计算机算法：设计和分析引论

[美] 萨拉·巴斯 著

计算机算法：设计和分析引论

[美] S. 巴斯 著
朱洪 游之墨 胡美琛 译
李为镒 校

复旦大学出版社

SARA BAASE
COMPUTER ALGORITHMS:
INTRODUCTION TO DESIGN AND ANALYSIS
First Published 1978
ADDISON-WESLEY PUBLISHING COMPANY

计算机算法：设计和分析引论

复旦大学出版社出版
新华书店上海发行所发行
华东师范大学印刷厂印刷
字数 295 千字　开本 850·1168　1/32　印张11.5
1985 年 5 月第一版　1985 年 5 月第一次印刷
印数：1——25,000册

书号：13253·18　　定价：2.05元

译 者 序

近年来，国外出版的离散问题算法书纷至沓来。Sara Baase 的 "Computer Algorithms: Introduction to Design and Analysis" 一书是积作者在美国加利福尼亚大学圣地亚哥分校多年教学经验而写成的。它取材适中，与其他同类书目相比侧重于算法的分析，比较符合我国国内大学生水平。有鉴于此，译者在采用本书作为复旦大学计算机科学系教材进行教学的同时，翻译出来以飨读者。

本书的取材，如第二章的外整序，第七章近似算法的几个例子等等都是颇有特色的。对某些问题的复杂性，如求最大项、次大项问题，键比较整序问题，和对若干实际问题近似算法的近似度都讨论得相当透彻。对于比较复杂的算法，如第三章的图论算法，第五章的快速傅里叶变换算法，第六章的 **UNION-FIND** 算法均解释得相当清楚。另外对一些比较深奥的理论问题，如计算复杂性相对于计算模型的独立性，完全性问题等等，只有在严格的数学基础上方能讲清楚，但是作者却用比较妥贴的非形式叙述方式，使学生得到感性的直觉的初步观念。

本书第 0，6，7 章由朱洪同志翻译，第 1，2 章由游之墨同志翻译，第 3，4，5 章由胡美琛同志翻译。翻译过程中，对原书的不妥之处，译者作了必要的修正和注明。最后由李为鑑同志对全书进行了校对。

限于译者水平，译文中难免会有错误和不当之处，敬请读者批评指正。

序　言

本书是打算给高年级一学期课程用的教材。它包含足够的材料，以备教师后半学期可以选用若干不同的专题和算法。要求读者事先熟悉链表、栈和树等数据结构，在少数的算法中用到了递归，另外还用到少量微积分知识。

本书的目的是双重的。一方面不仅讲授在计算机应用中常常遇到的实际问题的解法，而且讲授分析各种算法的基本技巧和原理。另一方面与讲授具体内容至少一样重要的本书的另一目的是——培养读者在遇到新算法时有总是思考下述问题的习惯：这个算法有多好？是否有更好的方法？因此，这里不是提出一系列完整的凭空想象的分析好的算法，而是先讨论问题，考虑一种或几种解决该问题的途径(就像读者可能是初次遇见这个问题一样)，然后开始创造一个算法，分析它，修改或摒弃它，直至得到一个满意的结果为止。诸如：如何能使它更有效？这里适宜采用哪种数据结构？如何建立起初始的数据结构或给变元赋初值？等等这些问题在全书中将屡见不鲜。答案一般就在问题之后，但是我建议在阅读下文之前暂停一会儿，思考一下你们自己的解答。学习不是一个被动的过程。

算法分析的主要之点是分析算法的工作量（包括最坏与平均两种情形）、空间用量和一个问题的复杂度下界。我希望读者将学会去了解一个算法在不同输入时的实际性态——这就是：哪类测试数据是成功的？算法的各段落以怎样的次序执行的？栈是以怎样的模式伸长与压缩的？在中间步骤数据结构是什么模样？输入方式（例如，以不同次序输入一个图的顶点和边）到性态的？这类问题出现于某些习题中，但不在课

们，因为这需要深入到许多例子的细节中。读者应当留意这些问题。

本书所讲述到的大多数算法具有实际的用处。我已经决意不强调那些虽有好的渐近性态然而对于有实用价值尺寸的输入其性态很差的算法（尽管它们中一些具有理论价值的算法还是被提到了）。书中选定某些具体算法往往基于下列各种不同的理由：问题的重要性，展示解决同一问题的不同途径，阐明分析技巧，阐明在许多问题里都采用的技巧(例如深度优先搜索)和阐明算法和技巧的发展与改进（例如 **UNION-FIND** 程序）。(第 2.9 节外整序是应我的很多正在工作的学生的要求而写的，他们认为这对他们的工作很有用)。超高复杂度问题的频繁出现促成了第七章。在那里向读者介绍了 $\mathcal{NP}$-完全性和组合问题的近似算法。

