图灵与计算理论

**年轻时期**

艾伦·麦席森·图灵，1912年生于英国伦敦。艾伦·麦

[](https://baike.baidu.com/pic/%E8%89%BE%E4%BC%A6%C2%B7%E9%BA%A6%E5%B8%AD%E6%A3%AE%C2%B7%E5%9B%BE%E7%81%B5/3940576/0/d57e999449eab223d21b70c3?fr=lemma&ct=single)艾伦·麦席森·图灵

席森·图灵少年时就表现出独特的直觉创造能力和对数学的爱好。

1926年，他考入[伦敦](https://baike.baidu.com/item/%E4%BC%A6%E6%95%A6/862" \t "_blank)有名的舍本(Sherborne)公学，受到良好的中等教育．他在中学期间表现出对自然科学的极大兴趣和敏锐的数学头脑。

1927年末，年仅15岁的图灵为了帮助母亲理解[爱因斯坦](https://baike.baidu.com/item/%E7%88%B1%E5%9B%A0%E6%96%AF%E5%9D%A6" \t "_blank)的[相对论](https://baike.baidu.com/item/%E7%9B%B8%E5%AF%B9%E8%AE%BA)，写了爱因斯坦的一部著作的内容提要，表现出他已具备非同凡响的数学水平和科学理解力。他对自然科学的兴趣使他在1930年和1931年两次获得他的一位同学莫科姆的父母设立的自然科学奖，获奖工作中有一篇论文题为“[亚硫酸盐](https://baike.baidu.com/item/%E4%BA%9A%E7%A1%AB%E9%85%B8%E7%9B%90)和[卤化物](https://baike.baidu.com/item/%E5%8D%A4%E5%8C%96%E7%89%A9)在酸性溶液中的反应”，受到政府派来的督学的赞赏，对[自然科学](https://baike.baidu.com/item/%E8%87%AA%E7%84%B6%E7%A7%91%E5%AD%A6" \t "_blank)的兴趣为他后来的一些研究奠定了基础，他的数学能力使他在念中学时获得过国王爱德华六世数学金盾奖章。

**科研时期**

1931年，图灵考入[剑桥大学](https://baike.baidu.com/item/%E5%89%91%E6%A1%A5%E5%A4%A7%E5%AD%A6" \t "_blank)国王学院，由于成绩优异而获得数学奖学金。在剑桥，他的数学能力得到充分的发展。

1935年，他的第一篇数学论文“左右殆周期性的等价”发表于《[伦敦数学会](https://baike.baidu.com/item/%E4%BC%A6%E6%95%A6%E6%95%B0%E5%AD%A6%E4%BC%9A" \t "_blank)杂志》上。同一年，他还写出“论高斯误差函数”一文。这一论文使他由一名大学生直接当选为

[](https://baike.baidu.com/pic/%E8%89%BE%E4%BC%A6%C2%B7%E9%BA%A6%E5%B8%AD%E6%A3%AE%C2%B7%E5%9B%BE%E7%81%B5/3940576/0/c8ab0bce43f91a04b600c8cd?fr=lemma&ct=single)跑步比赛中的图灵

国王学院的研究员，并于次年荣获英国著名的史密斯(Smith)数学奖，成为国王学院声名显赫的毕业生之一。

1936年5月，图灵向伦敦权威的数学杂志投了一篇论文，题为《论数字计算在决断难题中的应用》。该文于1937年在《[伦敦数学会](https://baike.baidu.com/item/%E4%BC%A6%E6%95%A6%E6%95%B0%E5%AD%A6%E4%BC%9A" \t "_blank)文集》第42期上发表后，立即引起广泛的注意。在论文的附录里他描述了一种可以辅助数学研究的机器，后来被人称为“图灵机”，这个设想最牛的地方在于，它第一次在纯数学的符号逻辑，和实体世界之间建立了联系，后来我们所熟知的电脑，以及还没有实现的“[人工智能](https://baike.baidu.com/item/%E4%BA%BA%E5%B7%A5%E6%99%BA%E8%83%BD/9180)”，都基于这个设想。这是他人生第一篇重要论文，也是他的成名之作。

1937年，图灵发表的另一篇文章“可计算性与λ可定义性”则拓广了丘奇(Church)提出的“丘奇论点”，形成“丘奇-图灵论点”，对计算理论的严格化，对计算机科学的形成和发展都具有奠基性的意义。

1936年9月，图灵应邀到美国[普林斯顿](https://baike.baidu.com/item/%E6%99%AE%E6%9E%97%E6%96%AF%E9%A1%BF" \t "_blank)高级研究院学习，并与丘奇一同工作。

在美国期间，他对群论作了一些研究，并撰写了博士论文。1938年在普林斯顿获博士学位，其论文题目为“以序数为基础的逻辑系统”，1939年正式发表，在数理逻辑研究中产生了深远的影响。

1938年夏，图灵回到英国，仍在[剑桥大学](https://baike.baidu.com/item/%E5%89%91%E6%A1%A5%E5%A4%A7%E5%AD%A6" \t "_blank)国王学院任研究员，继续研究数理逻辑和计算理论，同时开始了计算机的研制工作。

**二战经历**

第二次世界大战打断了图灵的正常研究工作，1939年秋，他应召到英国外交部通信处从事军事工作，主要是破译敌方密码的工作。由于破译工作的需要，他参与了世界上最早的电子计算机的研制工作。他的工作取得了极好的成就，因而于1945年获政府的最高奖——大英帝国荣誉勋章(O．B．E．勋章)。

1945年，图灵结束了在外交部的工作，他试图恢复战前在理论计算机科学方面的研究，并结合战时的工作，具体研制出新的计算机来。这一想法得到当局的支持。同年，图灵被录用为泰丁顿(Teddington)国家物理研究所的研究人员，开始从事“自动计算机”(ACE)的逻辑设计和具体研制工作。这一年，图灵写出一份长达50页的关于ACE的设计说明书。这一说明书在保密了27年之后，于1972年正式发表。在图灵的设计思想指导下，1950年制出了ACE样机，1958年制成大型ACE机。人们认为，通用计算机的概念就是图灵提出来的。

