可计算性与计算复杂性
Computability and Computational Complexity

主讲：金人超

华中科技大学计算机科学与技术学院
jrc@hust.edu.cn

联系方式：

- QQ: 281381994
- E_mail: jrc@hust.edu.cn
- 电话：13349945613
- 办公室：医学图像信息研究中心
  (东校区)

学习本课程的重要性与必要性

- 理论上了解计算机的强大之处和局限所在。了解计算机的前世今生，思考计算机的未来与永恒，超越时代。
- 理论是技术的基础。提高我们对程序的审美，提升算法的品味与优雅，帮助我们在现实中开发更加智能、速度更快、更安全的计算机系统。

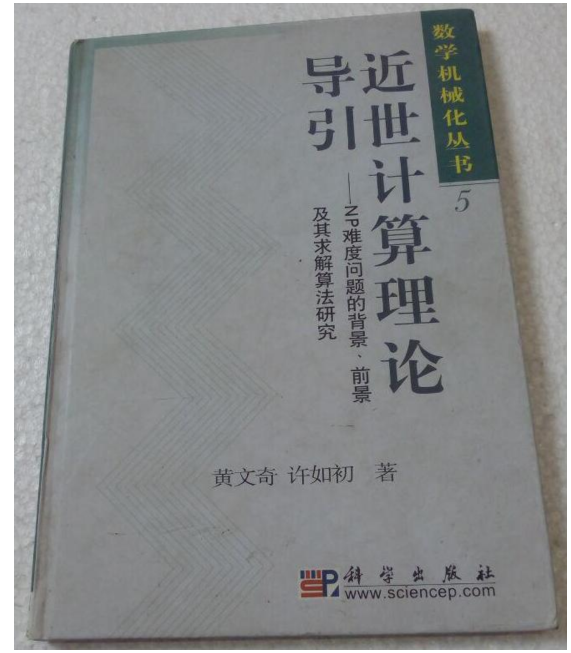
在这门课程里，我们将讨论以下问题：

- 什么是计算机的计算？
- 有没有计算机无法解决的问题？
- 有没有计算机无法快速解决的问题？
- 有没有计算机甚至无法近似地快速解决的问题？
- 如何开发高效对付这些问题的算法？

