可计算性和计算复杂性

JISUANXING HE JISUAN FUZAXING

KE

朱一清　编著

赵致琢　审校

国防工业出版社
National Defense Industry Press

可计算性和计算复杂性

朱一清　编著
赵致琢　审校

国防工业出版社
·北京·

内 容 简 介

本书深入浅出地介绍了研究可计算性的四个主要模型以及四个模型彼此之间的关系;介绍了计算复杂性的基本概念和重要的研究方法与一些研究成果。内容涉及递归函数、图灵机、λ演算、马尔可夫算法、计算复杂度的分类、NP完全理论、非一致复杂性等。分述于十章,书中附有习题。

本书可作为广大有志于突破计算复杂性研究僵局——"P=NP?"的科技工作者,计算机科学和元计算机科学工作者,数学和元数学工作者以及大专院校的教师和学生的入门书、教材和参考书,亦可作为计算机基础理论的参考书。

图书在版编目(CIP)数据

可计算性和计算复杂性/朱一清编著.—北京:国防工业出版社,2006.4

ISBN 7-118-04329-X

Ⅰ.可… Ⅱ.朱… Ⅲ.①可计算性-高等学校-教材②计算复杂性-高等学校-教材 Ⅳ.TP301

中国版本图书馆CIP数据核字(2005)第161045号

※

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)

天利华印刷装订有限公司印刷

新华书店经售

*

开本710×960 1/16 印张10¼ 字数183千字

2006年4月第1版第1次印刷 印数1—2500册 定价18.00元

(本书如有印装错误,我社负责调换)

国防书店:(010)68428422 发行邮购:(010)68414474

发行传真:(010)68411535 发行业务:(010)68472764

前　言

20世纪70年代，有关数字计算机的研究逐渐形成了一门独立的理论——计算机科学理论，为研究这门理论又同时诞生了它的元理论——元计算机科学。

计算机科学，如大家所见到的，它研究怎样去“做”计算机，研究计算机能够“做什么”；元计算机科学则研究计算机科学本身，研究计算机科学这门理论的局限性，研究计算机“不能做什么”。

本书阐述的“可计算性和计算复杂性”理论从内容来看，它不研究怎样进行计算（这由算法设计、程序设计等课程去解决），而研究计算机在资源受限时，计算机充其量能做什么，换句话说，超过某个界限，计算机“无法做了”。

因此“可计算性和计算复杂性”是关于计算理论的元理论。当然，“元理论”包含“理论”，甚至元理论和理论使用同样的语言，讨论“相同”的问题，以致人们“无法”区分它们了。但只要注意到它们研究的层次是不同的就容易知道你的研究是关于理论的还是元理论的了。例如，冯·诺伊曼（Von Neumann）机是计算机科学家和工程师工作的对象，而图灵（Turing）机是元计算机科学家使用的工具。

“可计算性和计算复杂性”理论要说明的问题涉及计算机的硬件和软件的数学特征，因此该理论本质是数学的，也就是说它可用纸笔写下的符号和式子来表达，偶尔借助于实验。伽利略（Galilei）说过，宇宙是用数学这支笔写就的，也许这也是我们的课程离不开数学的原因。数学是一个为攀登各种山峰而安扎的大本营，每个有志于攀登险峰的青年要在这个大本营里多多练习，多做准备。

本书的写作自始至终得到厦门大学计算机系教授赵致琢博士的关心，并由他审阅全书。该书的大部分内容在暑假全国计算机科学与技术高级研讨班二期和三期介绍过，并得到贵州大学计算机系陈孝威教授和陈笑蓉教授的指教，许多章节在贵州大学完成，最后由国防工业出版社出版。

对所有帮助我写作和出版的同仁表示衷心的感谢。

朱一清

目　录

第一章　概　论

可计算性和计算复杂性是两个紧密联系着的研究课题，可以作为一个独立的整体在计算机领域中被研究，也可分为两个专题研究。

人们提出各种要计算的问题总是会问：该问题可计算吗？如果可计算，则它是容易被计算的还是较难被计算的？前一个问题由那些从事可计算性研究的人们试图解决；而后一个则由专门研究计算复杂性的人来回答。