1945年到1948年，他在英国国家物理实验室工作，负责自动计算引擎的研究。

1948年，图灵接受了[曼彻斯特大学](https://baike.baidu.com/item/%E6%9B%BC%E5%BD%BB%E6%96%AF%E7%89%B9%E5%A4%A7%E5%AD%A6" \t "_blank)的高级讲师职务，并被指定为[曼彻斯特](https://baike.baidu.com/item/%E6%9B%BC%E5%BD%BB%E6%96%AF%E7%89%B9/81402)自动数字计算机(Madam)项目的负责人助理，具体领导该项目数学方面的工作．作为这一工作的总结。

1949年成为曼彻斯特大学计算机实验室的副主任，负责最早的真正意义上的计算机——“曼彻斯特一号”的软件理论开发，因此成为世界上第一位把计算机实际用于数学研究的科学家。

1950年，图灵编写并出版了《曼彻斯特电子计算机程序员手册》(The programmers’handbook for the Manchester electronic computer)。这期间，他继续进行数理逻辑方面的理论研究。并提出了著名的“图灵测试”。

1950年，他提出关于机器思维的问题，他的论文“计算机和智能(Computingmachiery and intelligence)，引起了广泛的注意和深远的影响。1950年10月，图灵发表论文《机器能思考吗》。这一划时代的作品，使图灵赢得了“人工智能之父”的桂冠。

1951年，由于在可计算数方面所取得的成就，成为英国皇家学会会员，时年39岁。

1952年，他辞去剑桥大学国王学院研究员的职务，专心在曼彻斯特大学工作．除了日常工作和研究工作之外，他还指导一些博士研究生，还担任了制造曼彻斯特自动数字计算机的一家公司——弗兰蒂公司的顾问。1952年，图灵写了一个国际象棋程序。可是，当时没有一台计算机有足够的运算能力去执行这个程序，他就模仿计算机，每走一步要用半小时。他与一位同事下了一盘，结果程序输了。后来美国新墨西哥州洛斯阿拉莫斯国家实验室的研究群根据图灵的理论，在MANIAC上设计出世界上第一个电脑程序的象棋。

**被迫害后逝世**

1952年，图灵的同性伴侣协同一名同谋一起闯进

[](https://baike.baidu.com/pic/%E8%89%BE%E4%BC%A6%C2%B7%E9%BA%A6%E5%B8%AD%E6%A3%AE%C2%B7%E5%9B%BE%E7%81%B5/3940576/0/507c389741393b7b54fb964a?fr=lemma&ct=single)

了图灵的房子实施盗窃。图灵为此而报警。但是警方的调查结果使得他被控以“明显的猥亵和性颠倒行为”（[同性恋](https://baike.baidu.com/item/%E5%90%8C%E6%80%A7%E6%81%8B)）。他没有申辩，并被定罪。在著名的公审后，他被给予了两个选择：坐牢或[荷尔蒙疗法](https://baike.baidu.com/item/%E8%8D%B7%E5%B0%94%E8%92%99%E7%96%97%E6%B3%95)。他选择了[荷尔蒙](https://baike.baidu.com/item/%E8%8D%B7%E5%B0%94%E8%92%99/33422)注射，并持续了一年。在这段时间里，药物产生了包括乳房不断发育的副作用。

1954年6月7日，图灵被发现死于家中的床上，床头还放着一个被咬了一口的苹果。警方调查后认为是剧毒的[氰化物](https://baike.baidu.com/item/%E6%B0%B0%E5%8C%96%E7%89%A9" \t "_blank)中毒，调查结论为自杀。当时图灵41岁。

**正式平反**

2009年，英国计算机科学家康明（John Graham-Cumming）发起了为图灵平反的在线请愿，截止到2009年9月10日请愿签名人数已经超过了3万，为此，当时的英国政府及首相戈登布朗不得不发表正式的道歉声明。

2012年12月，[霍金](https://baike.baidu.com/item/%E9%9C%8D%E9%87%91" \t "_blank)、纳斯（Paul Nurse，[诺贝尔医学奖](https://baike.baidu.com/item/%E8%AF%BA%E8%B4%9D%E5%B0%94%E5%8C%BB%E5%AD%A6%E5%A5%96" \t "_blank)得主）、里斯（Martin Rees，[英国皇家学会](https://baike.baidu.com/item/%E8%8B%B1%E5%9B%BD%E7%9A%87%E5%AE%B6%E5%AD%A6%E4%BC%9A" \t "_blank)会长）等11位重要人士致函英国首相卡梅伦，要求为其平反。

2013年12月24日，在英国司法部长克里斯・格雷灵（Chris Grayling）的要求下，英国女王终于向图灵颁发了的皇家赦免。英国司法部长宣布，“图灵的晚年生活因为其同性取向而被迫蒙上了一层阴影，我们认为当时的判决是不公的，这种歧视现象现在也已经遭到了废除。为此，女王决定为这位伟人送上赦免，以此向其致敬。” [1]

**主要成就**

[编辑](javascript:;)

图灵在科学、特别在[数理逻辑](https://baike.baidu.com/item/%E6%95%B0%E7%90%86%E9%80%BB%E8%BE%91/18105" \t "_blank)和计算机科学方面，取得了举世瞩目的成就，他的一些科学成果，构成了现代计算机技术的基础。

**可计算性理论**

[](https://baike.baidu.com/pic/%E8%89%BE%E4%BC%A6%C2%B7%E9%BA%A6%E5%B8%AD%E6%A3%AE%C2%B7%E5%9B%BE%E7%81%B5/3940576/0/cc506c8ba43d762fc8fc7ac8?fr=lemma&ct=single)剑桥大学国王学院的计算机房现在以图灵为名

　计算，可以说是人类最先遇到的数学课题，并且在漫长的历史年代里，成为人们社会生活中不可或缺的工具．那么，什么是计算呢？直观地看，计算一般是指运用事先规定的规则，将一组数值变换为另一(所需的)数值的过程．对某一类问题，如果能找到一组确定的规则，按这组规则，当给出这类问题中的任一具体问题后，就可以完全机械地在有限步内求出结果，则说这类问题是可计算的。这种规则就是[算法](https://baike.baidu.com/item/%E7%AE%97%E6%B3%95" \t "_blank)，这类可计算问题也可称之为存在算法的问题。这就是直观上的能行可计算或算法可计算的概念．

