计算机科学理论

程序设计途径

〔英〕J. M. 布雷迪 著

陆佑珊 陈慧清 译

陈火旺 校

科学出版社

1988

内 容 简 介

本书是从程序设计角度讨论计算机科学理论的一本专著，简洁地介绍了计算机科学理论的主要问题．全书由两大部分组成，共分八章．第一部分是元计算机科学，包括：抽象机方法，计算机科学中局限性的分析，函数或程序设计方法，图灵论题的论证．第二部分是计算机科学理论，包括：McCarthy 的开拓性研究成果，程序可靠性的测试法、证明法，关于形式语义的研究．书后的附录介绍了数学和程序设计的基础知识．

本书可作为计算机专业的学生，特别是研究生的教材，也可作为计算机专业科研人员的自学参考书．

J. M. Brady

THE THEORY OF COMPUTER SCIENCE

A Programming Approach

Chapman and Hall, 1977

计 算 机 科 学 理 论

程序设计途径

〔英〕J. M. 布雷迪 著

陆佑珊 陈慧清 译

陈火旺 校

责任编辑 那莉莉

科学出版社出版

北京朝阳门内大街 137 号

中国科学院印刷厂印刷

新华书店北京发行所发行 各地新华书店经售

1988 年 4 月第 一 版 开本：787×1092 1/32

1988 年 4 月第一次印刷 印张：11 3/8

印数：0001—5,200 字数：258,000

ISBN 7-03-000245-8/TP・15

定价：2.95 元

译 者 的 话

在我国，已经出版了不少关于计算机科学的书籍，它们从不同的基点，在不同的层次上介绍了计算机科学的各个分支，如程序设计方法、程序设计语言、编译技术、操作系统、数据库、网络，直至并行处理，等等．然而，在这些计算机科学的书丛之中，却缺乏一本较系统、全面地阐述计算机科学基础理论的书，这正是我们翻译本书的目的．

"任何一种理论都能够增进我们对某一学科领域的了解，使我们对该领域的知识系统化．但是，建立理论的一个更为直接的原因则在于：理论常常能够导致实践方面的重大进展．计算机科学理论亦然．"" 计算理论对于计算实践是有生命力的、切合实际的和刻不容缓的．"我们深信，计算机科学界的知名学者——J. M. Brady 教授这一专著的出版，不仅会加强我国计算机科学的基础研究，而且将促进我国计算机的普及与应用．

全书由两大部分组成．元计算机科学部分(序言，第一至第五章）由陆佑珊翻译，计算机科学理论部分（第六至第八章，附录)由陈慧清翻译，陈火旺教授对译稿进行了审校．

译文初稿于 1981 年完成．陈火旺教授、王兵山副教授等曾用此稿作为研究生教材，收到了良好的教学效果．在本书的翻译过程中，王玉、罗朝辉、陈海波、马庆鸣、吴兵同志给予了大力的支持与帮助，在此谨致谢意．

由于我们水平有限，文中的错误在所难免，殷切希望读者批评指正．

陆佑珊　陈慧清

序　言

二十世纪四十年代，第一台电子数字计算机问世．自那时起在三十余年的时间里，计算机的应用已经相当普及，当今的西方社会可能很少有人生活不受计算机的影响．绝大多数公司、政府部门和高等教育机构均有自己的计算中心，为数不少且日益增多的中等学校亦然．

尽管我们难以臆测再过三十年计算机会被用来做些什么事，但是，我们能够确切地预言：诸如自动操纵的交通管理信号灯、装配线、用于星际探索的自控车、自动时刻表和路由选择服务，以及象海床那样不适合人类的环境中自动操作的机器等将会出现．事实上，其中多数已具雏型．同样地，计算机成本的骤降意味着相当一部分人将拥有一台或数台能力足以和目前大型机相匹敌的计算机，就犹如目前的电视机达到的那种普及程度．这种状况最终会对信息传递、生活方式和社会结构产生怎样的影响，我们是无法估量的．

迄今为止，计算机的开发本质上具有工程或技术的特点．在工业和政府的大力扶持下，新型机器设计和元器件、新型程序设计语言、结构和应用均以不可思议的速度得到了发展，近来似乎仍无减缓之势．

