搞笑，第二版跋还煞有介事的搞了句名言然后继续精分。。此书作为睡前读物非常合适，每个段子都不长，睡前来一发，思考思考，然后就着了。。
11、经典中的经典，没的说
12、算法的灵感来自数学知识和生活经验，这本书讲的内容不多，不过很有用
13、翻译太烂了。
14、这两颗星是给翻译的
15、看了总感觉收获很少，确切地说就是看了之后能为我所用的东西几乎为0，可能不是每个人都能品味珠玑，或许确实是好书，只是有些好东西只能对某些人来说是好东西。这本书的前言有不少牛人说，此书如何如何，我就是被这些话所动。看来选书不能太相信别人的评论。
16、几年后再看第二次吧 太菜了
17、书看得不够明白吧。觉得有些东西很简单，有些东西很经典。
18、已购.
19、好多看不懂
20、最有价值的是程序正确性证明一章，而且该种手法贯穿全书，熟练过后，对提高个人代码质量有很大帮助。另一个可以参考的是，数据驱动的单元测试
21、很不错的一本书，货送得也很准时!
22、本书写的还算不错，里面的例子被很多公司扩展，并当作面试题，对于搞OI和ACM的人来说，里面的算法过于简单，对于一般的程序员来说，里面的内容还可以，总之能启发你进行思考。
23、有意思的编程故事
24、还不错，想研究算法的可以仔细研究下
25、面试宝典
26、短小精悍的算法书，缺点是时代稍久远，有些过时，但不失经典
27、看过1次，有收获。注意第二版与第一版的内容是不一样的:)
28、适合算法入门，很有启发性。
作者博学多识，其中引用的许多书籍也多非常有名。
中国人很悲剧的一件事情，就是一开始就啃国人出版的书，花了那么多的时间，读得却不是最好的。。
外国有很多算法书，写得都非常不错。。
29、不是算法教材，是修炼编程思维和实践能力的武器。几点感想：1，用好标准库。2，算法、代码时间空间上的预估和优化。3，经典算法、数据结构（如排序，查找，堆等）要能深入剖析。4，编码后的验证测试，利用循环不变式、函数进入退出条件、断言、边界条件等指导程序正确编写。5，要认真看习题。
30、一些通用的指导意见，思想可以借鉴，例子和程序有些比较古老，有些放到现在可能没有意义。可以一看的书，10分能给7分左右吧
31、很赞
32、大学的时候看的。比较难啊。。

33、此书当然是经典中的经典，但是中国电力出版社这一版的翻译实在太次了吧？！拿google翻译都比你强
34、一星给翻译。
35、写题
36、平时多思考吧！
37、讲解算法的一本好书
38、记得很早就已经听说过这本书，一直以为只是纯粹的算法书。曾经试着读过，觉得有点难度，后来也就不了了之。

之后读过了算法导论，最近重新尝试看这本书。发现这本薄薄的书讲的不完全是算法，而是一些实用的编程经验。怪不得书名叫编程珠玑。再次看这本书，并没有以前或者是其他人觉得那么难。相反，以前看算法导论的时候，讲的都是纯粹的算法，就算在解答数学题。而这本书通过大量的实践例子，来讲解如何去分析一个问题，运用合适的算法、数据结构来设计出解决方案。一题多解，深入浅出的方法让读者仿佛在一位经验丰富的导师一步一步的带领下，领悟算法中的美。

由于本书比较简洁，如果是刚入门的同学就先放一放，建议是学了一些算法基础后再接触，效果会更好。
39、这才是一个计算机专业学生要学的，语言有2门就差不多了，要用再学
40、翻译略差
41、能看出往日的光辉，但是内容确实陈旧了。引用 Knuth：那时候的很多想法，现在都写成 API 了。ps. 这本书讲的是高层次算法设计，跟 CLRS 并非同一类
42、很经典的一本书，适合没有压力的情况下梳理自己的思想，提升境界
43、相当不错
44、作者自己的心血，可惜我经验不足，读了还是不懂
45、简洁、精练的算法学习经典书籍
46、珠玉在前，而自出机杼！
47、基础讲的还不够，也许是作者的基础还不行吧
48、短小，精炼，没有更精妙的程序了。
49、入门书
50、应该多看几遍。 可惜还得早。
51、再也不编码了
52、找工作那段时间看的，写的挺好，比较薄，容易看，讲了一些思想细节
53、还不错 看过会有启发 但是没有那种眼前一亮的巨作感