由此产生一个更为基础的问题："某问题可计算"其确切含意是什么？亦即，"可计算性"定义是什么？

从资料可以看到，至少从 1900 年开始，数学家和逻辑学家就尝试给"可计算性"下定义了。1936 年，图灵(Turing)和丘奇(Church)同时发表了他们推荐的定义。今天，这两个定义皆作为正确的定义被接受下来了。

随着研究的展开，许多分属数学、元数学、计算机科学、元计算科学及利用各种不同模型的种种定义出现了。后来证明有关可计算性的若干定义彼此是等价的。

本章第一节和第二节中介绍可计算性的各种定义以及与可计算性的定义有关的概念。

第一节　相关定义

数字计算机所解决的"问题"指那种具有共性的"单个问题"的集合，并假定由图灵机或冯·诺伊曼机来处理的那种"问题"可由固定的方法转化为对数论函数的计算。在这样的前提下我们仅讨论数论函数。

定义 1　设 X, Y 是自然数集合 $\mathbf{N}$ 的子集，f 是函数，$f: X \to Y$，则称 f 是一个数论函数。

常见的数论函数有下面三种。

(1) $f: X \to \mathbf{N}$，$X \subseteq \mathbf{N}$；

$f: X_1 \times \cdots \times X_n \to \mathbf{N}, X_i \subseteq \mathbf{N}$，$i = 1, \cdots, n$。

(2) $f: X \to \{0,1\}$，$X \subseteq \mathbf{N}$；

$f: X_1 \times \cdots \times X_n \to \{0,1\}, X_i \subseteq \mathbf{N}, i = 1, \cdots, n$。

(3) $f: \{0,1\} \to \{0,1\}$,即真值函数;

$f: \{0,1\} \times \{0,1\} \to \{0,1\}$。

这种函数推广到谓词时得到数论谓词。关于谓词我们明确它的定义。

设 $P(x_1, \cdots, x_n)$是 n 元谓词,其中

$$x_1, \cdots, x_n \in \boldsymbol{I}$$

$\boldsymbol{I}$ 是个体域,如

$$P(x_1, \cdots, x_n) \Leftrightarrow \mathrm{T} \text{ 或 } P(x_1, \cdots, x_n) \Leftrightarrow \mathrm{F}$$

即

$$P: X_1 \times \cdots \times X_n \to \{0,1\}$$

其中 $X_i \subseteq \mathbf{N}, i = 0, \cdots, n$;0,1 分别表示"真"和"假",则 P 是数论谓词。

定义 2 谓词是以下(1),(2),(3),(4),(5),(6)的总称。

(1) **谓词** 刻画个体、命题所具有的性质和关系的词,称为谓词。

(2) **谓词变元** 以谓词为其值的变元。

(3) **谓词简空式** 当明确表示谓词联系的个体或命题的数目及位置时,用表示该谓词的符号,辅以括号、逗号和空位得到的式子,称为谓词简空式,记为 $H(,\cdots,)$或 $H(e_1, \cdots, e_n)$,$e_1, \cdots, e_n$ 表示 n 个空位(empty)。

(4) **谓词命名式** 在谓词简空式的空位处填上个体变元或命题变元的填充式,称为谓词命名式。

(5) **谓词填式** 指在谓词简空式的空位处填上个体或命题的填充式。

(6) ***n* 元谓词** 有 n 个个体变元的谓词命名式,称为 n 元谓词。

本书还涉及到量词、算子和摹状词。

定义 3

(1) **量词** 所谓量词是指把谓词变成另一个谓词的约束词。常见的量词:存在量词、全称量词、唯一性量词。

(2) **算子** 指把函数变成另一个函数的约束词。

(3) **摹状词** 将谓词变成函数的约束词,称为摹状词。

常见的摹状词:求最小根、求最大根、求唯一根。

总结一下上面的定义。

函数:数→数;

算子:函数→函数;

量词:谓词→谓词;