其间，已经采用了标准的科学过程．越来越多的著者筛选了所提出的概念并加以抽象，研制出了用来分析计算现象、解释观察到的规律性和建议新的发展方向的模型．顾名思义，本书介绍的是使用计算机的理论．我希望留给读者的一个印象是，计算理论对于计算实践是有生命力的、切合实际的和刻

不容缓的．第一章说明该理论如何在本书第二部分所列举的研究工作中得到检验．这些工作包括 John McCarthy 的开创性研究(第六章)、软件可靠性(第七章)以及程序设计语言的语义(第八章)等．一本入门书不应当包罗万象地论述某个领域，而应按明确陈述的准则选材．有限自动机和复杂性理论这两个题目本应收入第二部分，不过最终还是割爱了．抛弃前一个题目是因为有几本优秀的教科书已经论及；排除后一个题目的理由在于：我感到目前的工作受到更多启迪的是本书第一部分的思想，而非计算实践．

在计算机科学的发展史上，一个饶有兴味的怪事是，研究"可计算的"精确定义早于实际建造计算机！这一事实归因于数学家在研究数论与逻辑时也需要"能行过程"(如今我们所谓的程序)这个概念，并力图使这些直观精确化．本书的第一部分讨论我们称之为"元"计算机科学的理论．我们会发现，可计算性的思想与某些最古老的哲学悖论和数学原理有着紧密联系，尤其跟二十世纪最著名的智力进步之一——Gödel 定理密切相关．该定理实质上阐明，用诸如二阶谓词逻辑那样的形式语言能够证明的东西是有限的．

然而，从事计算机科学研究的激情本身并非撰写本书的充足理由！现在出版的书籍那么多，每一位著者都应该充分地证明增添自己的贡献是必不可少的．我以为，有下述三个主要的理由撰写本书，以区别于大多数涉及类似内容的其他著作．

第一，我确信计算理论必须直接和学生在计算实践方面的直觉相联系．这些直觉包括程序设计、数据结构、机器设计，等等．据此，例如在第三章里，我们把递归函数论中的定理看作程序设计问题，进而披露，迄今为这些定理的证明开发出来的程序设计技巧有限得令人瞠目．类似地，第五章把证

明图灵的形式系统与广义递归函数等价性的问题视为计算机问题，从而揭示出广义递归函数的一个不足之处.

这种方法，尤其对于第一部分的材料，应该与更常见的抽象数学处理加以比较，后者无视计算机科学的直观性，起因于上面提到的历史“偶然”. Marvin Minsky (1967) 基于机器的考虑是这种非难的一个例外. 其实可以看出，本书和 Minsky (1967) 的观点，犹如程序设计和计算机的硬件设计，有着同样的关系. 这种看法最终解释了本书书名的含义.

采用程序设计方法研究计算理论产生一个值得注意的结果：我否认阅读本书的主要前提是熟悉诸如集合论、代数结构和形式逻辑那样的现代纯粹数学概念. 当然，熟悉这些概念易于消化部分内容，但是我的经验表明，附录 A 包含足够的数学背景. 计算经验才是主要的前提，全书渗透着计算直观和实践.

本书的第二个显著特点与第一点密切相关. 这些年来，我发觉在我们的计算机科学同行里，关于计算机科学理论与元理论之间的关系有所混淆. 此外，在第一章里我将阐明，理论与元理论所关心的事情是相当不同的，因此，那种批评图灵机与 von Neumann 计算机全然不一的说法，简直文不对题.

第三，在本书的第二部分，我竭力校正一般作为理论（通常是第一部分的元理论）讲授的内容与计算机科学理论家所关心的问题之间的严重失调. 很少有计算机科学家积极地工作在递归函数领域，他们从事的是有关程序设计语言的形式语义、程序正确性等方面的研究. 不避浅陋，我设法梳理出支持语义和正确性模式的某些假定，使读者思索有关模式可能有的短处，并对计算机科学至今仍处于原始状态保持清醒的头脑.

本书概述的计算理论研究方法是作者在 Essex 大学讲授入门教程的五年内提出来的. 粗略地说,它是作为本科(二年级)学生两个学期和研究生一个学期的课程的. 讲授这门课的同时还要完成两项作业. 第一项要求学生用一种合适的高级程序设计语言实现一台通用图灵机(参阅习题 2.24); 第二项要求学生阅读六篇关于诸如 λ-演算或复杂性层次等课题的研究论文后, 写一篇短论表述他们的理解程度和评价. 我发现,后一项对于那些使人信服的理科学生特别有益,他们作为受教育者,往往过多地被授予形形色色的绝对真理,这种争论、赞同和价值评估对于科学是特有的.