《编程珠玑(第二版)》

精彩书评

1、如果不是因为翻译稍有一点点瑕疵，一定给评分顶级，所以如果可以读懂英文版，更加推荐了。这是一本每一页都值得细细去读，每道章后习题都值得认真去思考的书。内容不多，页数不多，但都是很有启发性的东西。如果没有时间（或者胆量:P）去读Knuth的TAOCP，不妨先读一读Programming Peals体会一下，就有兴趣接着读TAOCP啦。:D
2、我看过了 我看过了
3、就中国电力出版社这个版本，翻译真不是一般的烂。翻译真不是一般的烂。翻译真不是一般的烂。翻译真不是一般的烂。翻译真不是一般的烂。翻译真不是一般的烂。翻译真不是一般的烂。翻译真不是一般的烂。翻译真不是一般的烂。翻译真不是一般的烂。翻译真不是一般的烂。翻译真不是一般的烂。翻译真不是一般的烂。
4、不知道翻译问题，还是本来太高深，竟然看不懂。比如：21页(3.1调查程序)第三行中，“在实际问题中，共有25列，表示8个类别以及3个类似的输出页面……”这是毛呀，25列从何而来，输出页面有为何物。真是莫名其妙。既然是调查程序，这段程序的目标是什么都没有。。。（原谅我的无知，请辩驳解答）
5、本人才疏学浅！正如书名，珠玑编程中的精华部分！我必须承认有句话叫仁者见仁智者见智！我不想驳斥任何人的观点，所以不喜勿评!先说点题外话本人是个前端开发人员对C/C++这等底层语言很是崇拜，可能不如写这类底层语言的大牛小牛你们专业。书回正传，我曾google百度下这本书想真正的传达什么思想？如果真的是本算法书籍！我想书名肯定不会这么叫，我发现很多评论或者读后感都在讲这书如何如何牛X，如何如何字字珠玑！我想正在看评论的你会问“楼主，你怎么看？”。本书三个部门，1、基础 2、性能 3、应用。别着急我慢慢分析书中内容，咱先说基础部分，基础部分5章（开篇、啊哈算法、数据决定程序结构、编写正确程序、编程小事），当我第一次看目录的时候我也觉得每章之间没有任何关系，可以挑你喜欢的看，但我本人的习惯就是从头到尾，当我读完第一部分的时候，先不说里面有些算法如何精妙，我对第一部分最大的领悟就是它在教你怎么针对一个编程问题写好一个正确不冗余的程序。开篇我们可以理解就是正确沟通前提下正确的理解问题，如果一个问题你理解有一点失误可能导致的就是后续所有工作功亏一篑。在理解正确的前提下，你要考虑一个精妙的算法，‘懒’程序员往往不会先动手去编程，他们往往会想出至少两个解决方案，第三部门就是明确算法之后我们要用一种数据结构或者多种数据结构的组合来设计程序结构（数据结构决定程序结构），注意到现在我们还没有开始具体的代码实现，一切可以用伪代码勾勒在你的纸上，下一步就是写代码了（你最喜欢的一部分），第四章tell你如何写出正确的程序（具体内容例如断言等技巧），既然写完了初步程序你需要保证程序的正确性！第五章就讲了如何利用脚手架测试你的程序，书中经典的二分算法就是其中例子！第一部分到底讲什么呢？自己总结下吧！第二部分，首先我还要说我是一个前端攻城师，性能这章我觉得就是告诉你在程序正确性保证的前提下，我们要考虑如何高性能的跑起来！老实说作者当年所处的时代是个内存低，CPU差的时期！性能这块是个硬伤！在这个随处可见8G内存的时代（当然32系统的用户咱不敢恭维，当初串个机器的时候见一哥们买了两个4G内存，说回家装个XP，另外XP上的IE6容我fuck一下），CPU随处可见双核甚至4核（你电脑不是双核你都不好意思出门跟人家打招呼）。性能已经不是程序猿的主题了！加上商业时代，雇主也不会多花精力搞这个，赶紧卖出去就完事！不过你还是多看看怎么实现时间和空间双盈的算反，没准microsoft、google让你面试呢。第三部分，扯犊子扯了半天第三部门章节误导了一些高人！他们看了目录就认定这就是TM的算法书籍你看看，11章排序（这不明显是搞算法的嘛？）12章取样问题（这不是算法跟数学的巧妙结合吗？）13章搜索（这不也是算法问题嘛），等等。我想说第三部门主题是应用！它告诉你用前面思考编程方式在具体每个章节实现，用他前面教会你分析问题，定位问题，选择一个实现过程和一个好的数据结构，编写代码，测试and so on。所以我认为本书形散神不散。
6、原书很棒，虽然每一章都不长，但内容无愧于“珠玑”二字，讲述了很多技巧，方法。但遗憾的是，翻译太差，文字往往于原意南辕北辙，读起来很痛苦。
7、这本书非常好，篇幅不长，但对于思想的培育很有价值，而且给出的一些参考书都是经典。也是属于Bell Lab的光辉岁月的产物。缺点就是篇幅略短。p7 用一定的时间把小问题思考清楚可以减少大量