在20世纪以前，人们普遍认为，所有的问题类都是有算法的，人们的计算研究就是找出算法来。似乎正是为了证明一切科学命题，至少是一切数学命题存在算法，莱布尼茨(Leibniz)开创了数理逻辑的研究工作。但是20世纪初，人们发现有许多问题已经过长期研究，仍然找不到算法，例如希尔伯特第10问题，半群的字的问题等．于是人们开始怀疑，是否对这些问题来说，根本就不存在算法，即它们是不可计算的。这种不存在性当然需要证明，这时人们才发现，无论对算法还是对可计算性，都没有精确的定义！按前述对直观的可计算性的陈述，根本无法作出不存在算法的证明，因为“完全机械地”指什么？“确定的规则”又指什么？仍然是不明确的。实际上，没有明确的定义也不能抽象地证明某类问题存在算法，不过存在算法的问题一般是通过构造出算法来确证的，因而可以不涉及算法的精确定义问题。

解决问题的需要促使人们不断作出探索。1934年，[哥德尔](https://baike.baidu.com/item/%E5%93%A5%E5%BE%B7%E5%B0%94" \t "_blank)(Godel)在埃尔布朗(Herbrand)的启示下提出了一般递归函数的概念，并指出：凡算法可计算函数都是一般递归函数，反之亦然。1936年，克里尼(Kleene)又加以具体化．因此，算法可计算函数的一般递归函数定义后来被称为埃尔布朗-哥德尔-克里尼定义．同年，丘奇证明了他提出的λ可定义函数与一般递归函数是等价的，并提出算法可计算函数等同于一般递归函数或λ可定义函数，这就是著名的“丘奇论点”。

用一般[递归函数](https://baike.baidu.com/item/%E9%80%92%E5%BD%92%E5%87%BD%E6%95%B0)虽给出了可计算函数的严格数学定义，但在具体的计算过程中，就某一步运算而言，选用什么初始函数和基本运算仍有不确定性。为消除所有的不确定性，图灵在他的“论可计算数及其在判定问题中的应用”一文中从一个全新的角度定义了可计算函数。他全面分析了人的计算过程，把计算归结为最简单、最基本、最确定的操作动作，从而用一种简单的方法来描述那种直观上具有机械性的基本计算程序，使任何机械(能行)的程序都可以归约为这些动作。这种简单的方法是以一个抽象自动机概念为基础的，其结果是：算法可计算函数就是这种自动机能计算的函数。这不仅给计算下了一个完全确定的定义，而且第一次把计算和自动机联系起来，对后世产生了巨大的影响，这种“自动机”后来被人们称为“图灵机”。

图灵机是一种自动机的数学模型，它是一条两端(或一端)无限延长的纸带，上面划成方格，每个方格中可以印上某字母表中的一个字母(亦可为空格，记为S0)；又有一个读写头，它具有有限个内部状态。任何时刻读写头都注视着纸带上的某一个方格，并根据注视方格的内容以及读写头当时的内部状态而执行变换规则所规定的动作。每个图灵机都有一组变换规则，它们具有下列三种形状之一：

qiaRqi，qiaLqi，qiabqj

意思是：当读写头处于状态qi时如果注视格的内容为字母a则读写头右移一格，或左移一格，或印下字母b(即把注视格的内容由a改成b．a，b可为S0)。

图灵把可计算函数定义为图灵机可计算函数．1937年，图灵在他的“可计算性与λ可定义性”一文中证明了图灵机可计算函数与λ可定义函数是等价的，从而拓广了丘奇论点，得出：算法(能行)可计算函数等同于一般递归函数或λ可定义函数或图灵机可计算函数．这就是“丘奇-图灵论点”，相当完善地解决了可计算函数的精确定义问题，对数理逻辑的发展起了巨大的推动作用。

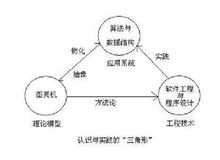
图灵机的概念有十分独特的意义：如果把图灵机的内部状态解释为指令，用字母表的字来表示，与输出字输入字同样存贮在机器里，那就成为电子计算机了。由此开创了“自动机”这一学科分支，促进了电子计算机的研制工作．

与此同时，图灵还提出了通用图灵机的概念，它相当于通用计算机的解释程序，这一点直接促进了后来通用计算机的设计和研制工作，图灵自己也参加了这一工作。

在给出通用图灵机的同时，图灵就指出，通用图灵机在计算时，其“机械性的复杂性”是有临界限度的，超过这一限度，就要靠增加程序的长度和存贮量来解决．这种思想开启了后来计算机科学中计算复杂性理论的先河。

**判定问题**

所谓“判定问题”指判定所谓“大量问题”是否具有算

[](https://baike.baidu.com/pic/%E8%89%BE%E4%BC%A6%C2%B7%E9%BA%A6%E5%B8%AD%E6%A3%AE%C2%B7%E5%9B%BE%E7%81%B5/3940576/0/8d158aee412ff5142df534cb?fr=lemma&ct=single)图灵机模型的理论

法解，或者是否存在能行性的方法使得对该问题类的每一个特例都能在有限步骤内机械地判定它是否具有某种性质(如是否真，是否可满足或是否有解等，随大量问题本身的性质而定)的问题。

判定问题与可计算性问题有密切的联系，二者可以相互定义：对一类问题若能找到确定的算法以判定其是否具有某种性质，则称这类问题是能行可判定的，或可解的；否则是不可判定的，或不可解的。二者又是有区别的：判定问题是要确定是否存在一个算法，使对一类问题的每一个特例都能对某一性质给以一个“是”或“否”的解答；可计算性问题则是找出一个算法，从而求出一些具体的客体来。

图灵在判定问题上的一大成就是把图灵机的“停机问题”作为研究许多判定问题的基础，一般地，把一个判定问题归结为停机问题：“如果问题A可判定，则停机问题可判定．”从而由“停机问题是不可判定的”推出“问题A是不可判定的”。

所谓停机指图灵机内部达到一个结果状态、指令表上没有的状态或符号对偶，从而导致计算终止。在每一时刻，机器所处的状态，纸带上已被写上符号的所有格子以及机器当前注视的格子位置，统称为机器的格局。图灵机从初始格局出发，按程序一步步把初始格局改造为格局的序列。此过程可能无限制继续下去，也可能遇到指令表中没有列出的状态、符号组合或进入结束状态而停机。在结束状态下停机所达到的格局是最终格局，此最终格局(如果存在)就包含机器的计算结果。所谓停机问题即是：是否存在一个算法，对于任意给定的图灵机都能判定任意的初始格局是否会导致停机？图灵证明，这样的算法是不存在的，即停机问题是不可判定的，从而使之成为解决许多不可判定性问题的基础。