最后,我诚挚地感谢在撰写本书时受到的启示、批评和鼓励. 首先是 John McCarthy, Peter Landin, Rod Burstall, Christopher Strachey, Tony Hoare, John Reynolds, Robin Milner, David Park, David Luckham, Peter Lauer 和 Bob Floyd 等作者, 他们的思想激发我思考计算机科学理论;其次感谢我在 Essex 大学的同事 Richard Bornat, Pat Hays, Ray Turner, Bernard Sufrin, John Laski, Tony Brooker, Lockwood Morris, Cliff Lloyd 和 Barry Cornelius. Ray Turner, Tony Brooker 和 Naomi Brady 阅读了数遍初稿, 提出了许多改进意见. Pam Short 承担了大量的打字任务. 我尤其感谢来自家庭的支持.

J. M. Brady

1976. 10

目　录

序言 …… v

第一章　概述 …… 1

1.1　引言 …… 1

1.2　第一部分：元计算机科学 …… 4

1.3　第二部分：关于计算机科学理论 …… 11

第一部分　元计算机科学

第二章　抽象机方法 …… 22

2.1　引言 …… 22

2.2　基本机和有限状态时序机 …… 23

2.3　输入序列；状态 …… 28

2.4　FSM 的局限性 …… 34

2.5　图灵机 …… 36

2.6　通用图灵机 …… 47

2.7　Shannon 对 TM 复杂性的度量 …… 50

2.8　其他类似计算机的抽象机 …… 56

第三章　计算机科学的局限性 …… 63

3.1　引言 …… 63

3.2　不可计算的函数 …… 66

3.3　可枚举性与可判定性 …… 70

3.4　不可解性成果概述 …… 86

第四章　函数或程序设计式方法 …… 116

4.1　引言 …… 116

4.2　一般递归函数 …… 121

4.3　Ackermann 函数以及对复杂性的再考察 …… 137

第五章　支持图灵论题的证据……147
5.1　引言……147
5.2　图灵机的一个 GRF 模拟程序……147
5.3　GRF 的 TM 编译程序……157
5.4　作为数据结构的数……162

第二部分　关于计算机科学理论

第六章　McCarthy 的开拓性研究……168
6.1　引言……168
6.2　McCarthy 的形式系统……170
6.3　S-表达式……181
6.4　形式系统的应用——程序等价性……190
6.5　形式系统的应用——程序设计语言的语义……206
第七章　程序的可靠性……215
7.1　引言……215
7.2　程序测试……220
7.3　证明程序正确性的归纳断言技术……230
7.4　Zohar Manna 的基础研究……256
7.5　述评……264
第八章　语义问题……269
8.1　引言……269
8.2　λ-演算……276
8.3　操作语义：强调机器的作用……285
8.4　数学语义：不强调机器的作用……298
附录A　数学预备知识……312
附录B　程序设计说明……332
参考文献……337
著者索引……346
汉英名词索引……348

第一章 概 述

1.1 引 言

近二十年来，计算机科学已经作为一门独立的学科出现．在此期间，人们建立了诸如进程、数据结构和分时等许多新概念．基本的计算活动是用**程序设计语言**编写**程序**，这些程序在**计算机**上执行时将按解题的要求对数据进行处理．犹如古典力学试图解释物体的运动一样，计算理论也试图建立一种能够解释所有这些现象的理论．

任何一门科学理论，主要的是研究如何用抽象数学表示该理论所涉及的客观对象．表示的方法应该便于用方程或定律的形式揭示数学关系．例如，在古典力学中，客观对象包括：距离、速度、力和加速度．距离通常用时间的函数来表示，速度则作为距离函数的导数．基本关系是牛顿的运动定律和他的万有引力定律．一个理论所揭示的关系可以用客观对象的术语来解释，以预测或阐明人们在现实世界中所观察到的现象；反过来，人们可以扩大理论的应用范围，对现实世界作进一步的观察．