一学期的课程可以包含下面内容：

第一章，第 1，3 和 4 节(第 2 节的算法语言作为课外阅读材料)；

第二章，第 1—5，8 节以及余下诸节之一节；

第三章，(或许略去最后一节)；

第四章；

第五章，第 1—3 节；

第七章，第 1，2 和 4 节；

如果时间许可，讲授某些略去的章节或第六章的某些节。

和正文一起，习题是本书的重要部分。读者应当全部浏览一遍，并且根据自己的时间和兴趣尽可能地多做。某些习题并未告诉你答案是什么；例如请读者去寻求一个问题的复杂度的好的下界而不是去证明一个给出的函数是下界。这是为了使问题的形式更加符合实际；解答找到后必须进行验证。

我们将用包含诸如 **IF-THEN-ELSE**, **WHILE**, **FOR** 和

[illegible]程序设计语言语句的高级语言以及用英文书写的指令

这样做的意图是使算法显得很清晰而不必担心语言

的细节。虽然在课文中我们没有给出实际的程序并进行分析，然而当数据结构和执行的细节是特别复杂或带有技巧性还或许正好是分析的关键时，这就是说不同的实现会引起算法复杂度实质性改变时，我们就要详细地进行讨论。

这本书出于若干理由专致于分析算法而不是分析程序。理由之一是因为一个好的算法即一个好的解法，是一个好的程序的必要组成部份；它是基本的东西。程序不可能比算法更好些，而且它将包含不少与方法和分析无关的细节。当然，学生一定要学会怎样去编写一个与算法一样好的程序。因此另一个不在课文中提供程序的理由是我希望学生去写那些程序。除了最后一章外，各章都包含一系列程序实习题，这些实习题给学生去实际地执行算法。某些还需要少量与执行有关的分析。我建议指导老师选择少许在课堂上讲授过的算法用学生使用过的程序语言作详细的示范和分析。

参考文献包含了小部分有关的文章和书。我不打算把本书所有材料的原始来源都包括在内，因为我觉得对于这样水平的一本书来说既非必要也不很有用。不管怎样，如果一个主要算法和章节有一个来源的话，它总被包括在内。一般说来，我试图放入足够的材料使有志向的学生将有一个进一步阅读的起点。

我向我过去几年的学生表示感谢，他们忍受了本书早期原稿中的许多谬误，排印错误及晦涩之处；我要感谢 Paul Young 和 Mike Machtey 在他们的讲授中采用本书的原稿并从一开始就鼓励本书的出版计划；我还要感谢 Donald Knuth 的书和 Richard Karp 的讲演，它们重新激发了我对计算机科学的兴趣；我特别要感谢 Jack Revelle，他不仅教会我本书的很多材料还教我怎样去学习。

圣地亚哥

一九七八年六月

S. 巴斯

目　录

0 数据结构和数学基础知识

0.1 各种记号和公式

$\|S\|$	S的基数(即S中元素的个数)，此处S是一个有限集合
$\|x\|$	x的长度(即x中字符的个数)，此处x是一个字符串
$O(f(n))$	阶至多为$f(n)$的函数；参见1.3节
$\Theta(f(n))$	阶恰好为$f(n)$的函数；参见1.3节
$\lfloor x \rfloor$	小于或等于x的最大整数，简称x的向下取整("*floor of x*")
$\lceil x \rceil$	大于或等于x的最小整数，简称x的向上取整("*Ceiling of x*")
$\log n$	$\log_2 n$
$\ln x$	x的自然对数，即以e为底的对数
$\wedge$	空指针
$\approx$	近似地等于
∞	无穷大或在讨论数据结构场或变量的值时，某个远大于上下文出现的任何数的数。
$\binom{n}{k}$	二项式系数：$\binom{n}{k}=\frac{n!}{k!(n-k)!}$

0.2 若干数学基础知识

本书中用到一系列数学概念，例如对数，概率和置换。大多

数时候只需要了解定义和这些概念的基本性质。本节之目的是提供这些数学资料的简短回顾和参考，读者应当在其他课程中已经明瞭它们中的某些内容。

对数

本书中最广泛地用到的数学工具是以2为底的对数函数。

定义 对于 $x>0$，$\log x$ 是使得 $2^y=x$ 的数 y；即 2 的 $\log x$ 乘幂为 x。

下述性质容易从定义中推出（除了性质 4 外，其他所有性质对于底不是 2 的对数函数也成立）。在所有场合，x 是任意正数和 a 是任何实数。

1. log 是严格递增函数；即如果 $x_1>x_2$，则 $\log x_1>\log x_2$。
2. log 是一对一函数；即如果 $\log x_1=\log x_2$，则 $x_1=x_2$。
3. $\log 1=0$。
4. $\log 2^a=a$。（对于其他的底 b，有 $\log x=\log_b x\log b$）
5. $\log(x_1x_2)=\log x_1+\log x_2$。
6. $\log(x^a)=a\log x$。
7. $x_1^{\log x_2}=x_2^{\log x_1}$。（为证明这个公式，只需要对等式两边取对数，然后指出它们是相等的。）