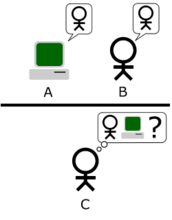
1937年，图灵用他的方法解决了著名的希尔伯特判定问题：狭谓词演算(亦称一阶逻辑)公式的可满足性的判定问题。他用一阶逻辑中的公式对图灵机进行编码，再由图灵机停机问题的不可判定性推出一阶逻辑的不可判定性。他在此处创用的“编码法”成为后来人们证明一阶逻辑的公式类的不可判定性的主要方法之一。

在判定问题上，图灵的另一成果是1939年提出的带有外部信息源的图灵机概念，并由此导出“图灵可归约”及相对递归的概念。运用归约和相对递归的概念，可对不可判定性与非递归性的程度加以比较。在此基础上，E．波斯特(Post)提出了不可解度这一重要概念，这方面的工作后来有重大的进展。

图灵参与解决的另一个著名的判定问题是“半群的字的问题”，它是图埃(Thue)在1914年提出来的：对任意给定的字母表和字典，是否存在一种算法能判定两个任意给定的字是否等价[给出有限个不同的称为字母的符号，便给出了字母表，字母的有限序列称为该字母表上的字。把有限个成对的字(A1，B1)，…，(An，Bn)称为字典．如果两个字R和S使用有限次字典之后可以彼此变换，则称这两个字是等价的]1947年，波斯特和A．A．马尔科夫(Markov)用图灵的编码法证明了这一问题是不可判定的。1950年，图灵进一步证明，满足消元律的半群的字的问题也是不可判定的。

**电子计算机**

图灵在第二次世界大战中从事的[密码破译](https://baike.baidu.com/item/%E5%AF%86%E7%A0%81%E7%A0%B4%E8%AF%91" \t "_blank)工作涉及到电子计算机的设计和研制，但此项工作严格保密。直到70年代，内情才有所披露。从一些文件来看，很可能世界上第一台电子计算机不是ENIAC，而是与图灵有关的另一台机器，即图灵在战时服务的机构于1943年研制成功的CO-LOSSUS(巨人)机，这台机器的设计采用了图灵提出的某些概念。它用了1500个电子管，采用了光电管阅读器；利用穿孔纸带输入

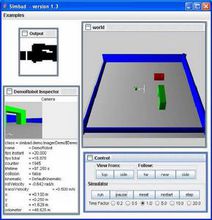
[](https://baike.baidu.com/pic/%E8%89%BE%E4%BC%A6%C2%B7%E9%BA%A6%E5%B8%AD%E6%A3%AE%C2%B7%E5%9B%BE%E7%81%B5/3940576/0/6d81800a19d8bc3e17e532b8848ba61ea9d345d6?fr=lemma&ct=single)图灵测试

；并采用了电子管双稳态线路，执行计数、二进制算术及布尔代数逻辑运算，巨人机共生产了10台，用它们出色地完成了密码破译工作．

战后，图灵任职于泰丁顿国家物理研究所(Teddington National Physical Laboratory)，开始从事“自动计算机”(Automatic Computing Engine)的逻辑设计和具体研制工作。1946年，图灵发表论文阐述存储程序计算机的设计。他的成就与研究离散变量自动电子计算机（Electronic Discrete Variable Automatic Computer）的约翰·冯·诺伊曼（John von Neumann）同期。图灵的自动计算机与诺伊曼的离散变量自动电子计算机都采用了二进制，都以“内存储存程序以运行计算机”打破了那个时代的旧有概念。

**人工智能**

1949年，图灵成为曼切斯特大学（University of Manchester ）计算实

[](https://baike.baidu.com/pic/%E8%89%BE%E4%BC%A6%C2%B7%E9%BA%A6%E5%B8%AD%E6%A3%AE%C2%B7%E5%9B%BE%E7%81%B5/3940576/0/7aad4ae7abe4e013b93820d0?fr=lemma&ct=single)机器人足球人工智能算法分析

验室的副院长，致力研发运行Manchester Mark 1型号储存程序式计算机所需的软件。1950年他发表论文《计算机器与智能》（ Computing Machinery and Intelligence），为后来的人工智能科学提供了开创性的构思。提出著名的“[图灵测试](https://baike.baidu.com/item/%E5%9B%BE%E7%81%B5%E6%B5%8B%E8%AF%95)”，指出如果第三者无法辨别人类与人工智能机器反应的差别， 则可以论断该机器具备人工智能。

1956年图灵的这篇文章以“机器能够思维吗？”为题重新发表．此时，人工智能也进入了实践研制阶段。图灵的机器智能思想无疑是人工智能的直接起源之一。而且随着人工智能领域的深入研究，人们越来越认识到图灵思想的深刻性：它们至今仍然是人工智能的主要思想之一。

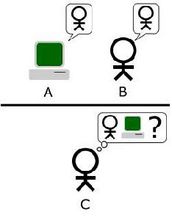
**数理生物学**

从1952年直到去世，图灵一直在数理生物学方面做研究。他在1952年发表了一篇论文《形态发生的化学基础》(The Chemical Basis of Morphogenesis)。他主要的兴趣是斐波那契叶序列，存在于植物结构的斐波那契数。他应用了反应-扩散公式，现在已经成为图案形成范畴的核心。他后期的论文都没有发表，一直等到1992年《艾伦·图灵选集》出版，这些文章才见天日。

**图灵试验**

1945年到1948年，图灵在国家物理实验室，负责自动计算引擎（ACE）的工作 。1949年，他成为曼彻斯特大学计算机实验室的副主任，负责最早的真正的计算机---曼彻斯特一号的软件工作。在这段时间，他继续作一些比较抽象的研究，如“计算机械和智能”。图灵在对人工智能的研究中，提出了一个叫做图灵试验的实验，尝试定出一个决定机器是否有感觉的标准。

[图灵试验](https://baike.baidu.com/item/%E5%9B%BE%E7%81%B5%E8%AF%95%E9%AA%8C)由计算机、被测试的人和主持试验人组成。计算机和被测试的人分别在两个不同的房间里。测试过程由主持人提问，由计算机和被测试的人分别做出回答。观测者能通过电传打字机与机器和人联系（避免要求机器模拟人外貌和声音）。被测人在回答问题时尽可能表明他是一个“真正的”人，而计算机也将尽可能逼真的模仿人的思维方式和思维过程。如果试验主持人听取他们各自的答案后，分辨不清哪个是人回答的，哪个是机器回答的，则可以认为该计算机具有了智能。这个试验