教材：

近世计算理论导引：**NP**难问题的背景、前景及其求解算法研究. 黄文奇、许如初著，北京：科学出版社，**2004**

（数学机械化丛书；**5**）



参考资料：

http://www.introtcs.org/public/index.html

http://stellar.mit.edu/S/course/6/sp16/6.045/

https://cn.udacity.com/course/computability-complexity-algorithms--ud061

第一章 计算的数学模型
——图灵（Turing）机

- 什么是计算？
- 什么是计算机？
- 计算的数学模型
  图灵机
- **Alan Turing（1912-1954）**

Alan Turing
（1912-1954）

图灵是一位非凡的数学家，计算机科学的先驱者，破译纳粹的著名密码的关键人物。在人工智能领域，图灵具有两项重要贡献：图灵检验法和图灵机。

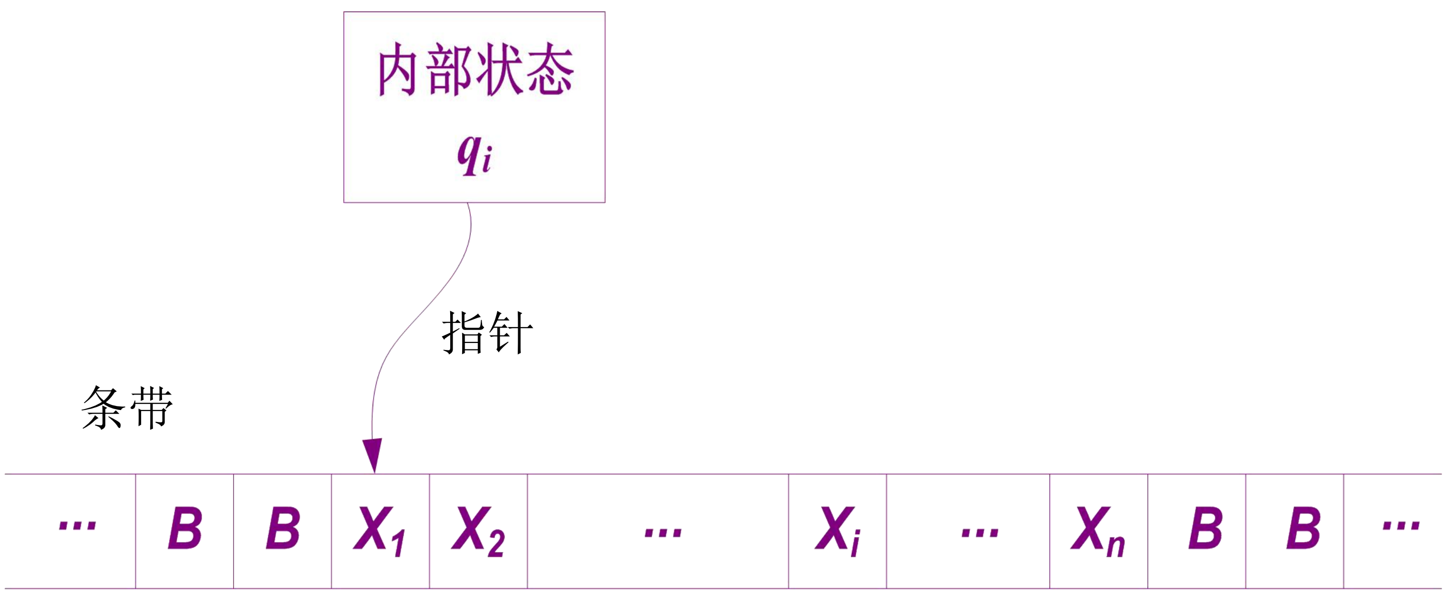
第一章 计算的数学模型
——图灵（Turing）机

- 在20世纪30年代，图灵在英国剑桥大学，成为了英国皇家学院的一名研究员。
- 他受到当时物理学革命性发展的影响。由于量子力学中观察者总是影响观察结果的原理，使哲学界发生了大混乱，该发展推翻了因果律和决定论的传统观念。
- 图灵被引向数学，因为看来数学所涉及的是绝对的实体，与观察者无关。
- 图灵致力于解答一个切中数学实体性核心的问题：是否有一机械的方式确定数学中的任何已知的语句是正确的还是错误的？为了回答这一问题，1936年，他提出了图灵计算机的概念。10年后，1946年，世界上第一台电子计算机ENIAC诞生。
- 图灵奖：1966年，ACM决定设立“图灵奖”——奖给计算机科学中最杰出的科学家，一般每年一名。

§1 Turing机的定义及其直观形象

硬件:

- 一条被划分为方格的双向无穷延伸的条带，每个方格内有一个符号（空白也看作是一个符号: ***B***）；
- 一根指针（可沿条带左右移动，可读写条带）；



§1 Turing机的定义及其直观形象

软件:

- 由有穷个字母构成的字母表 $\{S_0,S_1,\ldots,S_n\}$；
- 由有穷个内部状态构成的内部状态集$\{q_1,\cdots,q_m\}$ ；
- 由有穷条规则（指令）构成的规则集合——“程序”。

Turing机的行为规则（指令）只有如下三种类型，其形式为四重组。

- $q_i S_j S_k q_l$；
- $q_i S_j L q_l$；　　$0 \le j,k \le n \quad 1 \le i, l \le m$
- $q_i S_j R q_l$；

§1 Turing机的定义及其直观形象

初始时刻，**Turing**机带上的有穷个方格中分别给写上了字母表$\{S_1,S_2,\ldots,S_n\}$中的某一个符号，其它方格均为空白，记作B或S_0;

指针指着最左边一个非S_0方格的左边一格;

机器处于内部状态q_1;