摹状词:谓词→函数;
数论函数:自然数→自然数;
数论谓词:自然数→{真,假}。
尚未见到讨论"函数→谓词"的文章。

第二节　可计算性

可计算性的最初定义由图灵和丘奇给出。

定义 1(图灵论点)　可计算的函数和可用图灵机计算的函数等同。

图灵提出这个论点是说,人们说的可计算函数恰好是可用一种图灵机的抽象机所计算的函数。这种抽象机是图灵在 1936 年的一篇论文中设计的,现在人们简称为图灵机,简记为 TM。该装置极其简单,功能却很强大,它基本上模拟了人的学习过程——读和写。他在这篇论文中的原话是:"通常称作算法的过程,恰好可在抽象机上进行。"

关于 TM,当时人们意见不一。因为这个论点既不能证明,也不能否定。大部分人认为这个论点本身体现了计算机科学的一条原理。反对意见之一:TM 系统不够丰富,太简单。反对意见之二:TM 系统过于丰富,存在着理论上 TM 可计算的函数,而不是实际上可计算的函数。这个意见后来成为研究计算复杂性的一个推动力。

不久,克林(S. C. Kleene)证明图灵可计算的函数是一般递归函数。

定义 2(丘奇论点)　凡可计算的函数和 λ 可定义的函数相同。

丘奇提出的这个论点与图灵论点有异曲同工之妙。丘奇借助于一个语法极其简单功能却很强大的莱姆德(Lambda)系统(用希腊字母 λ 表示),让 λ 表达式来写可计算的函数。在这个 λ 演算系统中没有特殊的函数符号,例如$\sqrt{\ }$,sin 等,用"λ"这个希腊字作为所有可计算函数的标准省缺名,所有特殊的函数符号仅为简记而引入。这个系统中仅有两种运算,它们互为逆运算,一种是"抽象"运算,用符号 λ 表示,另一种是"作用"运算,未引进任何符号来表示。丘奇的意思是,凡可计算的函数,均可用冠以 λ 的表达式写出来。

最近的研究又发现,λ 演算的语法和语义可以看成范例型的程序设计语言。受 λ 演算理论的启发,$S-m-n$ 定理、递归定理等得到证明。

丘奇和克林相继完成了 λ 可定义的函数也是一般递归函数的证明。这样,TM 可计算的函数集与 λ 可定义的函数集是同一个函数集。人们常将丘奇和图灵的论点合为一个丘奇-图灵论点:可计算函数是一般递归函数。

定义 3(马尔可夫论点)　一切可计算的函数都有算法可计算之,且这些算

法均可化为正规算法。

这个论点与上述两个论点等价。

该定义中的“算法”应为“演算”,因为算法描述的过程一定会终止,而演算描述过程未必会终止。

定义 4

(1) 函数 f 是可计算的,指当自变元的值 x 给定以后,如 $f(x)$有定义,则可在有限步内得出该函数的值;此外,当 $f(x)$无定义时,亦能在有限步内判知,则说 f 是完全可计算的,如不能在有限步内判知,则说 f 是半可计算的。

(2) 谓词 A 是可判定的,指当自变元的值 x 给定以后,如 $A(x)$有定义且为真,则可在有限步内判知,当 $A(x)$无定义时或 $A(x)$有定义且为假时,亦能在有限步内判知,则说谓词 A 是完全可判定的。

注:“可计算函数”是广义概念,实际上指半可计算更合适。因为相应的 TM 不终止或演算过程不终止时,此可计算函数实为半可计算函数。定义 4 取自莫绍揆教授写的“递归论”。

定义 5

(1) 一个函数 f 有定义且有计算它的程序,则该函数是可计算的。

(2) 递归函数是可计算的。

(3) 可由 P-T(Post-Turing)程序计算的函数是可计算的。

P-T 程序指在机器的存储带上有一个序列,它记录函数 $f(x)$的计算过程 $x,\cdots,f(x)$。

P-T 机的计算是局部的:P-T 程序在一条线性带上操作,带是双向无穷的,在计算的任意步,带上是“1”和“B”的序列;机器读头读带上一个符号;读头动作:左移、右移、写、抹,指令相应有:左移、右移、写“1”、写“B”。