[](https://baike.baidu.com/pic/%E8%89%BE%E4%BC%A6%C2%B7%E9%BA%A6%E5%B8%AD%E6%A3%AE%C2%B7%E5%9B%BE%E7%81%B5/3940576/0/bd7faf35f8c14db6a61e12de?fr=lemma&ct=single)图灵试验

可能会得到大部分人的认可，但是却不能使所有的哲学家感到满意。图灵试验虽然形象描绘了计算机智能和人类智能的模拟关系，但是图灵试验还是片面性的试验。通过试验的机器当然可以认为具有智能，但是没有通过试验的机器因为对人类了解的不充分而不能模拟人类仍然可以认为具有智能。图灵试验还有几个值得推敲的地方，比如试验主持人提出问题的标准，在试验中没有明确给出；被测人本身所具有的智力水平，图灵试验也疏忽了；而且图灵试验仅强调试验结果，而没有反映智能所具有的思维过程。所以，图灵试验还是不能完全解决机器智能的问题。例如：质问者可以说：“我听说，今天上午一头犀牛在一个粉红色的气球中沿着密西西比河飞。你觉得怎样？”(你们可以想像该电脑的肩头上泛出的冷汗：)电脑也许谨慎地回答： “我听起来觉得这不可思议，”到此为止没有毛病。质问者又问： “是吗?我的叔叔试过一回，顺流、逆流各一回，它只不过是浅色的并带有斑纹。 这有什么不可思议的?”很容易想像，如果电脑没有合适的“理解”就会很快地暴露了自己、在回答第一个问题时，电脑的记忆库非常有力地想列犀牛没有翅膀，甚至可以在无意中得到“犀牛不能飞”，或者这样回答第二个问题“犀牛没有斑纹”。下一回质问者可以试探真正无意义的问题．譬如把它改变成“在密西西比河下面”，或者“在一个粉红色的气球之外”．或者“穿一件粉红色衣服”，再去看看电脑是否感觉到真正的差别。其实，要求电脑这样接近地模仿人类，以使得不能和一个人区分开实在是太过分了。一些专家认为，我们不该以电脑能否思维为目标，而是以能多大程度地模仿人类思维为目标；然后，让设计者再朝着这个目标努力。1952年，图灵写了一个[国际象棋](https://baike.baidu.com/item/%E5%9B%BD%E9%99%85%E8%B1%A1%E6%A3%8B" \t "_blank)程序。可是，当时没有一台计算机有足够的运算能力去执行这个程序，他就模仿计算机，每走一步要用半小时。他与一位同事下了一盘，结果程序输了。后来美国新墨西哥州洛斯阿拉莫斯国家实验室的研究群根据图灵的理论，在MANIAC上设计出世界上第一个电脑程序的象棋。

可计算性（Computability）”是可计算性理论的核心概念，具有深刻的数学内涵和哲学底蕴，图灵、丘奇、哥德尔等前辈的工作为此概念打下了坚实的基础，应该说对此概念的理解已经不成问题了，然而从“NP是可计算的”流行观念看，此概念并未得到人们充分而正确的解读，这或许是造成千禧年难题“P versus NP”的最根本原因。

我们从回答“什么是可计算性？”说起，来解读学术界流行观点的回答，指出“可计算性”蕴含着“机器”与“问题”二个不同的层次：

一，学术界回答“什么是可计算性？”的典型答案

1，维基（[https://zh.wikipedia.org/wiki/可计算性](https://zh.wikipedia.org/wiki/可计算性" \t "_blank)）

可计算性（calculability）是指一个实际问题是否可以使用计算机来解决。

2，维基（[https://en.wikipedia.org/wiki/Computability](https://en.wikipedia.org/wiki/Computability" \t "_blank)）

Computability is the ability to solve a problem in an effective manner. It is a key topic of the field of computability theory within mathematical logic and the theory of computation within computer science. The computability of a problem is closely linked to the existence of an algorithm to solve the problem.

One goal of computability theory is to determine which problems, or classes of problems, can be solved in each model of computation.

3，Michael Sipser的书“Introduction to the Theory of Computation”

此书是计算理论领域的经典著作，常被大学选为教材。书中干脆就不谈“可计算性”，直接用“可计算函数”代替（p. 234）：

COMPUTABLE FUNCTIONS：A Turing machine computes a function by starting with the input to the function on the tape and halting with the output of the function on the tape.

4，Stanford Encyclopedia of Philosophy（<http://plato.stanford.edu/entries/computability/>）

A mathematical problem is computable if it can be solved in principle by a computing device.

二，解读“可计算性（Computability）”与“可计算的（computable）”

可见，上述回答实际上是在答“什么是可计算问题（computable problem）？”，而不是直接答“什么是可计算性（Computability）？”，那么，我们不禁要问：“computability”与“computable problem”是一回事吗？

对于“computable problem”的标准答案是由丘奇－图灵论题给出的（https://en.wikipedia.org/wiki/Church–Turing\_thesis）：

J. B. Rosser (1939) addresses the notion of "effective computability" as follows: "Clearly the existence of CC and RC (Church's and Rosser's proofs) presupposes a precise definition of 'effective'. 'Effective method' is here used in the rather special sense of a method each step of which is precisely predetermined and which is certain to produce the answer in a finite number of steps". Thus the adverb-adjective "effective" is used in a sense of "1a: producing a decided, decisive, or desired effect", and "capable of producing a result".

In the following, the words "effectively calculable" will mean "produced by any intuitively 'effective' means whatsoever" and "effectively computable" will mean "produced by a Turing-machine or equivalent mechanical device". Turing's "definitions" given in a footnote in his 1939 Ph.D. thesis Systems of Logic Based on Ordinals, supervised by Church, are virtually the same:

"† We shall use the expression 'computable function' to mean a function calculable by a machine, and let 'effectively calculable' refer to the intuitive idea without particular identification with any one of these definitions."

The thesis can be stated as follows:

Every effectively calculable function is a computable function.[8]

Turing stated it this way:

"It was stated ... that 'a function is effectively calculable if its values can be found by some purely mechanical process.' We may take this literally, understanding that by a purely mechanical process one which could be carried out by a machine. The development ... leads to ... an identification of computability† with effective calculability." († is the footnote above, ibid.)