Turing机往后的行为动作由“程序”所指挥。

§1 Turing机的定义及其直观形象

- **Turing机如何执行"程序"？**

1.想清自己当前的内部状态q_i，

2.再看清当地的外部环境，即指针现在所指的方格上的字符S_j，

3.然后再在行为规则集中查，看那一条规则是适用的，即那一条规则的打头二字为q_iS_j。

4.如查到了，就按那一条规则的指示进行动作；如查不着适用的规则就停机。

§1 Turing机的定义及其直观形象

- 为了使**Turing**机在每一时刻都能确切地知道如何动作，而不致出现模棱两可的情形，指挥**Turing**机动作的四重组集应该满足一个约束条件：即四重组集中的任何两个四重组其打头的两个符号不能完全相同。可称这个条件为协调条件。

- 将字母表及内部状态表以及一个满足协调条件的四重组集联合在一起称为一部**Turing**机。

§1 Turing机的定义及其直观形象

- **观察如下实现函数$f(x)=x+2$的计算的Turing机，其中空格被记作B，S_1被记作1。**
- **字母表{1}，内部状态集$\{q_1,\ldots,q_5\}$， 行为准则集("程序")为**

$$\left\{\begin{array}{llll} q_1 & B & R & q_2 \\ q_2 & 1 & R & q_2 \\ q_2 & B & 1 & q_3 \\ q_3 & 1 & R & q_3 \\ q_3 & B & 1 & q_4 \\ q_4 & 1 & L & q_4 \end{array}\right\}$$

……进入计算
……右移找尾
……找到尾后进入加工态，先加1
……右移找B
……再加1，进入回头状态
……找左头

§1 Turing机的定义及其直观形象

- 再观察一个永不停机的**Turing**机。

字母表：**{1}**，内部状态集：$\{q_1, q_2\}$， 行为准则集为：

$$\left\{\begin{array}{l} q_1\ B\ R\ q_2 \\ q_1\ 1\ R\ q_2 \\ q_2\ B\ L\ q_1 \\ q_2\ 1\ L\ q_1 \end{array}\right\} \begin{array}{l} \cdots\cdots q_1\text{右移}q_2 \\ \cdots\cdots q_2\text{右移}q_1 \end{array}$$

练习：构造一个计算$f(x)=x+1$的图灵机，x为正整数，用二进制数表示。写在纸上，用手机拍了发在群里，立刻完成。

§1 Turing机的定义及其直观形象

- 格局（瞬像）

  当前带上所有符号;

  当前指针位置;

  当前内部状态。
- 初始格局及图灵机的输入
- 终止格局(停机格局)及图灵机的输出
- 计算：是一个格局的有穷序列 $c_1,c_2,\ldots c_m$，其中c_1为初始格局，c_m为终止格局，c_i到c_{i+1}的变化符合“程序”的某条指令。$i=1,2,\ldots,m-1$

§2 Turing机所计算的函数和所接受的语言、计算复杂度

2.1 Turing机所计算的m元（$m\geq 1$）函数

设有字母表$A=\{S_1,\ldots,S_n\}$，A^i为字母表A上所有长度为i的字符串（字）的集合，A^*为字母表A上所有有限长度字符串(字)的集合，即

$$A^* = A^0 \cup A^1 \cup A^2 \cup A^3 \cup \cdots$$

考虑从$A^*\times A^*\times \ldots\times A^*$到$A^*$的部分函数$f$，即函数$f$在$A^*\times A^*\times \ldots\times A^*$的某个子集上有定义，而在其补集上无定义（或者说定义为↑）。全函数是在$A^*\times A^*\times \ldots\times A^*$上处处有定义的部分函数。

例如$A=\{a,b\}$, $m=2$，$f(a,ab)=aab$, $f(ab,a)=aabb$ …

§2 Turing机所计算的函数和所接受的语言、计算复杂度

设f是如前所述的一个m元部分函数。若有Turing机T，对任意$(x_1,\cdots,x_m)\in(A^*)^m$，若$f(x_1,x_2,\ldots,x_m)=y$，则$T$以$x_1,x_2,\ldots,x_m$为输入开始计算，最终停机，纸带上输出$y$; 若$f(x_1,x_2,\ldots,x_m)$无定义，则$T$以$x_1,x_2,\ldots,x_m$为输入最终永不停机。则我们称$f$为**Turing机$T$所计算的$m$元（部分）函数**，或称**Turing机$T$计算了$m$元（部分）函数$f$。**