的时间。p68 要懂得估算，并为各种规划设置足够的安全系数。p81 dr.dobb's essential books on algorithms and Data structure
8、传说功力不强的人阅读高深的武功秘籍容易伤身甚至走火入魔。看来这本书已经逼近自己的极限。不过好消息是挺过这个过程传说功力就能上一甲子。我阅读本书的前两章是一个翻过一》退回去一》再翻过的痛苦过程，直到我把所有东西都搞懂。如同前言所说，不要急着看完它，多想想。相比某些奇技淫巧华而不实的编程难题，书中列举了许多现实中实实在在的困难需求，以及魔法一般的解决方法。这些方法在我看来是如此帅气，以至于有的我即使看了答案，还要花1天甚至更久的时间来理解（感谢电力出版社那个烂烂烂的翻译）。比如这个问题 编程珠玑（第二版）第二章 问题A给定一个包含32位整数的顺序文件，它至多包含40亿个这样的整数，并且次序是随机的。请查找一个此文件不存在的32位整数（2^32>40亿，所以必然有遗漏）。内存空间只有上百字节以及若干备用文件的磁盘空间可以使用http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15750-s04/www/HW4_sol.txt天杀的这答案翻译还翻译错了，我花了一天的时间想了一下，然后看了上面的链接，再想了半天，总算明白了。网上有网友的解释跟代码实现，但我一看，错了。作者明明白白的说明，这个解法，不需要对表进行排序。而网友给出的答案，是对表进行分割，然后分割到小文件可以放进内存以后，进行排序，这不是作者答案的意思。我自己还在学习笔记里记下：二分查找法之前需要对表进行排序。现在看了这个问题以后，完全推翻了笔记中的结论。作者之前说明，只要给定一个范围，并且能将这个范围二分，并且保证答案就在这个更小的范围里头，或者原本就不存在就可以了。实际的答案意思是（如同大学里某些牛人教授，跳过了关键解题步骤，让我这种一般人情和以堪）：算法读取所有记录，将他们分为高位为1，以及高位为0两类放到不同文件里（用低位也可以），这个过程不需要多少工作内存，几十个byte足够。ok，书作者没有说清楚的在这里。通过这次分拣，他就知道遗漏的数字在哪一堆，不需要排序。why？因为第32位为1的数字必定有2^31个，算法中必定有个计数器，做完跟2^31比较一下，比它小的话，那么遗漏的数字肯定就在这一堆。依次类推，在遗漏数字的这一堆，再二分为31位为1跟31位为0两堆，再二分……刚刚发现这个东西，絮絮叨叨写出来作为笔记，高手勿笑，谢谢观看，再见！
9、第7章的开头:"那就是 Bob Martin 介绍的"封底计算工程技术的精彩(古怪)方式. 该思想在工程学校中是标准食粮, 但对大多数从业工程师来说, 则是面包和黄油了. 不幸的是, 忽视计算的现象太常见了"看到这里, 我被 "但对大多从业工程师来说" 里面的"但"字搞糊涂了, 这里明明是并列而不是转折怎么用"但"呢?找出英文版的来对照了一下, 果然, 别人明明用的是 and:The idea is standary fare in engineering schools and is bread and butter for most practicing engineers. 我晕, 太不负责了吧.后面这句也不对,原文是: "Unfortunately, it is too often neglected in computing"原文的意思是抱怨这种思想经常被忽视, "idea"是主语.他的翻译把"忽视计算的现象"当主语, 完全牛头不对马嘴嘛.我不要求翻译到达信雅达的程度, 至少要做到不要曲解原文的意思吧.这么一个好书, 被这样粗制滥造, 痛惜啊.
10、这几天看了，C语言程序设计：现代方法 发现编程珠玑真的已经 out of date 了；这本书毕竟是上个世纪的产物了，那时候计算机的资源还十分紧缺，对时间空间都要锱铢必较 现在的情况大不相同了，计算机硬件飞速发展，如今写程序应该更注重可读性、可维护性、健壮性 这本书中竟然有把一个线性搜索拆分成若干个IF语句来提高性能的极端行为，我认为是不可取的 顺便推荐一下《C语言程序设计：现代方法》，目前最好的C教程了，中文版翻译也不错
11、打开书本看了第一章，突然被这一章所吸引住了。真正完美的将算法和编程完美的结合，第一次这编程有美的感受，或者说编程能像数学一样优雅。也告诉我们，向别人求助时一定要准确的定位问题。只有准确的定位出问题才能真正找到适合的算法。在看第一章这前我将前言看了一下，然后我也思考了一下怎样实现，没想到最后的实现结果是如此优雅。所以我决定放慢节奏，认真品读。因为才第一章，对其他章节的评价为时太早。