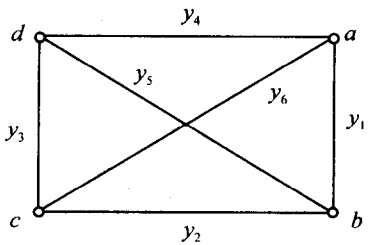
定义 5 来自戴维斯(Davis)的《可计算性和不可解性》一书。

定义 6 一个求解的问题可化为布尔线路的图、函数,则这个问题是可计算的(可解的),并能计算它的复杂度。

定义 6 由萨维奇(John. E. Savage)给出。定义 6 的计算模型与上述 5 种不一样,称为非一致计算模型。只要把求解的问题转化成逻辑电路,那么这类问题是可计算的。因此,凡能由谓词公式来表达的也是可计算的。

例如,一个 4 个节点的无向完全图与逻辑电路无关,但若用一定的方法转化成一个逻辑电路,则可讨论它的可计算性和计算复杂性了。

例 1 设 $G=\langle V,E\rangle$, $V=\{a,b,c,d\}$, $E=\{y_1,y_2,y_3,y_4,y_5,y_6\}$。



G 中有 4 个三角形：$\triangle abc$，$\triangle bcd$，$\triangle cda$，$\triangle dab$。

$\triangle abc$ 成立当且仅当

$$y_1, y_2, y_6 \text{ 存在}$$

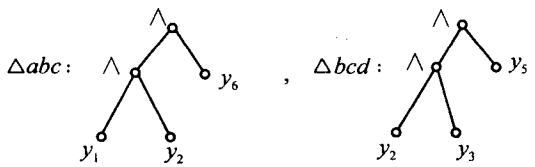
当且仅当

$$y_1 = y_2 = y_6 = 1$$

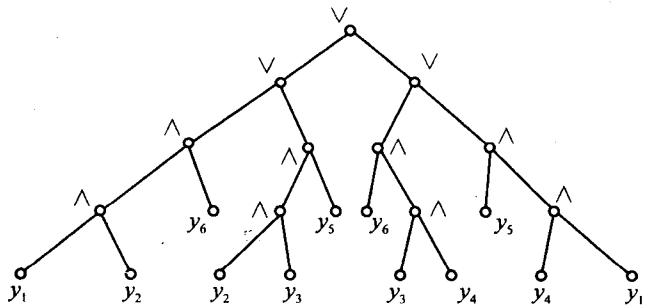
当且仅当

$$y_1 \wedge y_2 \wedge y_6 = 1$$

将 $\wedge$ 理解为“合取”，或“与门”；边 y_i 的存在理解为 $y_i = 1$。因此，三条边组成一个三角形化归为一个二元树表达，其节点是“与门”，对应逻辑乘（合取）；若干个三角形用逻辑加（析取）表示。只要约定：$+\overset{\text{对应}}{\text{——}}\vee\overset{\text{对应}}{\text{——}}\text{or}$，$\times\overset{\text{对应}}{\text{——}}\wedge\overset{\text{对应}}{\text{——}}\text{and}$ 便行。



$G = \langle V, E \rangle$ 变为下述逻辑电路。



实际上，图论中不少问题都是借助非一致计算模型来研究的。

定义 7　n 元数论谓词是可计算的指它的特征函数是可计算的。

这个定义将谓词的计算性化归为函数的计算性。

由丘奇等人的努力，计算函数的算法和程序的存在性进一步转化为函数的递归性。对不可计算的函数相应也有下面几种说法。

(1) 函数 f 不是递归的(这等于说 f 是不可计算的)。

(2) 某问题是不可判定的(若将该问题转化为函数，则化归出的函数不是递归的。这问题有点类似于证明题，它无程序或算法可用)。

(3) 某问题是不可解的(这问题相当于求作题，求解该问题的程序和算法不可能有)。

事实上存在不可计算的问题。著名的有 TM 的停机问题，PCP 问题的解等。有人把哥德巴赫(Goldbach)猜想也归入不可解类，但我们不知道这对不对。曾经把费马(Fermat)定理($x^n + y^n = z^n$，当 $n > 2$ 时无整数解)归入这一类，现在看来这是错了。1995 年，美国的安德罗·怀尔斯教授在他的一个学生帮助下，证出了这个定理。

