第十章 NP 和 NP—完备性理论

10.1 NP－完备性理论概述

1971 年，S.A.Cook 发表了著名的论文：

“The complexity of theorem proving procedures”, Proc. 3rd ACM Symp. on the Theory of Computing, ACM(1971), pp. 151-1510.

成功地证明了第一个 NP 完备问题，从而为 NP 完备性理论奠定了基础，给计算复杂性理论开辟了一个新的领域。

由于这一卓越的成就，Cook 荣获了 1982 年度的图灵奖，这是当今世界上对计算机科学家的最高荣誉奖赏。

在这篇简洁又精致的文章中，Cook 做了几件重要的事情：

1. 他强调了“多项式时间可规约性”的重要意义。所谓多项式时间规约是指可以用多项式时间算法实现的所需要的变换规约。如果我们有从第一个问题到第二个问题的多项式时间规约，那么就一定能把第二个问题的任何多项式时间算法转换成第一个问题的多项式时间算法。
2. 他把注意力集中在判定问题的 NP 类上，这类问题可以用非确定型计算机在多项式时间内解决。（如果问题的解不是“是”就是“否”，则称这个问题是判定问题。在实际中遇到的表面上难解的问题，当把它们表成判定问题时，大多数都属于这一类。）
3. 他证明了 NP 中的一个名叫“可适定性”问题的具体问题具有这样的性质：NP 中的所有其它问题都可以多项式规约为这个问题。如果可适定性问题可以用多项式时间算法解决，那么 NP 中的所有问题也都可以用多项式时间算法解决。如果 NP 中的某个问题是难解的，那么可适定性问题也一定是难解的。所以：在某种意义下，可适定性问题是 NP 中“最难的”问题。
4. 他认为 NP 中的一些其它问题可能和可适定性问题一样，具有这种成为 NP 中“最难的”问题的性质。他证明了问题“给定图 G，它有包含 k 个顶点的完全子图吗？其中 k 是给定的自然数”，就是这种情况。

随后，Karp 证明了许多著名的组合问题，包括 TSP 问题的判定问题形式确实恰好与可适定性问题一样难。

以后，证明了各种各样的其它问题在难度上等价于这些问题，并给这个等价类起名：NP－完备类，它有 NP 中所有“最难的”问题组成。现在已经证明了几千个 NP－完备的各种不同的问题。

- P 与 NP 的问题已被公认为数学和理论计算机科学中至今尚未解决的最重要的问题之一；
- 尽管大多数研究工作者猜想 NP－完备问题是难解的，然而在证明或否定这个广泛的猜想方面几乎没有取得任何进展；
- 但是，即使没有证明 NP－完备问题蕴涵着难解性，知道一个问题是 NP－完备的，至少暗示着要想用多项式时间解决这个问题必须有重大突破。

10.2 判定问题与语言

开始我们用通俗的语言简单介绍了在计算复杂性研究中经常遇到的一些概念，并依此给出了算法复杂性的大致分类。有了这些基本的认识，下面将从逻辑与数学上给出有关概念的严格定义，对问题、算法的复杂性做进一步的分类，介绍在计算复杂性理论研究中经常用到的一些刻画问题求解难易程度的不同复杂性类型的概念、含义，并简要论述它们之间的相互关系。同上面一样，为了给出不同类型问题的准确定义，就先要对问题、算法等基本概念进行精确地刻画，故下面先对问题给出一个一般性的描述方法。

由于实际中问题的千变万化，不同的问题常常有着不同的表述方式。为了能在统一的框架下刻画、研究任何问题的可解性与求解它的复杂程度，并起到便于描述算法的定义以及相关的理论结果，我们就需要对上面所给出的关于问题的定义再进行抽象与统一化。目前所广泛采用的描述问题的方法主要有两种：一是将任一问题转化为所谓的可行性检验问题(feasibility problem), 二是把问题转述为判定问题（decision problem），我们将采用后一种术语来描述实际中的任何问题。

判定问题是答案只有是与非两种可能的问题。一个判定问题 Π 可简单地由其所有例子的集合 D_Π 和答案为是的例子所构成的子集 $Y_\Pi \subseteq D_\Pi$ 来组成。不过，为了反映实际问题所具有的特性，通常所采用的标准描述方法由两部分组成。第一部分用诸如集合、图、函数等各种描述分量来刻画判定问题的一般性例子，以下用“例子”来表示这一部分；第二个部分则陈述基于一般性例子所提出的一个是—非提问，以下简称这一部分为“问题”。因此，一个例子属于 D_Π 当且仅当它可通过用具有特定性质的某些对象来替代一般性例子中的所有一般性描述分量而得到，而一个例子属于 Y_Π 当且仅当限定于该例子时，它对所述提问的回答为“是”。