对任意一个m元部分函数f，如果存在一台图灵机计算它，则称它是**图灵可计算的**，否则就不是。

§2 Turing机所计算的函数和所接受的语言、计算复杂度

注意：任意一台图灵机，既可以计算一个1元函数，也可以计算一个2元函数，3元函数，…

问题1：一个m元部分函数，如果能被一台图灵机所计算，则一定能被无穷台图灵机所计算。为什么？

问题2: 处处无定义的函数是不是图灵可计算的？

问题3: 全函数一定是图灵可计算的吗？

§2 Turing机所计算的函数和所接受的语言、计算复杂度

问题4：如果f和g分别可以被图灵机计算，而且可以复合。那么它们的复合函数是否也可以被图灵机所计算？如何得到计算复合函数的图灵机?

问题5：想想你能想到的函数，它们是否是图灵可计算的?

问题6：这里只讨论字符串函数，对那些非字符串函数怎么办？比如常见的整数函数，实数函数？甚至图像、声音?

- 提问？

§2 Turing机所计算的函数和所接受的语言、计算复杂度

2.2 Turing机所接受的语言

定义1 语言 L是字符表A上的某些字（有穷字符串）所构成的集合。即$L \subseteq A^*$

问题1：给定A^*后，有多少个不同的语言L?

定义2 设Turing机T，对于某个字u，如果从以u为输入的初始格局开始，T最终会停机，就称T接受字u。T所接受的A^*中的所有字所构成的集合是T所接受的语言。

问题2：图灵机T所接受的语言与T所计算的一元部分函数之间的关系是什么样的?

思考：每个语言都可被某个图灵机接受吗?

§2 Turing机所计算的函数和所接受的语言、计算复杂度

2.3 计算复杂度

时间复杂度、空间复杂度，统称为**时空开销**。

时间复杂度定义为

$$T(x_1,\cdots,x_m):(A^*)^m \to N\cup\{\uparrow\}$$

$$T(x_1,\cdots,x_m)=\begin{cases}\text{执行指令次数}, & \text{若} f(x_1,\cdots,x_m)\downarrow\\ \uparrow, & \text{若} f(x_1,\cdots,x_m)\uparrow\end{cases}$$

§2 Turing机所计算的函数和所接受的语言、计算复杂度

空间复杂度：设f为图灵机M所计算的m元函数，若 $f(x_1,x_2,\ldots,x_m)$有定义，将M对输入$x_1,x_2,\ldots,x_m$的计算过程中读写头所"注视"过的最左边的格子和最右边的格子之间的纸带上的部分，称为工作部分，这个"工作部分"所拥有的格子数称为M计算 $f(x_1,x_2,\ldots,x_m)$的空间复杂度。

注意：格子可重复利用。

$$S(x_1,\cdots,x_m):(A^*)^m \to N \cup \{\uparrow\}$$

$$S(x_1,\cdots,x_m)=\begin{cases}\text{工作部分的格子数，若}f(x_1,\cdots,x_m)\downarrow \\ \uparrow, \qquad\qquad \text{若}f(x_1,\cdots,x_m)\uparrow\end{cases}$$

§2 Turing机所计算的函数和所接受的语言、计算复杂度

- 思考：

图灵机模型下的时空开销是否能反映现实计算机上的时空开销？

真实的时间开销以时、分、秒为单位，而图灵机的时间开销以计算步数为单位，这样合适吗？

时间复杂度的大小和空间复杂度的大小之间有什么联系？

§3 Church-Turing论题

Church-Turing Thesis: 只要世上有一台机器能够实现对部分（或全）函数*f*的计算，则一定存在一台Turing机，它能够实现对*f*的计算。

一句话：图灵可计算当且仅当（机器）可计算。

注1。这里的“一台机器”也可能是由一支笔、一张纸及一个严格地按一套计算规则进行计算的人所构成的。

注2 。**Church－Turing**论题是不可能被证明的。即无法从理论上排除存在着某种装置，用它能计算**Turing**机无法计算的函数。但是迄今为止，任何已发现的具体的直观的“可计算函数”，都是能被**Turing**机所计算的。