《编程珠玑(第二版)》

章节试读

1、《编程珠玑(第二版)》的笔记-第13页

阿

2、《编程珠玑(第二版)》的笔记-第217页

p7
开篇问题示范的普遍原则：
1，恰当的问题
2，位图数据结构
3，多轮（multiple pass）
4，时间和空间的权衡
5，简单的设计
6，程序设计阶段

p37
程序验证：
1，断言，输入、变量、输出之间的关系
断言很重要
2，迭代控制结构
证明循环的正确性：初始化、中间过程、终止都满足条件。并用数学归纳法证明中间所有步骤都满足条件。
3，函数：
进入和返回值是否满足条件和所需功能

p64
估算技巧
1，72法则
时间是y，年利率是r%，如果r×y=72，则投入的钱会翻番
2，一年有pi×10^7秒

p79
一维最大和子序列问题的启示
1，保存状态，避免重复计算
2，将信息预处理到数据结构中
3，分治算法
4，动态规划算法，能不能将子问题的最优解扩展
5，累积，在处理范围时，累积从头开始的和从而求得中间的范围和
6，下限，证明算法的下限

p91
代码优化
1，%运算开销大约是算术运算的10倍
2，如果程序的大部分时间花在访问内存上，那么企图减少计算时间是毫无用处的。
3，展开循环可以加速代码，因为减少了对循环变量的赋值
4，保存数组最末尾的元素，可以在for循环中取消测试是否达到数组尾的条件，从而加速
5，将三角函数的使用用几何计算代替