实际中几乎所有问题都可直接或间接地转述为判定问题，故这一转换在不失其一般性的同时，使得我们能用一通用的方式来描述各种问题。例如，与旅行商问题相对应的判定问题可描述如下：

例子 待访问城市的有限集合 $C=\{C_1,C_2,\cdots,C_m\}$、每对城市 $C_i,C_j\in C$ 之间的距离 $d(C_i,C_j)\in Z^+$（这里 Z^+ 表示正整数的集合，下同）及一个界 $B\in Z^+$。

问题 在 C 中存在一个总长不超过 B 的、通过所有城市的旅行路线吗？

也就是说，是否存在 C 的一个排序 $<C_{\pi(1)},C_{\pi(2)},\cdots,C_{\pi(m)}>$，使得

$$\sum_{i=1}^{m-1} d(C_{\pi(i)},C_{\pi(i+1)})+d(C_{\pi(m)},C_{\pi(1)})\le B.$$

这个例子还提示我们怎样将一个优化问题转化为判定问题。进而，若函数值相对容易计算，则所得判定问题不会比相应的优化问题更困难。鉴于优化问题的广泛存在性，这里简述一个类似于该例结论的，关于优化问题与其对应的判定问题之间的一般性关系的结论。它可指导我们去分析其它类型的问题与其相应的判定问题之间的关系。

一个最优化问题可看作是其所有实例的集合，而每一个实例为一元素对（F,c），其中 F 是可行解集合、c 是目标函数。对于一个最优化问题，我们可以提出下面三种不同的模式：

最优化模式：求最优的可行解。

求值模式：求最优值。

判定模式：给定一个实例（F,c）和一个整数 L，问是否存在一个可行解

$$f \in F\text{，使得 } c(f) \le \mathrm{L}?$$

这三种模式之间存在着密切的关系。

一方面，在求目标函数值 $c(f)$不太困难的假设下，以上三种模式每一个都不比前一个更困难。因此，如果对判定模式有效算法的存在性有否定的回答，则对其它模式的回答也是否定的。

另一方面，在非常一般和现实的假设下，即设最优值是一个整数，且这个整数或其绝对值的对数被输入长度的一个多项式所界定，则可用对半搜索的技术证明，只要判定模式存在多项式时间算法，则求值模式也存在多项式时间算法。但若已知求值模式的多项式时间算法，求最优化模式的解目前还没有通用的好方法。不过，一种变形的动态规划方法（称为递归算法）可适用于很多问题。对于这些问题而言，三种模式的求解复杂性至少相对于多项式时间算法的存在性来说是等价的。

虽然所有的判定问题均可用统一的形式来描述，但从数学上讲还不够严密，且不易基于它给出算法的严格定义。为克服这些不足，我们再引进一个与判定问题有着很自然的对应关系的另一个概念，这就是语言（language）。利用语言这一术语就可在数学上精确地研究计算复杂性理论。

字符

如果我们用计算机来解决某个图问题，通常不会把这个图的图片输入到计算机中，而是将问题的某个适当编码输入到计算机，通过一个字符序列来表示它，而这个字符序列字符取自某个固定的有限“字母表”Σ。例如，我们可以取Σ为 ASCII 码或{0,1}。

对于字符的任一有限集合Σ，用Σ^*表示由Σ中的字符所组成的所有有限长度字符串的集合。若 L 为Σ^*的一个子集，则称 L 为字符集Σ上的一个语言。例如，101,001,111,1010110｝就是字符集Σ＝{0,1}上的一个语言。

判定问题与语言之间的对应关系是通过编码策略来实现的。具体地，一旦选取了某个固定的字符集Σ，则对任一判定问题 Π及其任一编码策略 e，则 Π和 e 将会把Σ^*中的所有字符串划分为三个部分：那些不是 Π的例子的编码（每一编码对应于Σ^*中的一个字符串）、那些对 Π的回答为非的例子的编码和那些对 Π的答案为是的例子的编码。这最后一类编码正是我们要与 Π通过 e 来联系的语言，并且对应于 Π和 e，定义语言：