“如果一个函数的值能被某个纯粹的机械过程求得，那么此函数就是能行可计算的”，可见，“可计算函数”是须要前提的，此前提就是存在着一台可以求解函数值的机器，换句话说，“可计算性”涉及到“问题”与“机器”二个层次，“机器的能力”为本，“问题的性质”为末。于是，用“可计算问题”代替“可计算性”，实际上是忽略了“可计算性”蕴含的二个层次的关系，从而造成认知混淆，。。。

所以，对于“可计算性”的理解应该回到对“机器的能力”的认知上，即回到对“图灵机”的认知上来。追本溯源，“图灵机”是在图灵1936年那篇重要论文《论可计算数及其在判定问题上的应用》（ On Computable Numbers, with an Application to the Entscheidungsproblem）中提出来的，让我们一起回到这篇具有历史性意义，但又令人望而生畏，至今使人无法深入的奠基性的论文。

三，《论可计算数及其在判定问题上的应用》中关于“图灵机”的论述

图灵在论文中开门见山指出，“按照我的定义，一个数是可计算的，如果它的十进制的表达能被机器写下来。”就是说，图灵强调预期的结果一定要被机器写下来，才能认为一个数是可计算的。

接下来，他开始设计这样的机器，将之称为“computing machine”。于“Computing machine”，图灵又区分了“circular machine”和“circle-free machine”，“circular machine”是因某些因素如“死循环”而无法写下计算结果的机器；而“circle-free machine”没有这样阻碍，能写下预期结果，换句话说， “circle-free”表达了图灵称之的“可计算性（computability）”。

然而，一个问题是否具有“computability”须要判断，这就是图灵这篇奠基性的论文的主题，回答“可计算性的判断”，进而回答著名的希尔伯特第十问题，。。。

让我们回到问题：“什么是可计算性？”，相对于流行答案“可计算性是指一个实际问题是否可以使用计算机来解决”，应该说：“可计算性是指算法（图灵机）具有解决一个实际问题的能力”，“可计算问题是指存在着具有可计算性的算法的问题”。。。

附：

1，图灵论文“On Computable Numbers, with an Application to the Entscheidungsproblem”原文：[https://www.cs.virginia.edu/~robins/Turing\_Paper\_1936.pdf](https://www.cs.virginia.edu/~robins/Turing_Paper_1936.pdf" \t "_blank)

2，《论可计算数及其在判定问题上的应用》的前言，第一，二章部分译文：

“可计算数”简单说是，其十进制的表达用有限的手段可计算的实数。虽然本文的主题表面上讲可计算数，然而几乎可以同样容易定义和研究变量为整数或实数或可计算变量的可计算函数，可计算谓词等。在每种情况下，基本的问题是一样的，我选择可计算数来解释，是因为这样可以涉及最少的技术细节。不久我希望给出可计算数与可计算函数等之间的关系，这将包括用可计算数表达的实数变量的函数理论。按照我的定义，一个数是可计算的，如果它的十进制的表达能被机器写下来。

在§9,10，我给出一些论点，想指出可计算数包括所有的能自然看作可计算的数。比如，它们包括代数数的实数部分，Bessel函数的大小的实数部分，PI, e等等。然而可计算数并不包括所有可定义的数。

尽管可计算数类如此之大，在许多方面类似于实数类，它仍然是可枚举。在§8章，我调查某些证明似乎证明相反的论点。通过正确应用一个论点，可达成一个与哥德尔的结论表面上相似的结论。这些结果可有价值的应用，尤其是，指出（§11）Hilbertian Entscheidungsproblem没有解。

The “computable” numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable numbers, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbrous technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

In §§ 9, 10 I give some arguments with the intention of showing that the computable numbers include all numbers which could naturally be regarded as computable. In particular, I show that certain large classes of numbers are computable. They include, for instance, the real parts of all algebraic numbers, the real parts of the zeros of the Bessel functions, the numbers X, e, etc. The computable numbers do not, however, include all definable numbers, and an example is given of a definable number which is not computable.

Although the class of computable numbers is so great, and in many ways similar to the class of real numbers, it is nevertheless enumerable. In §8 I examine certain arguments which would seem to prove the contrary. By the correct application of one of these arguments, conclusions are reached which are superficially similar to those of Gödel [1] . These results {231}  have valuable applications. In particular, it is shown (§11) that the Hilbertian Entscheidungsproblem can have no solution.

1，计算机器（Computing machine）

我们已经说过，可计算数是那些可用有限的手段计算而得的表达为十进制的数，这需要较明确的定义。在第九章之前，不对定义作真正合理化的说明，目前我仅说理由是人类的记忆需要限制。

可将一个在计算实数的人与某机器比较，此机器具有有限数目的状态：q1;q2;…qk（m-格局，m-configurations）。给机器提供一条纸带（类似纸张），机器在上面运行，纸带被分为段（方格，squares），每个方格上放置一个符号。在任何时刻，只有一个方格，比如第r个方格，放置符号了S（r），“在机器里”，我们可以称此方格为“扫描方格”，“扫描方格”里的符号称为“扫描符号”，“扫描符号”是机器“直接感知”的唯一符号。然而，通过变化格局，机器可以有效记忆已经“见过（扫描）”的符号。机器在任何时刻可能的行为由m－格局qn，扫描符号S（r）决定，这对（qn，S（r））称为“格局”：于是，格局决定机器可能的行为。在有些格局，扫描方格是空的（即没有符号），机器就在扫描方格上写下一个新的符号；在别的格局，它擦掉扫描符号。机器也可能改变正在扫描的方格，但是仅仅向左右移动一个方格。此外，对任何一个这样的操作，m－格局可能变化。某些写下的符号将形成数字序列（ sequence of figures），它是正在计算的表达为十进制的实数，别 的符号只是“辅助记忆”的粗糙纪录，只有那些粗糙纪录可能被擦掉。

我的观点是，这些操作包括了所有用在计算一个数时所需的操作，为此论点辩护要在读者较熟悉机器理论后才会较容易。在下一节，我将着手发展此理论，假设读者已经知道“机器”，“纸带”，“扫描”等。

1. Computing machines.

We have said that the computable numbers are those whose decimals are calculable by finite means. This requires rather more explicit definition. No real attempt will be made to justify the definitions given until we reach §9. For the present I shall only say that the justification lies in the fact that the human memory is necessarily limited.