我读到机械工业出版社的《计算理论导引》一书的前言，说到自然数的素因子分解问题。赛普塞（Michael Sipser）悲观地写道："一个大的自然数，譬如说有 500 位，……，现今还没有人知道怎样才能在宇宙毁灭之前做完这件事(指分解成素数的乘积)!" 这也是一个不可解的问题？但在此书中译本印刷(2002 年 2 月）不久，于 2002 年 8 月 6 日，印度数学家马宁德拉·阿格拉沃夫与两个大学生合作找到一个巧妙的算法，确定一个数是否是素数，而不管这个数有多大。

但人们似乎并不介意这个算法，全球狂热地寻找新素数的人越来越多。除美国克雷研究所外，全球约有 211 000 部计算机加入这个寻找世界上最新素数的"游戏"，而无一例外地还用类似于 1600 年法国人梅森（Mersenne）的方法。他们日夜兼程，志愿加入"互联网梅森素数搜索计划"（GIMPS），做这种从前只有数论专家做的事。成千上万台计算机马不停蹄地一路狂跑。2003 年 12 月，在美国密歇根大学研究生薛佛的计算上找到迄今最大的素数 $2^{20\,996\,011} - 1$ 且为梅森数，长达6 320 430位。记得 1994 年克雷研究所得到当时世界上最大的梅森素数是 $2^{859\,433} - 1$，有 258 716 位，是第 32 个梅森素数 M_{32}，即 $M_{32} = 2^{859\,433} - 1$。

2002 年，找到的次最大的梅森素数有 400 多万位。

看来把某一个问题归入不可解类，其本身也许是一个难解型的问题。

第三节 计算复杂性

上一节我们介绍了几种可计算性的定义，由克林、图灵、丘奇等人的努力，证

明了可计算函数是一般递归函数，在后面的有关章节我们将提供这种证明。但定义1、定义2和定义3，即图灵论点、丘奇论点和马尔可夫论点都是非形式概念，其本身的正确性我们苦于无法提供证明，只能给出“合理的证明”，只能由“迄今为止未有反例”来作证。可计算的概念直觉上等同于一般递归的概念，其实我们真的不知道一般递归函数是否包括所有的可计算函数。我们只能知道（只能证明），我们所知道的（我们能够证明的）。在下一节我们将会总结一下人们对可计算性概念的几个反面的看法。

有一点肯定的是，可计算的函数至少有可数无穷 $\aleph_0$ 那么多，甚至不可数无穷之多。

结论1　可计算的函数集至少是可数无穷集。

证明　假定依戴维斯的定义（第二节定义5），对于每个可计算的函数都存在一个计算它的程序或算法，又假定每一个计算机程序或算法都能有穷地表示。因此，计算机程序的集合与某个有穷字母表上的全体有穷长的符号串的集合等势。因而，所有计算机程序的集合是可数无穷的，可计算的函数集至少是可数无穷集。　□

结论2　可计算的函数集有不可数无穷之多。

证明　假定可计算函数组成一个可数无穷集。考虑将整数映射到{0,1}的数论函数 f，

$$f:\mathbf{N}\to\{0,1\},S=\{f\mid f:\mathbf{N}\to\{0,1\}\},|S|=\aleph_0$$

S 中每个 f 均与 $\mathbf{N}$ 的元素一一对应，即 $f\in S$ 时，均有编号 i，$i\in\mathbf{N}$，$f_i\in S$。S 与可计算函数集等势。

构造函数 g，$g:\mathbf{N}\to\{0,1\}$

设 $n\in\mathbf{N}$，$\begin{cases}g(n)=0 & \text{如 } f_n(n)=1\\ g(n)=1 & \text{否则}\end{cases}$

如果 f_i 可计算，则 g 亦可计算：

$$g(1)\neq f_1(1),g(2)\neq f_2(2),\cdots,g(n)\neq f_n(n)$$

所以

$$g\in S$$

否则，设

$$g(x)=f_j(x)$$

则

$g(j)=f_j(j)$与 $g(j)\neq f_j(j)$矛盾。因此 S 不是可数无穷集，可计算函数集不是可数无穷集。 □

在这么多的可计算函数里，这些函数被计算时，它们的难度相当吗？答案是否定的。研究可计算函数的计算难度就是被称为计算复杂性理论的内容。问题的难度分为两大类，顽型和易型。