$\mathrm{L}[\Pi,e]=\{x \in \Sigma^*:\ \Sigma$为$e$所使用的字符集、而$x$为某个例子$I \in Y_{\Pi}$在$e$下的编码$\}$。

如果一个结论对语言 L[Π,e]成立，我们就说它在编码策略 e 之下对问题 Π 成立。我们的正式理论将通过这一方式应用到判定问题中去，也就是说，计算复杂性理论所直接考虑的是对语言或字符串集合的分析。由于这个与形式语言理论之间的相似性，形式语言和自动机理论中的许多结果在计算复杂性与理论计算机科学中很有用。鉴于判定问题与语言之间的对应关系，今后在许多情况下我们交替地使用（判定）问题、语言这两个术语而不加区分。这样做的另一个原因是：对于任何两个合理的编码策略 e 与 e'，某个性质要么对 L[Π,e]和 L[Π,e']均成立，要么对二者皆不成立。因此，只要仅考虑合理的编码策略，则可以直接说某个性

质对 Π成立或不成立，而不具体提及编码策略，并常常将 L[Π,e]简记为 L[Π]。

需要说明的是，每当我们撇开编码策略直接说判定问题 Π具有某个特性时，其隐含的意思是：如果有必要的话，我们就可以指定一合理的编码策略 e, 使该特性对 L[Π,e]成立。然而，当这样独立于编码策略讨论问题 Π时，就失去了对输入长度的具体确定办法，而我们需要像这样的一个参变元以便能确切表述时间复杂性的概念。为此，今后我们总是隐含假设对每个判定问题 Π, 均有一个不依赖于编码方式的函数 Length: $D_{\Pi}\to Z^{+}$。该函数的值将与从任一合理的编码策略所得的关于 Π的任一例子的输入长度多项式相关。所谓多项式相关，是指对于 Π的任一合理编码策略 e，都存在两个多项式 p 与 p'，使得如果 $I\in D_{\Pi}$ 且 x 为 I 在 e 下的编码，则有 Length[I] < p (| x |)且| x |≤p' (Length[I])，这里| x |表示字符串 x 的长度。易知 Π的任意两个合理编码策略将导出多项式相关的输入长度。故对于同一问题，可能存在许多这样的函数，而我们的结果将对满足上述条件的任一个函数 Length 成立。例如，对旅行商问题对应的判定问题的任一例子 I，我们就可以定义

$$\mathrm{Length}[I]=m+\lceil \lg B\rceil+\max\{\lceil \lg d(C_i,C_j)\rceil : C_i,C_j\}.$$

有了语言这个非常接近于计算机可识别的字符串的概念，就可以严格地来定义作用于它（从而相应的判定问题）上的算法的概念，并进而对问题进行分类。

10.3 算法的严格定义与 P 类问题

要给出算法的准确定义，就首先要选定一个可用于描述计算的计算模型。第一个给出这种模型的是英国数学家图灵，后人称他所提出的模型为图灵机。本质上讲，图灵机是一个具有序列存贮载体（常用一个带有无限多个方格的线性带表示）的、按照具体指令可完成左或右移动、放置标记、抹去标记及在计算终止时停机等四种基本操作的、用于描述算法的语言。它在原理上与我们现在用于和计算机交流的，更为复杂的各种程序语言同样有力。由于其简单性，图灵机及其各种变形已被广泛用于计算复杂性的理论研究。这里，我们先选择确定性单带图灵机（deterministic one-tape Turing machine，以下常称为确定性图灵机，并简记为 DTM）作为计算模型。

一个 DTM 是由一个有限状态控制器、一个读写头和一条双向的、具有无限多个带格的线性带所组成。

一个 DTM 程序（program）应详细规定下列信息：

(a) 带中所用字符的一个有限集合 Γ，它应包括输入字符表子集 $\Sigma\subset\Gamma$ 和一个特别的空白符号 $b\in\Gamma-\Sigma$。

(b) 一个有限状态集 Q，它包括初始状态 q_0 和两个特有的停机状态 q_Y 和 q_N。

(c) 一个转移函数 $\delta:(Q-\{q_Y,q_N\})\times\Gamma\to Q\times\Gamma\times\{l,r\}$。

这样一个程序的运行很简单，假设对 DTM 的输入为字符串 $x\in\Sigma^{*}$，则该字符串首先被一格一个字符地顺序存放在带格 1 到带格| x |中，所有其他的带格开始时存放的均为空白符 b。该程序从初始状态 q_0 开始它的运算，并且读写头先位于带格 1。然后计算按下述规则一步一步地进行。若当前状态 q 为 q_Y 或 q_N，则计算终止，且如果 $q=q_Y$，就回答是，否则回答非。若当前状态 $q\in Q-\{q_Y,q_N\}$，