贯穿全书，我们常只用到正整数的对数，而不是任意正数的对数，并且常用到接近于对数的整数而不是其精确值。设 n 是一个正整数。如果 n 是 2 的乘幂，就是说对某个 k 有 $n=2^k$，则

$$\log n=k。$$

如果 n 不是 2 的乘幂，则存在一个整数 k 使得 $2^k<n<2^{k+1}$，而 $\lfloor\log n\rfloor$（小于等于 $\log n$ 的最大整数）是 k，以及 $\lceil\log n\rceil$（大于等于 $\log n$ 的最小整数）是 $k+1$。$\lfloor\log n\rfloor$ 和 $\lceil\log n\rceil$ 将会常常用到。读者应当验证出 $n\leqslant 2^{\lceil\log n\rceil}<2n$ 和 $n/2<2^{\lfloor\log n\rfloor}\leqslant n$。

概率

假设在给定的情况中，k 个事件 $s_1, s_2, \cdots, s_k$ 之一可能发生。对每个事件 s_i，我们都赋予一数 $p(s_i)$，称之为 s_i 的概率，它满足

1. $0 \leqslant p(s_i) \leqslant 1,\ 1 \leqslant i \leqslant k$ 且

2. $p(s_1)+p(s_2)+\cdots+p(s_k)=1$。

我们很自然地把 $p(s_i)$ 解释成预期 s_i 发生的次数，对所有情况发生总次数之比。如果 $p(s_i)=0$ 那末 s_i 是不可能发生的；如果 $p(s_i)=1$，那末 s_i 总是发生的（然而注意，定义并不要求赋予的概率对应于现实世界中任一事物）。可能最常见的解释概率意义的例子是投掷硬币和骰子。如果所考察的情况（通常称为"试验"）是投掷硬币，则硬币落在地上时，可能以"正面"朝上或者以"反面"朝上。我们令 $s_1=$"正面" 和 $s_2=$"反面"，并指定 $p(s_1)=1/2$ 和 $p(s_2)=1/2$。（如果读者争辩说硬币落地时也可能侧边竖立，则我们令 $s_3=$"侧边" 且规定 $p(s_3)=0$。）如果投掷六面骰子，存在六个可能的事件：对于 $1 \leqslant i \leqslant 6$，$s_i=$"骰子落地时，第 i 面朝上" 且 $p(s_i)=1/6$。一般说来，如果存在 k 个事件，每个事件以相同机会发生，对每个 i，我们令 $p(s_i)=1/k$。

常常需要说到若干个专门事件之一发生的概率或具有一个特殊性质的事件发生的概率。设 S 是 $\{s_1, s_2, \cdots, s_k\}$ 的一个子集。则 $p(S)=\sum_{s_i \in S} p(s_i)$。例如，当投掷骰子时，出现的面号被 3 整除的可能性是 $p(\{s_3, s_6\})=p(s_3)+p(s_6)=1/3$。

置换

粗略地说，n 个对象的置换是这些对象的重新排列。设

$$S=\{s_1, s_2, \cdots, s_n\}。$$

注意到 S 中元素是按照它们的下标排列的；即 s_1 是第一个元素，s_2 是第二个元素，等等。S 的一个置换是一个从集合 $\{1, 2, \cdots, n\}$

到它自身上的一对一函数π。我们想象π是重新安排S的元素：移动第i个元素s_i到第$\pi(i)$位置。我们简单地用列出值$(\pi(1), \pi(2), \cdots, \pi(n))$来描述$\pi$。例如，对于$n=5$, $\pi=(4, 3, 1, 5, 2)$重新排列S的元素如下：s_3, s_5, s_2, s_1, s_4。

以特定次序$s_1, s_2, \cdots, s_n$给出的具有n个元素的集合的置换个数是$n!$。为了理解这一结论，观察到第一个元素可以移至n个位置中任一个；在它位置确定后，第二个元素可以移至余下的$n-1$个位置中任何一个；第三个元素可以移至余下的$n-2$个位置中任何一个，等等。所以可能的重新排列总数是

$$n\times(n-1)\times(n-2)\times\cdots\times 2\times 1=n!。$$

若干和式

在分析算法时，有几个和式常常出现。在这里列出其中一些公式，并附有简单的可能帮助读者记忆的提示。

1.
$$\sum_{i=1}^{n} i=\frac{n(n+1)}{2}$$

怎样来记住它呢？写出整数1至n，第一个整数1和最后一个整数n配成一对；第二个整数2和倒数第二个整数$n-1$配成一对，等等。注意到每对整数的和是$n+1$。如果n是偶数则共有$n/2$对；因此和是$\frac{n}{2}(n+1)$。如果n是奇数，有$(n-1)/2$对，剩下中间一个数$(n+1)/2$；于是其和是

$$\frac{n-1}{2}(n+1)+\frac{n+1}{2}=\frac{n}{2}(n+1)$$

2.
$$\sum_{i=0}^{k} 2^i=2^{k+1}-1$$

怎样记住它呢？考虑到每项2^i是一个二进制数中的一位，

$$\sum_{i=0}^{k} 2^i=\underbrace{1\ 1\cdots 1}_{k+1\text{个}1}$$

如果将这个数加1，结果是