定义 1(顽型问题(intractable problems)) 指用抽象 TM 作模型时需要比较多的时间和空间资源，它超出了多项式复杂性的范围。

也就是说，解此类问题时，找不到多项式复杂性的算法。举一个几乎每本书都举的经典例子。设有两个算法，一个是耗时(耗空间)3^n；另一个是 n^3，当 $n=60$ 时，用百万次计算机解，前者要 10^{15}a，后者只要 0.2s。可见多项式算法才是现实所要的算法。

定义 2(易型问题) 指用抽象 TM 作模型时，该问题有多项式复杂性时间和空间的算法。

计算复杂性理论就是研究和断定一个问题是否是顽型问题的理论。

研究时通常利用时间复杂性函数 $T(n)$和空间复杂性函数 $S(n)$，考虑当 n 增大时，$T(n)$和 $S(n)$的极限情形，称作渐近时间复杂性和渐近空间复杂性。

对算法的复杂性大多只给出它们关于问题大小的数量级。例如对某个常数 $c>0$，一个算法在 cn^2 时间内处理完大小为 n 的输入，就说这个算法的时间复杂度为 $O(n^2)$，读作大喔 n^2，或说 n 的平方级。另外还有 $o(n^2)$，读作小喔 n^2，等等。现约定如下(以下记号在本文中用到)。

(1) $f(n)=O(g(n))$

若 $f(n)$，$g(n)$是数论函数，当 n 充分大后，$f(n)\leqslant cg(n)$，c 为常数，则记为

$f(n)=O(g(n))$，$f(n)$为大喔 $g(n)$。

(2) $f(n)=o(g(n))$

若 $\lim\limits_{n\to\infty}\dfrac{f(n)}{g(n)}=0$，则

$f(n)=o(g(n))$，$f(n)$为小喔 $g(n)$。

(3) $f(n)=\Theta(g(n))$

如果 $f(n)=o(g(n))$且 $g(n)=o(f(n))$，则 $f(n)=\Theta(g(n))$。

(4) $\sup\limits_{n\to\infty}f(n)$，$\inf\limits_{n\to\infty}f(n)$

$\sup_{n\to\infty} f(n)$表示 n 趋于无穷时，$f(n)$，$f(n+1)$，…最小上界（上确界）的极限。$\inf_{n\to\infty} f(n)$表示 $f(n)$，$f(n+1)$，…在 n 趋于无穷时，最大下界（下确界）的极限。

(5) $f(n)$a.e.…：$f(n)$除有穷多个值外……

a.e.即 almost everywhere，“几乎处处”的意思。

(6) $f(n)$i.o.…：$f(n)$对无穷多个值……

i.o.即 infinite often 之意思，“有无穷多个值”使 $f(n)$……

对易型问题的研究借助于抽象的 TM。一种是确定型的 TM，记为 DTM (Determine Turing Machine)；另一种是不确定的 TM，记为 NTM (Nondetermine Turing Machine)。易型问题由于使用上述两种不同的 TM，又区分为两类问题。

定义 3

(1) P 类问题　一个问题用 DTM 作工具时，有多项式复杂性的算法，则该问题被称为是一个 P 类问题。

(2) NP 类问题　若某问题用 NTM 为工具时，有多项式复杂性的算法，则该问题就属 NP 类问题。

现重要的问题是：“P 类问题就是 NP 类问题？反之亦然？”，这即所谓“P＝NP?”。

由 P 类问题和 NP 类问题又引申出下面的概念。

1．NP 完全问题

有人指出，NP 类中有一小类问题，具有以下性质：迄今为止没有找到以 DTM 为工具时的多项式算法，但一旦其中一个问题找到这种算法，那么 P＝NP。这一小类问题被称为 NP 完全问题。这些问题大多是经过长期研究，但找不到多项式算法的问题，而 NP 完全却把它们联系在一起。因而，多数人猜想 NP≠P。