且 $s \in \Gamma$ 为读写头当前扫描之带格中的字符，而转移函数此时对应的取值为 $\delta(q,s)=(q',s',\Delta)$，那么，该程序将执行这样的几个操作：读写头抹去当前带格中的 s，代之以 s'，同时，如果△ $=l$，则读写头左移一格；如果△$=r$，则就右移一格。最后，有限状态控制器将从状态 q 变到状态 q'。这样就完成了程序的一步计算，并为下一步计算做好了准备，除非已处于停机状态。

除运算速度可能比较慢以外，可以设计一个 DTM 程序来完成任何可在某一通常计算机上实现的任一计算。有了 DTM 这个计算模型以及定义于它上的程序的概念，就可以给出算法及其时间复杂性函数的严格定义。

首先，称一个带有输入字符表 Σ 的 DTM 程序 M 接受 $x \in \Sigma^*$，当且仅当它作用于输入 x 时停机于状态 q_Y。由程序 M 所识别的语言记为 $L_M=\{x \in \Sigma^*$：M 接受 $x\}$。注意到语言识别的这一定义并不要求 M对 Σ^*中的所有字符串均停机，而只需对 L_M 中的字符串停机即可。因此，若 $x \in \Sigma^*-L_M$，则 M 的计算要么停机于状态 q_N，要么永不停机。只有当一个 DTM 程序对定义于其输入字符表上的所有可能字符串均停机时，我们才称其为一个算法。相应地，称一个 DTM 程序 M 在编码策略 e 之下求解判定问题 Π，如果 M 对定义于其输入字符表上的所有输入字符串均停机且有 $L_M=L[\Pi,e]$。

一个 DTM 程序 M 对于输入 x 的计算所用的时间定义为该程序从开始直至进人停机状态为止所运行的步数。由此可给出时间复杂性函数的定义：对于一个对所有输入 $x \in \Sigma^*$均停机的 DTM 程序 M，定义其时间复杂性函数 T_M: $Z^+ \to Z^+$为

$T_M(n)=\max\{m:$存在一个 $x \in \Sigma^*$，$|x|=n$ 使得 M 对输入 x 的计算需要时间 $m\}$。

若存在一个多项式 p，使得对所有的 $n \in Z^+$，有 $T_M(n) \le p(n)$，则称程序 M 为一个多项式时间 DTM 程序。

至此，我们就可以给出称之为 P 类(the class P)的第一类重要语言的正式定义如下：

P＝{L：存在一个多项式时间 DTM 程序 M，使得 $L=L_M$}。

若存在一个多项式时间 DTM 程序，它在编码策略 e 之下求解 Π，即 $L[\Pi,e] \in$ P，则称该判定问题 Π属于 P。鉴于合理编码策略之间的多项式时间等价性，我们将常常省略具体的合理编码策略，只简单地说某判定问题 Π属于 P。

类似地，所有现实的计算（机）模型相对于多项式时间是等价的，这里所给 P 类语言的定义可以按照适合于任一种现实计算(机)模型的程序去重新描述，并必然导致相同类型的语言。因此，以后我们将不再提及这里所用的计算模型 DTM，而只说多项式时间算法。进而，再考虑到上述关于判定问题与其在任一合理编码策略下的语言之间的相互关系，我们今后将常常按照通常的习惯做法，直接就判定问题，具体地，在其任一例子或相关分量上，而不是它们的某个编码描述上，以几乎独立于具体计算模型的方式去讨论该判定问题及其可能存在的不同类型求解算法。

10.4 NP 类问题

不难看出，上面所定义的 P 类语言只能用来描述那些存在有效算法的问题，即那些在确定性图灵机上，从而在任一合理的计算模型上可在多项式时间内获得

解决的问题。然而，在实际中存在许多别的重要问题，对于它们，至今尚未找到有效的求解算法。为了描述、分析这样的一类问题，我们需要借助于另一类图灵机作为计算模型。在引进正式的定义之前，我们先通过一个例子来说明这类问题的共同特点，从而解释相应图灵机模型产生的原因。