§3 Church-Turing论题

注**3**。我们接受**church-Turing**论题。如果我们能用某种机器（比如说用**C**语言编写的程序）计算某个（部分/全）函数，则我们就直接说我们能用图灵机计算它，而不去真的要求谁构造一个图灵机来证实这一点。当然，如果不怕麻烦的话，所要求的图灵机是可以构造出来的，但显然没有什么实用价值。

问题:

一台当今最强大的计算机所计算的所有函数一定也能被**Turing**机所计算吗？那么集群并行机呢？**GPU**阵列呢？图灵机能进行图像处理吗？

§3 Church-Turing论题

20世纪30年代到40年代，数理逻辑学家相继提出了四种计算模型 ：

图灵机(A.Turing, 1936)

递归函数（K.Godel和S.C.Kleene，1936）

λ演算（A.Church，1935）

波斯特系统（E.Post，1936 ）

可以证明，这些模型在可计算性上是等价的。即：

直观可计算=图灵可计算=递归= λ可定义=波斯特可计算

§3 Church-Turing论题

除了这里介绍的4元组图灵机外，还有3元组、5元组图灵机，以及单向、双带、多带等图灵机的各种变形。

从可计算性角度来讲，能够证明多数图灵机和这里介绍的图灵机是功能上等价的。也有少数是功能受限制的，但没有本质上功能更强大的。

§4 Turing机的编码

通用图灵机

在图灵1936 年的论文《On Computable Numbers》（从某种意义上说，计算机科学的创始文件）中，图灵证明了：我们可以构建一个图灵机 U，作为其他图灵机的解释器。

换句话说，U 的输入纸带可以包含另一个图灵机的描述，然后逐步模拟。这样的机器 U 称为**通用图灵机**。

因为有了通用机器，我们每次想要解决新问题时就不再需要构建新硬件，而只需写出一个新软件。因此有人将图灵的这一普遍性结果称为“**软件行业存在引理**”！

§4 Turing机的编码

- 字符串的编码

给定字母表A，建立一个A^*到N的一一对应关系. $N=\{0,1,2,3\ldots\}$

这种对应是可计算的（“能行的”）。即：存在一个程序，对任意一个字符串，计算出字符串的序号；存在另一个程序，对任意一个序号i，计算出第i个字符串。

前面所讨论的m元部分函数都是字符串函数，现在每一个字符串函数都可“编码”成一个数论函数的版本：

$$f\colon\ N\times N\times\ldots\times N\rightarrow N\cup\{\uparrow\},$$

反之亦然。

每一个语言L都可“编码”成N的一个子集A，反之亦然。

所以我们也可以说图灵机T接受N的某个子集A。A正好就是T所计算的一元部分数论函数f的定义域。

§4 Turing机的编码

- 二元函数可以“编码”成一元函数,因为

$$N \times N \sim N$$

这里“~”表示存在可计算的一一对应函数。

- 对任意正整数m，m元函数皆可“编码”为一元函数。因为

$$N \times N \times \ldots \times N \sim N$$

这种“编码”和“解码”都是“能行的”。

因此以下我们只关注一元函数。

§4 Turing机的编码

- **Turing机的编码**

设T是全体图灵机的集合，建立T到N的一一对应关系。例如一台图灵机:

$$\left[\begin{array}{cc} \underset{\text{字符集}}{\{S_1, S_2\}};\underset{\text{内部状态}}{\{q_1, q_2, q_3\}} & \underset{\text{四重组集}}{\begin{pmatrix} q_2 & S_0 & S_1 & q_3 \\ q_3 & S_2 & L & q_2 \\ q_1 & S_1 & S_0 & q_2 \end{pmatrix}} \end{array}\right]$$