任何一种理论都能够增进我们对某一学科领域的了解，使我们对该领域的知识系统化．但是，建立理论的一个更直接的原因则在于：理论常常能够导致实践方面的重大进展．计算机科学理论亦然．1.3 节介绍本书的第二部分，描述当代计算机科学实践中的若干不足之处．计算理论可望在这一方面作出重要贡献．

每一个理论都存在着一个与之相应的元理论（希腊语："元"，即"关于"），元理论分析这个理论本身．例如，对这个理论中的基本概念给予精确的定义，并研究这个理论的局限性．数学的元理论叫做元数学，它把集合、证明和定理这些基本概念形式化，并且试图揭示它们的局限性．元数学的一个较为重要的结果，即谓词逻辑的不可判定性，意味着不能写出这样单一的程序，即对所有逻辑上合法的语句，它能判定哪些是定理，哪些不是．

一般而言，理论确立有用的**正面**结果，而元理论多半由**反面性**的结果组成，它限定这种理论的研究范围．下一节所介绍的本书的第一部分是元计算机科学的一个导论，旨在对"可计算性"下精确的定义，并建立诸如图灵机停机问题的不可解性这样的元结果．不可解性的含意是，我们不可能写出一个单独的程序 P 来判定输入给它的任何一个程序 Q（如同把程序作为数据提交给编译程序那样）能否在有限的时间内终止它的计算．此外，本书第二部分将介绍计算机科学家已经发现的正面结果的一些研究领域．

计算机科学理论与其元理论的目的不同，这反映在它们所提出的对可计算性的不同处理方法上．我们在本书第一部分将要见到的系统——图灵机和一般递归函数能够使元计算机科学的结果易于证明，但使用很不方便．相反地，象我们将在第二部分（第六章）见到的 McCarthy 系统虽则在证明结果时要方便得多，也实用得多，但在证明定理时却困难得多．如果我们因为第一部分的形式化不实用，或者与现代计算机或程序设计语言不一致而对它们横加指责（如同计算机科学界许多人所做的那样），那是不恰当的，因为那不是形式化的目的．

事实上，作为元计算机科学理论基础的"递归函数论"的

建立，比实际建造计算机要早许多年！例如，我们将在第二章中遇到的图灵机建于 1936 年．之所以出现这种表面上荒谬的情况，主要是因为(元)数学家很早就研究过诸如“能行过程”或“算法”(或我们现在所说的“程序”) 等问题．1900 年在第二次国际数学家会议上，本世纪最伟大的数学家之一 David Hilbert，(1862 年—1943 年) 作了富于启发性的著名演讲，提出了二十三个问题，有的至今仍未得到解决．其中第十个问题是：能否编制一个程序，以判定任意一个有限多元的整系数多项式是否拥有整数根．1971 年 Matyasevich 证明不可能存在这样一个程序．

采用计算机科学的观点来处理第一部分的内容是本书的一个显著特点．这一部分内容通常是作为数学的组成部分，但与计算机科学多少有点关系，我们却认为它是**计算机科学的元理论**．因此，我们将始终注意把所引入的概念与计算机科学联系起来．有关程序设计等方面的问题，我们将依靠直觉解决，而例题则从计算机科学中提取．这一点特别适用于第三章，因为其中讨论的是**不**可计算函数或不可解问题．典型的例题取自 Luckham, Park 和 Paterson (1970) 的著作及形式文法理论．这些例题与比较常见的数学或逻辑的例题(这方面的例题因限于篇幅，此处不再列举)不同，它们是在回答计算机科学提出的问题的过程中被发现的．

我们特别要推荐一种思想，它是处理第三、第四、第五各章材料的依据．最好通过一个例子来说明．在第三章中我们将遇到各种集合 S 是否可枚举的问题．定义可枚举的一种方法是借助于从自然数集 N 到 S 上的可计算函数 $f:N \to S$．还有一种方法，尽管它不太形式化，但对于计算机科学家在直观上却更富有吸引力．这就是，如果存在一个程序能够连续地打印 S 的元素，且 S 的每一个元素从打印过程的开始能够在有

限的时间之内被列出，则称 S 是可枚举的. 因此，我们"证明"可枚举性通常就是设计一个执行枚举过程的程序，从而得到**程序作为构造性证明**这样一个概念. 在第四、第五两章中，这一概念得到了极其充分的发挥，我们还将阐述如何运用程序设计方法洞察那些传统上被认为是困难的证明. 我们试图使读者相信，应用广泛的程序设计技巧能够十分有效地研究可计算性理论. Hoare 和 Allison (1972) 的主张似乎也相差无几，我们还将多次提到他们的论文.