考虑旅行商问题对应的判定问题，即给定城市的一个集合，每两个城市之间的距离以及一个界 B，我们问是否存在一个总长不超过 B 的、通过所有城市的一个环游。对于这一问题，到现在还没有一个能真正求解它的多项式时间算法。考虑该问题的一个特定例子 *I*,假定某人声称对这个例子的答案应为是。如果我们有些怀疑，就会要求他给我们提供一个满足所要求特性的环游来证明他的断言。所幸的是，要证明他的断言的真假性对我们来说却是一件简单的事情。我们首先检验他所提供之答案是否确实为一个环游，即检查它是否包括所有的城市，且每个城市仅经过一次。如果是，就计算该环游的长度并将其与所给的界 B 进行比较以得出最终的结论。进而，我们很容易设计出一个时间复杂性为 Length[*I*]的多项式的一般算法来完成上述论证或检验过程。

许多别的判定问题都具有这种多项式时间可验证性。就是说，对于相应判定问题的任一个例子，一旦我们通过某种办法给出了其答案的一个猜测或估计，就能设计出一个多项式时间算法来验证其真实性。需要说明的是，多项式时间可验证性并不意味着多项式时间可解性。因我们并没有考虑为了找出解的一个猜测所花去的时间，而这常常需要我们在可能有指数多个猜测的集合中去选取一个较合理的猜测。现引人另一类图灵机以便刻画这一过程。

非确定性单带图灵机（non-deterministic one-tape Turing machine，常称为非确定性图灵机，并简记为 NDTM）完全是一种假想的机器，通常有两种方法描述它：一种是多值模型、一种是猜想模块模型。多值模型认为它和确定性图灵机的共同之处是也包括：

(a)带中字符的集合 Γ，使得 $\Sigma \subset \Gamma$ 且 $b \in \Gamma - \Sigma$。

(b)有限状态集 $Q \supseteq \{q_0, q_Y, q_N\}$。

而不同之处在于其控制功能为：

(c) $\overline{\delta} : (Q - \{q_Y, q_N\}) \times \Gamma \to Q \times \Gamma \times \{l, r\}$ 的一个子集。

确定性图灵机在任一状态一次只能做一种运算，非确定性图灵机则不同，理论上，它在同一时刻里可以独立、并行地完成（无限）多种运算，也就是说是多值的。这在实际中显然是不可能的。

猜想模块模型是一种更形象、直观的描述方法，用一种稍微非标准的、但等价于标准形式的语言来讲，可将 NDTM 描述成：除多了一个猜想模块（guessing module）外，它与 DTM 有着完全相同的结构，而这个猜想模块带有自己的仅可对带写的猜想头，它提供了写下猜想的办法，并仅用于此目的。

基于这一模型，一个 NDTM 程序则可用完全类同于一个 DTM 程序的方式来进行定义，并用相同的记号（包括带中字符集 Γ，输入字符表 Σ，空白符号 *b*，状态集 *Q*，初始状态 q_0，两个停机状态 q_Y 和 q_N，以及状态转移函数：$\overline{\delta} : (Q - \{q_Y, q_N\}) \times \Gamma \to Q \times \Gamma \times \{l, r\}$。但对于一个输入 $x \in \Sigma^*$, NDTM 程序的计算却与 DTM 程序的不同，它把计算分为两个不同的阶段：猜想阶段和检验阶段。

第一个阶段为猜想阶段一开始，输入字符串 *x* 被写在带中格 1 到格 $|x|$ 的

带格中，其余带格均为空白字符。读写头将在扫描带格 1，而猜想头在扫描带格 -1。有限状态控制器处于不起作用的状态，猜想模块处于起作用状态，并一次一步地指示猜想头：要么在正被扫描的带格中写下Γ中的某一字符左移一格；要么停止。若为停止，则这时猜想模块变为不起作用的而有限状态控制器则变为起作用的并处于状态q_0。猜想模块是否保持起作用、以及如果起作用的话，那么从Γ中选择哪个字符等均由猜想模块以某一完全任意的方式来决定。因此，猜想模块在其停止以前可以从Γ^*中写下任一字符串，而且可以永不停止。