2．完全性问题

由 NP 完全问题启发，人们研究各种完全问题，即研究不同的完全性问题的集合。例如确定的空间完全性问题，即 DSPACE 完全问题；不确定的空间完全性问题，即 NSPACE 完全问题。仿此，PTIME 完全问题，等等。

3．NP 完全理论

这种理论并不打算找出 NP 完全问题的算法，仅仅着眼于这类问题的等价性的证明，即证明这类问题难度相当。

4．NP 难题

设某问题是已知的 NP 类问题之一，另一问题能在多项式时间内转换成已知的 NP 问题，则该新问题是一个 NP 难题。

5．对难题的分类有如下研究结果

$$P\ SPACE = NP\ SPACE$$

但尚不知

$$NP\ TIME = P\ TIME \text{ 和 } P\ TIME = P\ SPACE$$

还有其他结果在后面再介绍。

据目前资料说明，直接证明 P＝NP 很难，间接证明 P＝NP 与直接证明一样难，证出的希望同样地差。如果 NP≠P，那么许多现实而又迫切的问题都不可能在现行的计算机上解决。

第四节　对可计算性定义的质疑

由图灵－丘奇论点我们定义了可计算函数的集合，并由克林等人的证明说明了一般递归函数集，λ 可定义的函数集，TM 计算的函数集，马尔可夫算法计算的函数集，用布尔线路表达的函数集和 P－T 机计算的函数集一样大。

由第三节结论 2，利用康托的对角线法证明至少有不可数之多的可计算函数，也就是说我们人类面对的将是不可数多个可计算函数。然而我们人类到今天为止总共计算了有穷多个可计算的函数，还有无穷多个可计算函数我们连影子也未见到。有穷毕竟是无穷的初始段，留在初始段后面那不可数无穷长的尾巴无法掌握。俗话说尾大难握，这长长的尾巴永远有可能随时掉转方向，因此这里有一个漏洞：一般递归函数包括了所有可计算的函数吗？

另一个疑问是，罗莎·培特认为一般递归函数之所以处处可计算，是因为“摹状式有根”，而摹状式的根牵涉到存在性。而古典意义的存在未必能构造出来。这里的“存在”应该是能行的，而能行的标准又是“可以表示为一般递归函数”。这里有一个循环无法逃脱，用直觉主义的观点来看，这里也有一个漏洞。

近几十年计算机事业的发展，重大成就的取得得益于直觉主义学派的努力。直觉主义认为，数学和逻辑相比，数学是第一位的，逻辑是第二位的（弗雷格(Frege)、拉塞尔(Russell)，布劳威尔(Brouwer)都表示过）。同时，一个命题的语义不在它的真值条件中，而在它的验证和证明之中。简单地说，直觉主义者坚持，你说此物有，那你就将它拿出来或者构造出来。这正是与我们的计算机行为一致的地方。

练习题

1. 将下述语句译成用谓词、量词表达的逻辑式子。

(1) 罗格确认了2008年北京奥运会会徽并慎重地在申请书上盖上刻有他的名字的中国印。

(2) 29届奥林匹克运动会的会徽似印非印,似京非京,寓意是舞动的北京。

(3) 有不可数多个可计算的函数。

(4) 哈勃望远镜对行星和恒星的观察达到历史最高水平。

(5) $f(x)$在区间(a,b)内一致连续(用$\varepsilon-\delta$表示一致连续,然后翻译成逻辑式子)。

2. 你还见到哪些(除本章介绍过的)可计算性的定义吗? 它们与本章介绍的定义相等价吗? 为什么?

3. 你对可计算性理论的基础有哪些看法? 还有哪些疑问?

4. 请估计下列有关 n 的数量级是大喔,还是小喔,为什么?

(1) $3n=O(n)$?

(2) $2n^2=O(n)$?

(3) $n^2=O(n\log^2 n)$?

(4) $3^n=2^{O(n)}$?

(5) $n=o(2n)$?

(6) $n=o(\log n)$?

(7) $1=o(1/n)$? 为什么?

(8) $1=o(n)$? 为什么?