6，用宏代替函数，可以加速过程层次从而加速
7，设法减少分页和增加高速缓存命中率

p103
数据空间技术
1，稀疏数据结构
用数组实现稀疏矩阵
2，数据压缩
压缩的编码方式
3，动态空间分配
4，垃圾回收
5，压缩程序空间
（1）使用函数
（2）用解释器替换长的程序文本
6，转换成机器语言

p112
循环不变式
设计循环时，列出循环每一步时都满足的条件。

p116
排序
1，c++库sort的效率很高
2，一端扫描的简单快排在所有元素都相同时，时间复杂度n^2
3，两端扫描的快排在上述情况下仍是nlogn，但增加了交换次数
4，元素较少时采用插入排序：快排中加入if u-l <cutoff then return，再调用插入排序

p149
堆
1，堆的两个重要操作：
（1）shiftup，在最后插入元素，并不断在树的层次上移，用于增加元素
（2）shiftdown，从根将元素下移，每次与较小的child交换，用于删除元素，删除时，将最后的元素与根交换
2，用堆实现优先级队列
3，用优先级队列的插入和提取进行排序

p163
后缀数组
1，用一个数组记录字符串每个位置的地址
2，应用1：最长重复字串
将后缀数组排序，比较相邻位置的最长公共序列
3，应用2：生成随机文本
马尔可夫文本的生成，用单词级别的后缀数组。在排序的后缀数组中，随机根据前一个单词选择后一个单词。

p183
代码优化规则：
用空间换取时间：
1，扩展数据结构，增加内部信息，减少计算

2，存储预先计算好的结果
3，缓存
4，懒惰计算法：除非需要，策略避免计算某个元素

用时间换取空间：
1，压缩
2，解释程序

循环规则：
1，将代码移出循环
2，合并测试条件。高校的内部循环应尽可能减少测试条件
3，循环展开

逻辑规则：
1，利用代数恒等式，有时候逻辑表达式代价昂贵，将其改写
2，简化的单调函数，测试是否超过阈值，避免先计算再判断阈值
3，注意逻辑测试的顺序，把经常成功的放在很少成功的前面
4，预计算逻辑函数
5，消除布尔变量，通过if-else语句替代对布尔变量赋值

过程规则：
1，压缩函数层次，例如使用宏代替max函数
2，利用常用情况
3，将递归重写为迭代
4，消除尾递归
5，并发

表示规则：
1，初始化编译时间，程序执行前，尽可能初始化变量
2，利用代数恒等式，改写昂贵的代数表达式，例如用位移实现幂和%
3，消除通用子表达式
4，配对计算，如果两个表达式经常同时计算，建立新的过程将他们成对计算
5，利用单词并发，同时进行很对位上的位访问

3、《编程珠玑(第二版)》的笔记-第28页

能用小程序实现的，就不要用大程序

4、《编程珠玑(第二版)》的笔记-第81页

8.4的scanning algorithm简直是神一样的存在。。。

谁能告诉我这到底是怎么想出来的。。。

一直不崇尚太过于技巧性的东西，相反更想知道一般性的思考规律，毕竟代码的简洁和易懂才是王道。。。

好吧。。。

本来觉得给四颗星也够了，但是因为书很薄，所以加星！

5、《编程珠玑(第二版)》的笔记-第28页

几个原则：将重复性代码改写到数组中。
封装复杂的结构。
尽可能地使用高级工具。
让数据去构造程序。

6、《编程珠玑(第二版)》的笔记-第41页

整个Column4其实就是要告诉我们：

养成好的编程习惯！！！

“ understand the code at all times, and resist those foul urges to 'just change it until it works'. ”