总之，确定计算机科学的范围与价值的将是可计算性理论的贫富程度. 作者还竭力使读者相信，理论并不象一位可尊敬的计算机科学的教授曾经声称的那样，"如锯木屑那样干巴巴"，而是生动的，充满着活力和引人入胜的.

引言这一章的其他篇幅，将比较详细地综述第一部分(第二至第五章)和第二部分(第六至第八章)所涉及的内容.

1.2 第一部分：元计算机科学

在前一节中我们看到，元计算机科学的两个主要目的是：(1) 给"可计算性"下精确的定义；(2) 根据这个定义揭示可计算性理论的局限性(倘若存在这种局限性的话)，从而确定计算机科学的论题.

1.2.1 "可计算性"的定义

计算机科学家通过"算法"、"能行过程"、"程序"、"过程"、"子程序"这些范畴，对"可计算性"概念有了深刻的直观的了解. 然而，尽管我们确认可计算性理论的描述应当与计算机科学系学生的直观相联系，但是仍需提请注意的是：对计算机科学理论最基本的部分的理解，不可过分地依赖于直观. 假

如集合论和 Russell 悖论的经验(参阅附录A，A.5.2 节)还不足以使读者相信这一点的话，本节的最后部分还将评述直观的局限性.

对于计算机科学家来说，给“可计算性”下定义，最清楚的说法是：凡可用某种程序设计语言 L 描述的东西就确实是可计算的，其中 L 可以选择 FORTRAN, ALGOL 60, 等等.元计算机科学必定会否决这种提议，因为正如我们在前一节所看到的，元理论系统的特点在于它们自身应该非常简单，以便能够比较容易地给出元定理的证明. 然而，这并不排斥下述可能性，即我们可以吸取这种提议中有益的成分作为计算机科学的组成部分. 但是，这样做有个问题，即对于任何一个程序，我们必须能够准确地推断它计算的是什么东西. 然而，这个问题，即所谓语义问题决非轻而易举的(参阅第八章).

正象我们在前一节中已经看到的，至少从 1900 年以来，数学家和逻辑学家就试图给“可计算性”下一个精确的定义. 这种努力在 1936 年达到了顶峰，那时英国的 Turing 和美国的 Church 同时(独立地)发表了他们所推荐的定义，今天皆作为正确的定义被接受下来.

Church 引入 λ-演算表示可计算函数的标准方法（见 6.1 节和 8.2 节），用于证明如前所述的谓词逻辑的不可判定性. 如今，Church 的工作已经成为过程描述方法的基础，这种方法用于诸如 LISP (McCarthy 等，1962) 和 ALGOL 68 (van Wijngaarden 等，1969) 那样不同的程序设计语言；他的方法还用于确立计算结构的意义的形式定义(参阅第八章). 由此可以看出 Church 的工作与计算机科学的关系.

图灵试图通过精确地描述他认为的“直观概念”给出“可计算性”的形式定义. 他在一篇著名的论文里提出一类直观而合理的“抽象机”，这种抽象机即我们今天所称的图灵机(后

面也缩写为 TM)．他的论文的要点在于捍卫下述主张：

（图灵论题）通常能够称作算法的过程，恰好是可以在图灵机上执行的过程．

必须明白，图灵论题既不能证明也不能否证．这个论题本身体现着计算机科学的一条基本原理或定律．对于它的合理性，可以讨论，但无法加以证明．（牛顿运动"定律"的情况与此类似．直至更为精确的爱因斯坦相对论问世之后，人们才对这些定律产生疑义．）

现在我们介绍两种赞同和两种反对图灵论题的意见．

赞同意见之一：图灵研究可计算性概念的直观方法是难以挑剔的．第二章介绍的是另一种直观的研究方法．

赞同意见之二：自 1936 年以来，许多伟大的逻辑学家、数学家和计算机科学家构造了"可计算性"的另外一些定义．但是，可计算函数的集合在各种情况下都相同，从这个意义上说，他们最终获得的结论是**相同**的．这种一致性表明图灵确实发现了正确的概念．