We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions q1, q2, ..., qR which will be called “m-configurations”. The machine is supplied with a “tape”, (the analogue of paper) running through it, and divided into sections (called “squares”) each capable of bearing a “symbol”. At any moment there is just one square, say the r-th, bearing the symbol S(r) which is “in the machine”. We may call this square the “scanned square”. The symbol on the scanned square may be called the “scanned symbol”. The “scanned symbol” is the only one of which the machine is, so to speak, “directly aware”. However, by altering its m-configuration the machine can effectively remember some of the symbols which it has “seen” (scanned) previously. The possible behaviour of the machine at any moment is determined by the m-configuration qn and the scanned symbol S(r). This pair qn, S(r) will be called the “configuration”: thus the configuration determines the possible behaviour of the machine. In some of the configurations in which the scanned square is blank (i.e. bears no symbol) the machine writes down a new symbol on the scanned square: in other configurations it erases the scanned symbol. The machine may also change the square which is being scanned, but only by shifting it one place to right or 1eft. In addition to any of these operations the m-configuration may be changed. Some of the symbols written down {232} will form the sequence of figures which is the decimal of the real number which is being computed. The others are just rough notes to “assist the memory”. It will only be these rough notes which will be liable to erasure.

It is my contention that these operations include all those which are used in the computation of a number. The defence of this contention will be easier when the theory of the machines is familiar to the reader. In the next section I therefore proceed with the development of the theory and assume that it is understood what is meant by “machine”, “tape”, “scanned”, etc.

2，定义

自动机（Automatic machines）

如果每个阶段机器的运动（在第一章的意义上）完全由格局确定，我们称此机器为«自动机»（a-machine）。

为了某种目的，我们可能使用一些机器（choice machines or c-machines），它的运动部分由格局决定。当这样的机器到达一个这样模棱两可的格局时，只有当外界的操作做某个任意的选择，它才能继续运行。如果我们用机器处理公理系统时会出现这样的情况，在本文中，我只处理自动机，因此往往忽略前缀a-。

计算机器（Computing machines）

如果一台机器打印两类符号，第一类（称为数字）全是0和1，其它被称为第二类符号，则机器将被称为“计算机器”。如果给机器装置一条空白纸带，让它运动起来，从正确的初始m－格局出发，机器打印的第一类符号的子序列称作“机器计算的序列”；在表达为二进制的十进制实数前放上小数点，称作“机器计算的数”。

在机器运动的任何阶段，扫描方格的数，在纸带上所有符号的完整序列，及m－格局，被说成是描述那时刻的“完全格局”。机器和纸带在一系列完整格局之间的变化被称作“机器的运动”。

循环和非循环机器（Circular and circle-free machines）

如果计算机不再写下有限数目的第一类符号，被称作“循环（Circular）”；否则，被称作“非循环（circle-free）”。

如果一台机器达到一个格局，从此不再运动，或者即使继续运动，只能打印第二类符号，而不能打印任何第一类符号，此机器则是“circular”。“circle-free”的意义将在第8章解释。

可计算序列和可计算数（Computable sequences and numbers）

一个序列被说成“可计算的”，如果能够通过一台“circle-free machine”计算而得。一个数是可计算的，如果它与由“circle-free machine”计算的数只差一个整数。

我们应该避免混淆，说可计算序列比说可计算数更经常。

2. Definitions.

Automatic machines.

If at each stage the motion of a machine (in the sense of §1) is completely determined by the configuration, we shall call the machine an “automatic machine” (or a-machine). For some purposes we might use machines (choice machines or c-machines) whose motion is only partially determined by the configuration (hence the use of the word “possible” in §1). When such a machine reaches one of these ambiguous configurations, it cannot go on until some arbitrary choice has been made by an external operator. This would be the case if we were using machines to deal with axiomatic systems. In this paper I deal only with automatic machines, and will therefore often omit the prefix a-.

Computing machines.

If an a-machine prints two kinds of symbols, of which the first kind (called figures) consists entirely of 0 and 1 (the others being called symbols of the second kind), then the machine will be called a computing machine. If the machine is supplied with a blank tape and set in motion, starting from the correct initial m-configuration, the subsequence of the symbols printed by it which are of the first kind will be called the sequence computed by the machine. The real number whose expression as a binary decimal is obtained by prefacing this sequence by a decimal point is called the number computed by the machine.

At any stage of the motion of the machine, the number of the scanned square, the complete sequence of all symbols on the tape, and the m-configuration will be said to describe the complete configuration at that stage. The changes of the machine and tape between successive complete configurations will be called the moves of the machine.{233}

Circular and circle-free machines.

If a computing machine never writes down more than a finite number of symbols of the first kind it will be called circular. Otherwise it is said to be circle-free.

A machine will be circular if it reaches a configuration from which there is no possible move, or if it goes on moving, and possibly printing symbols of the second kind, but cannot print any more symbols of the first kind. The significance of the term “circular” will be explained in §8.

Computable sequences and numbers.

A sequence is said to be computable if it can be computed by a circle-free machine. A number is computable if it differs by an integer from the number computed by a circle- free machine.

We shall avoid confusion by speaking more often of computable sequences than of computable numbers.

希尔伯特第十问题：求出一个整数系数方程的整数根，称为丢番图（约210-290，古希腊数学家）方程可解。1950年前后，美国数学家戴维斯（Davis）、普特南（Putnan）、罗宾逊（Robinson）等取得关键性突破。1970年，巴克尔（Baker）、费罗斯（Philos）对含两个未知数的方程取得肯定结论。1970年。苏联数学家马蒂塞维奇最终证明：在一般情况下，答案是否定的。虽然得出了否定的结果，却产生了一系列很有价值的副产品，其中不少和计算机科学有密切联系。