当有限状态控制器处于状态q_0时，检验阶段就开始了。从此时起，计算将在该 NDTM 程序的指示下，按照与 DTM 程序完全相同的规则进行。而猜想模块及其猜想头在完成了将所猜字符串写到带上的任务后将不再参与到程序的执行中去。当然，在检验阶段，前面所猜的字符串可能会被，而且通常将被考察。当有限状态控制器进入两个停机状态中之一个时，计算就会停止。若停机于q_Y，则称为是一个可接受的计算，否则（包括永不停机）则称为是一个未接受的计算。

一般说来，对于一个给定的输入字符串 x，NDTM 程序 M 将会做无限多个可能的计算，对每一个Γ^*中的可能猜想串都有一个相应的计算。如果这些计算中至少有一个为可接受的计算，则称 NDTM 程序 M 接受 x，相应地，M 所识别的语言定义为

$$L_M = \{x \in \Sigma^* : \text{M 接受 } x\}。$$

定义一个 NDTM 程序 M 接受$x \in L_M$所需要的时间为：在 M 对 x 的所有可接受计算中，程序从一开始直到进人停机状态q_Y为止在猜想和检验阶段所进行的步数的最小值。而 M 的时间复杂性函数 $T_M: Z^+ \to Z^+$则相应地定义为

$$T_M(n) = \max\left[\{1\} \cup \left\{\begin{array}{c} m : \text{存在一个长度为} n \text{的} x \in L_M，\text{使得M} \\ \text{要接受它所需要的时间为} m \end{array}\right\}\right]$$

上式意味着程序 M 的时间复杂性函数的值仅依赖于 M 在那些可接受的计算中所进行的计算步数，而且约定，当不存在长度为 n 的输入字符串，使得 M 可接受它时，则令 $T_M(n)$等于 1。

若存在一个多项式 p，使得对所有的$n \geq 1$，有$T_M(n) \leq p(n)$，则称 NDTM 程序 M 为一个多项式时间 NDTM 程序。

有了上述这些准备工作，则我们现在可以引进另一类重要的语言：NP 类(the class NP)，其定义如下：

NP＝{L：存在一个多项式时间 NDTM 程序 M，使得 LM＝L}。

称判定问题 II 在编码策略 e 之下属于 NP，若 $L[II,e] \in$ NP。与 P 类语言时的讨论一样，只要编码策略 e 是合理的，则可简单地称 II 属于 NP。

10.5 多项式时间的归结和 NP 完备性

- NP 包含了我们所关心的许多问题，而人们怀疑这些问题是 P 的。事实上，全部合理的组合最优化问题的判定形式都是 NP 的。
- P = NP? 现在人们普遍认为 P 是 NP 的真子集，但是尚无正式的证明，并且这个所谓的 P = NP 现在是计算机科学家们正面临的一个最重要的理论问题，尽管缺乏任何肯定和否定的证明，人们还是在一定程度上揭示了这个奥秘——归结的概念。

如果 P 与 NP 不同，那么 P 与 NP-P 之间的差别是有意义的而且是重要的。

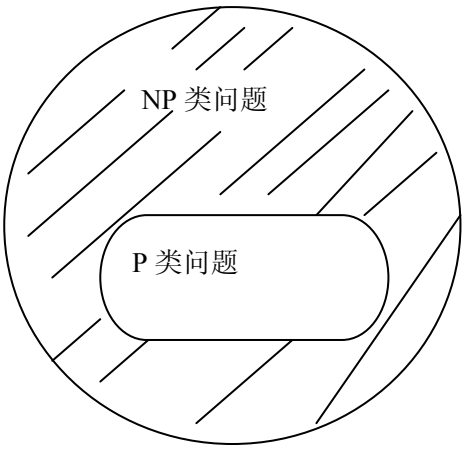
P 中的问题：有多项式算法

NP-P 中的问题：是难解的

所以，给定判定问题 $\pi \in NP$，如果 $P \neq NP$，我们自然希望知道 $\pi \in P$ 还是 $\pi \in NP - P$。

当然，只要我们不能证明 $P \neq NP$，就没有希望证明任何具体问题属于 NP-P。由于这个原因，NP 完备性理论集中考虑证明较弱的结论：

若 $P \neq NP$，则 $\pi \in NP - P$。



对 NP 世界的暂定看法

另一方面：一旦假定我们有个解决某问题的有效算法，则解决另一个计算问题就很容易。

如：最大流问题 → 分层网络最大流（子问题）

适定性问题 → 整数规划

定义：设 A_1 和 A_2 都是判定问题，称 A_1 在多项式时间内归结为 A_2，当且仅当 A_1 有个多项式时间的算法 $\mathbf{A_1}$，并且 $\mathbf{A_1}$ 是多次地以单位费用把 A_2 的（假象）算法 $\mathbf{A_2}$ 作为子程序的算法。称 $\mathbf{A_1}$ 叫做 A_1 到 A_2 的多项式时间归结。