但是懒惰是人类的天性。。。

说到底做任何事情都要搞清楚问题的本质。。。

另外C4其实是针对loop的分析。。。recursion自然是另外一大块难啃的骨头。。。

7、《编程珠玑(第二版)》的笔记-第19页

第二章言简意赅地讲述算法的概念。本章围绕三个问题展开：
1、至多包含40亿个32位整数的顺序文件，找到一个其中遗漏的32位整数。
参考算法：二分法查找，以中间值为界，数字少的一侧必有数字遗漏。
2、将n位的字符串循环左移i位。
参考算法（字符串看做数组a[n]）：
1）将temp = a[0]; a[0] = a[i]; a[i] = a[2*i]; a[2*i] = a[3*i]; 以此类推，若没有完成循环，则继续temp = a[1]; a[1] = a[i+1]; a[i+1] = a[2*i+1]; 以此类推，最终必然可以完成；
2）字符串看做ab两个部分，假设b比a长，则b分为(b^l)和(b^r)，(b^r)和a等长，则可以首先交换这两个部分，a(b^l)(b^r)变为(b^r)(b^l)a，这样a的位置正确，再交换前面的部分，则可以看做一个递归问题；（此种解法考虑a和b的长度，且递归对代码要求较高）
3）设f(x)为一个翻转函数，例：f(abc) = cba; 那么对于ab，只要f(f(a)f(b))即可；（此处让人惊叹！）

3、找出变位词集。（如“ stop ”“ tops ”“ pots ”为一个变位词集。）
参考算法：给每个词增加一个键，为排序后的词，如以上三个词对应一个键“ opst ”，键相同的词算作一个词集。
但是，若实现代码，以上叙述远远不够，还需要更精细的工作。做一名优秀的程序员，还是要多实干、多编写代码！加油~~

8、《编程珠玑(第二版)》的笔记-第9页

程序员的主要问题不一定是技术上的，更可能是心理上的；因为他试图解决一个错误的问题，所以他不能取得进步。通过打破概念上的障碍，转而解决一个更简单的问题，这样我们最终解决他的问题了。
James L.Adams所著的《Conceptual Blocking》研究了这种跳跃，通常它可以说是一种通向创造性思维的

令人愉悦的激励。Adams将概念性障碍定义为“妨碍问题解决者正确认识问题或获得解答的心理屏障”。

9、《编程珠玑(第二版)》的笔记-第167页

Tom Duff：尽可能盗用别人的代码。
库非常伟大，任何时候都尽可能使用它们。
重用，重构，天下程序一大抄，^C^V编程法

10、《编程珠玑(第二版)》的笔记-第79页

几个重要的算法设计技术：
1 保存状态，避免重新计算
2 将信息预处理到数据结构中
3 分治算法
4 扫描算法（有关数组的问题可以通过提问“如何将x[0...i-1]的解决方案扩展为x[0...i]的解决方案？”得到解决）
5 累积
6 下限

11、《编程珠玑(第二版)》的笔记-第7页

简单的设计。法国作家和设计师**说：设计师的至高境界不是他不能再往作品中添加什么东西，而是他不能再从中取走什么东西。

12、《编程珠玑(第二版)》的笔记-第7页

开篇的例子很有启发性！
程序员应当精细地定义问题，充分了解和应用问题的条件和限制。
书中的例子是十分巧妙的：对于不重复的7位电话号码磁盘排序，在主存受到大约1M限制的情况下，采取一个千万位长的字符串，每一位对应一个号码，存在该号码则标1，不存在则标0，输入完成，排序自然完成。
当然这个例子有特殊性，电话号码无重复、且每个号码没有对应附加信息。也正是利用了这些限制，才有了这样一个绝妙的方法。
它吸引我继续阅读下去！