可用自然的方式写成字母表$\{S,q,0,1,\ldots,9,R,L,/\}$上的一个字：

$$S1S2//q1q2q3//q2S0S1q3/q3S2Lq2/q1S1S0q2$$

§4 Turing机的编码

- 由前面的讨论可知，字母表$A=\{S,q,0,1,\ldots,9,R,L,/\}$上的所有字的集合与$N$可以建立一一对应关系，所以可将$A$上所有字按顺序排列:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	……
×	×	○	×	○	○	○	×	×	×	○	○	○	○	×	×	○	○	×	×	×	○	×	×	×	○	○	
		0		1	2	3				4	5	6	7			8	9				10				11	12	

- 按序找到第0个符合图灵机“语法”的字（符合打o，不符合打×），编号为0，第1个编号为1，第2个编号为2，……,以此类推，可将所有的图灵机列出（语义上一致的图灵机是允许重复出现的）。

问题：试证明此编码和解码过程都是可计算的（“能行的”）。

§4 Turing机的编码

有了这种对应关系后，可将全体图灵机按序号列出：

$$T=\{T_0,T_1,T_2,\ldots,T_i,\ldots\}$$

每个图灵机计算一个一元部分可计算函数，所以，全体一元部分可计算函数的集合也可按此序号列出：

$$\Phi=\{\varphi_0, \varphi_1,\varphi_2,\ldots, \varphi_i,\ldots\}$$

思考：是否所有一元部分函数f：$N\to N\cup\{\uparrow\}$都被列出了？退一步问，是否所有的一元全函数f：$N\to N$都被列出了？

每个图灵机接受一个N的子集，全体可被图灵机接受的子集也可按此序号列出：

$$\mathrm{W}=\{w_0,w_1,w_2,\ldots,w_i,\ldots\}$$

w_i就是函数φ_i的定义域，被称为递归可枚举集。

思考：是否N的所有子集都被列出了？

§4 Turing机的编码

在递归论术语中，递归函数=可计算的全函数。

任一递归函数（可计算的全函数）φ_j的值域可枚举为

$\{\varphi_j(0),\varphi_j(1),\varphi_j(2)\ldots\} = \{\varphi_j(x)|x\in N\}$

或说用处处停机的图灵机T_j枚举为 $\{T_j(0),T_j(1),T_j(2)\ldots\}$。因此这种集合被称为递归可枚举集（**r.e.集**，**recursively enumerable set**）

r.e.集也被称为“能行可数集”，与普通可数集的区别在于：**r.e.**集是“可以用机器数”的。

可证明每个非空的w_i都是一个**r.e.**集，反之，每个**r.e.**集都是某个w_i，即：

非空集合A是某个部分可计算函数φ_i的定义域，当且仅当它是某个可计算的全函数φ_j的值域。即$\exists i, A=w_i$当且仅当$\exists j, A=\{\varphi_j(x)|x\in N\}$。

§4 Turing机的编码

- 每个N的子集A都有一个特征函数

$$\chi_A : N \to \{0,1\}$$

$$\chi_A(x) = \begin{cases} 1, & 若 x \in A, \\ 0, & 若 x \notin A. \end{cases}$$

称集合A是（图灵）可计算的（可判定的），当且仅当它的特征函数χ_A是（图灵）可计算的。

问题1 集合可计算与函数可计算这两个概念的区别和联系是什么？

问题2 可计算的集合与r.e.集是不是同一概念？它们之间的联系是什么？

§4 Turing机的编码

问题1 试证明:

- 若A是可计算的，则A一定是r.e.集。
- 若A是可计算的，则A的补集也是可计算的，A的补集也是r.e.集。

反之，若A是r.e.集，不能推出A是可计算集，也不能推出A的补集是r.e.集。但是，可以证明:

问题2试证明:

- 若A和A的补集都是r.e.集，则A是可计算集，A的补集也是可计算集。

思考：是否有不是可计算集的r.e.集？是否有r.e.集的补集不是r.e. 集？