一般递归函数(general recursive function)亦称**递归函数**，是指一类具有能行可计算的全数论函数。不仅如此，现在一般认为，能行可计算的全数论函数恰好就是一般递归函数，一般递归函数的概念最初是由美籍奥地利数学家[哥德尔](https://baike.baidu.com/item/%E5%93%A5%E5%BE%B7%E5%B0%94)于1934年定义的，也就是现在所谓的埃尔布朗-哥德尔可计算函数，即若一个[数论函数](https://baike.baidu.com/item/%E6%95%B0%E8%AE%BA%E5%87%BD%E6%95%B0" \t "_blank)可由某个等式系ε定义，则哥德尔称f为一般递归的。1936年，美国逻辑学家、数学家[克林](https://baike.baidu.com/item/%E5%85%8B%E6%9E%97/3755" \t "_blank)(S.C.Kleene)引进了μ递归函数的概念，并进而证明了它恰好与哥德尔的一般递归函数类一致。此后，一般递归函数的概念便经常用μ递归的形式给出。[莫绍揆](https://baike.baidu.com/item/%E8%8E%AB%E7%BB%8D%E6%8F%86" \t "_blank)于1965年利用一般递归式的概念提出了一般递归函数的另一种等价定义形式：从本原函数出发，经叠置和一般递归式作用而生成的函数称为一般递归式。注意到，比起等式系或μ算子来，一般递归式更好地体现了一般递归的意义，因此，利用一般递归式引入一般递归函数概念当是最为合理的。而且也是原始递归函数概念的一种自然推广。此外，一般递归函数恰好是部分递归函数中的全函数，不过与全体部分递归函数不一样，全体一般递归函数不是能行可列的 [1]  。

中文名

一般递归函数

外文名

general recursive function

别    名

递归函数

属    性

一类具有能行可计算的全数论函数

相关概念

原始递归函数

提出者

哥德尔

**目录**

1. 1 [发展简要](https://baike.baidu.com/item/%E4%B8%80%E8%88%AC%E9%80%92%E5%BD%92%E5%87%BD%E6%95%B0/19056949?fr=aladdin#1)
2. 2 [基本介绍](https://baike.baidu.com/item/%E4%B8%80%E8%88%AC%E9%80%92%E5%BD%92%E5%87%BD%E6%95%B0/19056949?fr=aladdin#2)

**发展简要**

[编辑](javascript:;)

一般递归函数(gereral recursive function)是[原始递归函数](https://baike.baidu.com/item/%E5%8E%9F%E5%A7%8B%E9%80%92%E5%BD%92%E5%87%BD%E6%95%B0)的推广。二十世纪初，为了一般地定义能行过程便引进了一般递归函数的概念。这个概念最初是[哥德尔](https://baike.baidu.com/item/%E5%93%A5%E5%BE%B7%E5%B0%94)在厄尔勃朗的信的启示下提出来的。后来，[克林](https://baike.baidu.com/item/%E5%85%8B%E6%9E%97/3755)改进了哥德尔的定义并发展了一般递归函数的理论。在此之前，邱吉引进了

https://gss3.bdstatic.com/-Po3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D12/sign=a217383bc2177f3e1434f80f72cf6ecf/1e30e924b899a9013fd7f1bb16950a7b0308f559.jpg

记号，定义了

https://gss3.bdstatic.com/-Po3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D12/sign=a217383bc2177f3e1434f80f72cf6ecf/1e30e924b899a9013fd7f1bb16950a7b0308f559.jpg

可定义函数的概念。以后，波斯特和图灵又引进了图灵机器并定义了可计算函数的概念。后来证明所有这些概念都与一般递归函数等价。由此可见，一般递归函数是非常重要的 [2]  。一般递归函数集包含原始递归函数集为其真子集。

**基本介绍**

[编辑](javascript:;)

凡原始递归函数都是一般递归函数，反之不然。要找一个非原始递归函数的一般递归函数并不是一件容易的事。这就说明日常遇到的数论函数几乎都是原始递归函数。第一个找出非原始递归函数的例子的是德国数学家阿克曼，从而证实了一般递归函数的确比原始递归函数为广。

常见的一般递归函数的定义有两种，第一种都是在原始递归函数的定义中加进另一种则使用一般递归式，摹状式而得。

一般递归式通常写为：

https://gss0.bdstatic.com/-4o3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D424/sign=1f213fec04b30f24319aed01fc94d192/377adab44aed2e73ef7c12a18c01a18b86d6faa2.jpg

其中，

https://gss3.bdstatic.com/7Po3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D13/sign=88c0b062a2ec8a13101a53e3f50380d8/e4dde71190ef76c678036fa69616fdfaae51676e.jpg

为归宿于零的函数，即从任何一值(作为第一变元)经过有限次迭置之后

https://gss1.bdstatic.com/-vo3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D13/sign=d82a5f47a451f3dec7b2bd6795ee39a9/83025aafa40f4bfbbed109d9084f78f0f736183d.jpg

的值必为零。

摹状式一般写为：

https://gss1.bdstatic.com/-vo3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D260/sign=bc8f0b518113632711edc535a18ea056/c8ea15ce36d3d5391b50984e3187e950342ab0ca.jpg

但要求对任何

https://gss3.bdstatic.com/-Po3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D55/sign=90a9ae8e8c44ebf86971643ad8f9ff39/d009b3de9c82d1583776ce3c8b0a19d8bd3e4243.jpg

都有

https://gss0.bdstatic.com/94o3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D13/sign=a6b0283fb2014a90853e42bea877a615/5243fbf2b211931308d9683d6e380cd790238da1.jpg

使得

https://gss3.bdstatic.com/7Po3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D187/sign=82d3ae6a9a2397ddd2799c0c6e80b216/f2deb48f8c5494ee17f4989426f5e0fe98257e6d.jpg

其中，

https://gss0.bdstatic.com/-4o3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D17/sign=1e84e95025738bd4c021b636a08bb9a0/b3b7d0a20cf431ad0fdfd5b44036acaf2edd982a.jpg

表示满足(1)的最小自然数

https://gss2.bdstatic.com/-fo3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D13/sign=62e3956c242eb938e86d7ef1d46227f1/d50735fae6cd7b89a855ac0e042442a7d9330e05.jpg

。

已经证明这两个定义是等价的。邱吉建议把一般递归函数考虑为能行可计算函数的数学定义。这个论断被称为**邱吉论题**。由于可计算函数是个无定义概念，所以，邱吉论题是无法证明的。但是，根据种种理由，大家都相信它是正确的 。这样，凡是有一个计算过程或一个算法可以将函数值计算出来的函数便都是一般递归函数。利用这一结果解决了许多悬而未决的判定问题，重新研究了传统的描述集合论，阐明了半直觉主义数学中所使用的能行性概念，促进了新兴的计算机科学的发展