上述定义中至关重要的一点是以单位费用的条款。这意味着算法 $\mathbf{A_2}$ 被看成占用单位执行时间的单一指令，当然这不现实，不过：

命题 1. 如果 A_1 在多项式时间内归结为 A_2，而 A_2 有多项式时间的算法，则 A_1 也有多项式时间的算法。

一类特殊的多项式时间归结是特别重要的：

定义：称判定问题 A_1 多项式地变换到另一个判定问题为 A_2，如果给定任意符号串 x，我们在（$|x|$ 的多项式时间内能构造出符号串 y，使得 x 是 A_1“是”

的实例，当且仅当 y 是 A_2“是”的实例。

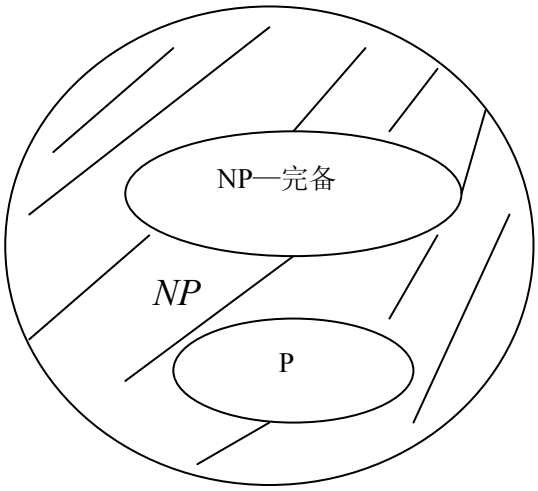
多项式时间变换可看作一个多项式时间归结，它正好在 A_1 的算法末尾调用 A_2 的子程序。算法的其余部分只是构造 A_2 的算法输入 y。

如：适定性　　→（变换）　整数线性规划
　　一般最小费用流　→（变换）　运输问题
都是多项式时间变换。

定义：判定问题 $A \in NP$ 称为是 NP—完备的，如果所有其它的 NP 问题都能多项式归结到 A。

根据命题 1，若问题 A 是 NP—完备的，那么它有令人生畏的性质：若 A 有有效算法，则每个 NP 问题也有有效算法，当然包括 TSP 问题、整数规划问题、适定性问题和团等一些老大难问题。反之，如果 NP 中的某一个问题是难解的，那么所有 NP—完备问题也都是难解的。

假设 $P \neq NP$，则 NP 的构造如图：



对复杂性世界的猜测图

注意：NP 不是简单地划分成“P 的领域”和“NP—完备的领域”。若 $P \neq NP$，则在 NP 中一定存在既不能在多项式时间内解决，又不是 NP—完备的问题。

我们将会看到 NP—完备问题确实存在。

为了证明一个问题 π 是 NP—完备的，我们必须证明：

（1）$\pi \in NP$

（2）某个已知的 NP—完备问题 π' 多项式变换到 π。

但是，在我们能够使用这个方法之前，我们还需要有第一个 NP—完备问题。这样一个 NP—完备由 Cook 定理给出。

10.6 Cook 定理

荣获“第一个”NP—完备问题名誉称号的是布尔逻辑中的一个判定问题，通常称作适定性问题）缩写为 SAT）。

定理(Cook)：适定性问题是 NP—完备的。

为了证明适定性问题是 NP—完备的，必须证明：

（1）$SAT \in NP$

（2）所有其它 NP 问题都能多项式归结到该问题。

推论：整数规划问题是 NP—完备的。

10.7 NP—完备性证明

如果每一个 NP—完备的问题都要象适定性问题的证明那样复杂，那么已知的 NP—完备问题的种类就不会这么多。但是，只要我们证明了一个问题是 NP—完备的，那么证明其它问题的 NP 完备性的过程就大大简化了。给定问题 $\pi \in NP$，我们只要证明某个已知的 NP—完备问题 π' 可以多项式归结到 π 就够了。所以，从现在起，判定问题 π 的 NP—完备性的证明过程将由下述四步组成：

(1) 证明 $\pi \in NP$

(2) 选择一个已知的 NP—完备问题 π'