反对意见之一：图灵的系统不够丰富．图灵机太简单了，也许人们认为"自然"可以计算的函数，图灵机却无法进行计算．（应该说，现在很少有人会赞同这种论点了，但在 1936 年，情形可不是这样的．）

反对意见之二：图灵的系统过于丰富了．存在着**理论上**使用图灵机能够计算的函数，而**不是现实的**可计算函数．比方说，如果用任意一台图灵机来计算一个函数（即使是每微秒执行一次运算），要算出答案需十倍于宇宙生命的时间，那么把这样一个函数叫做"可计算的"当然不大实际．

总的说来，尽管许多人在考虑第二种反对意见时有些疑虑，然而图灵的主张在今天还是被接受了．不过，这种意见倒成为研究程序"复杂性"的一个推动力（参阅 1.3 节）．

概括地说，已经有两种构思“可计算性”定义的主要方法，即所谓的“抽象机方法”和“函数”或“程序设计式”方法.

前一种方法包括 Turimg (1936), Wang (1957), Shepherdson 和 Sturgis (1963), Elgot 和 Robinson (1964) 以及 Minsky (1967) 的工作，这种方法旨在建立执行机构(处理机)的某些数学模型，使得计算可以在这个机构上“运行”. 于是，所谓“可计算”乃指在一台抽象机上的可计算.抽象机方法将在第二章里说明.第二章包括：图灵机模型的直观描述、“通用机”、Shannon 的机器复杂性概念. 这一章的结尾还概述一些其他的抽象机方法.

函数或程序设计式方法包括 Church (1941), Herbrand-Gödel-Kleene (参阅 Hermes 1965, 第 5 章), Hilbert-Kleene (一般递归函数), Kleene (1936) 以及 McCarthy (1963) 的工作，这种方法旨在建立“进行程序设计”的函数的数学系统. 我们将在第四章中介绍一般递归函数和 Grzegroczyk (1953) 的复杂性概念时说明这一点.

支持图灵论题的第二个论据（“一致性”论据）有一个实证，即当一个函数按照图灵的定义是可计算的时候，它必定属于一般递归函数 (GRF). 因为我们正在倡导把 GRF 看作程序设计语言，所以我们应该问一问，**按照程序设计的术语，**什么相当于 GRF 和图灵机的等价性. 我们一方面有一个语言的方法 GRF，另一方面又有一个机器的方法 TM. 我们已经证明，对应于每个 GRF 存在一个 TM，反之亦然. 很明显，我们获得一对程序：其一是 GRF 中的 TM “模拟程序”，其二是从 GRF 到 TM 的“编译程序”. 第五章描述这样一对程序的主要特性. GRF 系统是元计算机科学的一部分，第五章还特别揭示了这种程序设计式方法如何显露出它的不足之处. 我们知道，把等价问题视作编译问题的一例是

相当困难的，难怪在1936年人们不把 Church 和 Turing 的概念看作“显然”是等价的，这种看法有助于我们理解其缘由.（我感谢 George Barnard 教授提供这一段历史资料.）

1.2.2 可计算性的局限性

在精确地定义了可计算的概念之后，我们也许会问：是所有的函数都可计算，抑或能够证明存在着不可计算的“良定义”函数？换句话说，是否存在良定义的问题，我们能够证明不存在可能求解它的程序. 请注意，这里所谓的“可能”，乃指“理论上的可能”，而非“现有资源所决定的财政上的可能”，或者“程序设计技巧现状所决定的可能”. 第三章专门探讨这个问题，并且正如前一节指出的，我们将看到那样的问题确实存在着，我们能够证明它们没有算法解. 我们还将看到，关于我们能够编制程序求解的问题在理论上的局限性是极其微妙的；诚然如 Minsky 所指出的（1967，p.7）：“没有理由假设机器有任何人类所不具有的局限性.”为了获得对可计算性的局限性的感性认识，我们考虑计算机科学中导致不可解问题的两个专题，即程序等价性和形式文法等价性.

1.2.3 我们直觉的局限性

我们再论述两个问题来结束这一节，着重说明为什么需要一个精确的“可计算的”概念，并以此为基础建立计算机科学的理论. Hoare 和 Allison（1972）以另一种形式提出了这个基本课题.

1. 描述

一个算法可以看作是对一个执行机构的每步动作的无二义的描述. 归根结底，一个算法是一种描述，因而我们有理由