13、《编程珠玑(第二版)》的笔记-第1页

这本书非常好，篇幅不长，但对于思想的培育很有价值，而且给出的一些参考书都是经典。

p7 用一定的时间把小问题思考清楚可以减少大量的时间。

p68 要懂得估算，并为各种规划设置足够的安全系数。

14、《编程珠玑(第二版)》的笔记-第二章 啊哈！算法

写了三个问题：
1.给定至多40亿个32位整数，如何在几百字节的内存限制下找出一个文件中不包含的32位整数。
如果内存足够的情况，可以很容易的用第一章的位图的方法，得出答案。那么在内存限制的情况下，

只能多花时间了。书中的方法是用二分法。一般的二分法要先进行排序，但是如果要排序的话，那内存就不只几百字节了，不如用原来的方法。书中方法如下：
先找到一个包含所有输入数的一个范围，比如0~2^32－1；取这个范围的中点，然后进行统计，在中点以下的数有几个，在中点以上的数有几个，对比它们本业应该有的数量，就可以确定在哪个范围内有缺数（元素重复无影响）；在那个范围下继续二分。

2.对长度为n的序列左移i个数，内存限制在几十字节。
比较容易想到的是每次左移一位，重复i次。但是还有更高效的方法。
（1）x[0]移到临时变量t中，x[i]移到x[0]中，x[2*i]移到x[i]中（下标均对n取模），……，最后当要把x[0]中的移到上一个位置时，则变为把t中的元素移到那个位置（好像没讲清楚 -_-|||）。这样就完成了一轮的移动，如果还有元素未被移动，则从x[1]开始下一轮的移动，直到所有的元素都被移动。这样就只利用了一个临时变量就完成了整个移动过程。
（2）递归方法：把待移动部分和另一部分取出一个等长的子列进行交换，如ab，a分为cd，cdb-->bdc，接下来只要再交换dc即可，这就回到最初的问题。
（3）两次转置。原始序列为ab。a，b先分别转置，再一起转置。这种方法，编码较容易。
所有的方法都是为了解决内存少的问题。

3.给定一本词典，找出所有的变位词（所包含的字母都一样的单词）。容易想到的是先倾给每一个单词进行加工，单词内先进行排序，形成单词的一个签名。接下去我的想法是用并查集，书中给的是进行单词间的排序。如果等价类比较多的话，书中的方法比较适合，如果比较少的话，则并查集会快一些。

15、《编程珠玑(第二版)》的笔记-第一章 开篇

第一章以一个问题开始：如何进行磁盘排序。这是一个程序员给作者提出的问题。经过一番交流，作者给这个程序员所要做的事情一个更好的定义：如何给n个不个重复的正整数进行排序，n<=10^7，内存限制在1M左右。内存的限制正是程序员想要进行磁盘排序的原因。但并不是非要磁盘排序才能解决他的问题。这就告诉我们：解决问题的第一步是更好地定义你的问题。
如果能在内存中读入所有的数，那就完全不需要磁盘排序了。所以问题转化为如何在1M左右的空间内存储10^7个数。这里用的是位图的表示方法。平常的数据表示是用几个bit来表示一个数，是存储单元上的内容表示数据；而位图则是用存储单元的位置表示数据。也就是说第i个位置上为1表示有i 这个数，为0则表示没有。这样就只用一个bit就表示了一个数，效率大大提高。而且当数据输入完后，就已经完成了排序。这相当于是一次桶排序。效率的提高来自于利用了数据的特性：有限域的稠集，也就是数据不重复，而且不是非常的稀疏。
书中有一句话还不错：设计师的至高境界不是他不能再往作品中添加什么东西，而是他再也不能从中再取走什么东西。

16、《编程珠玑(第二版)》的笔记-第16页

看到这我想起了《你画我猜》的作弊工具。。。

17、《编程珠玑(第二版)》的笔记-第七章 封底计算

这一章讲的不仅仅是编程。很多的东西根据现有的信息，可以自己估算出一个大概，而不是一味地去相信别人给的数据。
一个是72法则，讲的是若增长率是r%，那么过72/r 的时间，数量就翻一番。
一个是利比特法则，系统中的平均数量等于进入速率*滞留时间

《编程珠玑(第二版)》

版权说明