(3) 构造从 π' 到 π 的变换 f

(4) 证明 f 是一个多项式的变换

六个基本的 NP—完备问题

这 6 个问题是最常用的问题，并且是已知的 NP—完备问题的“基本核心”。

1. 3 适定性问题（3SAT）

2 适定性问题：每个子句限于恰好由两个文字组成，则这个问题能在线性时间内求解——P 问题。

3 适定性问题：每个子句限于恰好由三个文字组成，称为 3 适定性问题。

定理 1. 3SAT 是 NP—完备的。

证明：因为 3SAT 是 SAT 的特殊情况，所以，3SAT 是 NP 问题。

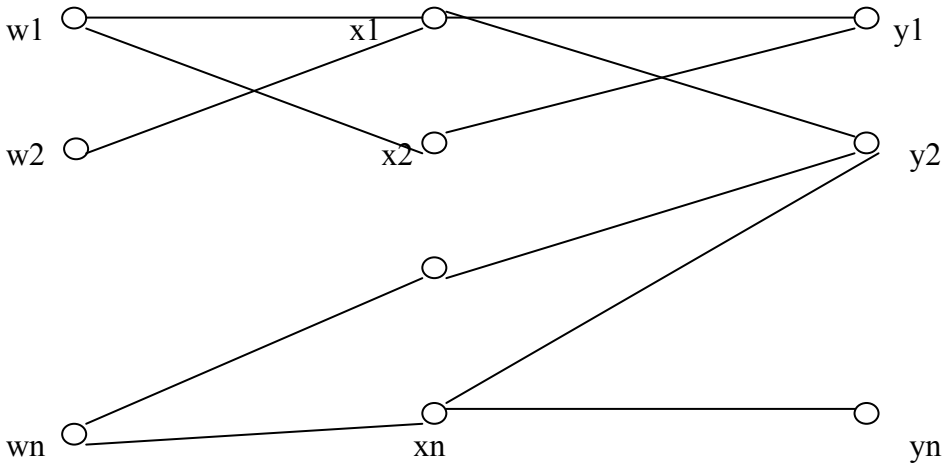
可以证明：任给 SAT 实例，都可以变换称（拆成）3SAT 实例。

2. 3 维匹配问题（3DM）

图 G = (V,E)

3 维匹配问题是经典的“婚姻问题”的推广。

二维匹配问题：



3 维匹配问题

定理 2　3DM 是 NP—完备的。

证明：3SAT ⟶ 3DM

3. 顶点覆盖(VC)和团

顶点覆盖实例：图 G = (V, E)和正整数 $k \le |V|$。

问：G 是否有大小不超过 k 的顶点覆盖，即是否有子集 $V' \subseteq V$ 使得 $|V'| \le k$ 并且对每一条边 $(u,v) \in E$，u 和 v 中至少有一个属于 V'。

独立集实例：图 G = (V, E)和正整数 $k \le |V|$。

问：G 中是否存在 k 点集合 I，使得 I 中任何两点无边相连。

容易验证：独立集、团和顶点覆盖之间的下述关系.

引理：对于任意的图 G = (V, E)和子集 $V' \subseteq V$，下述命题是等价的：

(a) V' 是图 G 的顶点覆盖

(b) $V - V'$ 是图 G 的独立集

(c) $V - V'$ 是图 G 的补图 G^C 中的团

其中 $G^C = (V, E^C)$，且 $E^C = \{(u,v) : u,v \in V \text{且} (u,v) \notin E\}$。所以，在相当强的意义下，这三个问题可以看作只不过是相互“不同变形”。此外，引理中列出的关系使得把这三个问题中的任何一个问题变换到另外一个问题成为很简单的事情。

如：顶点覆盖 ⟶ 团：G = (V, E)，$k \le |V|$ 构成 VC 的一个实例，由图 G^C 和整数 $J = |V| - k$ 就得到对应团的实例。

所以，只要证明了这三个问题中任何一个是 NP—完备的，则所有这三个问

题都是 NP—完备的。

定理 3 顶点覆盖是 NP—完备的。

证明：3SAT ⟶ 顶点覆盖。

4. 哈密顿回路 (HC)

实例：G = (V, E)

问：G 是否包含一条哈密顿回路，即是否有 G 的顶点排列次序 $v_1, v_2, \cdots, v_n$ 使得 $(v_n, v_1) \in E$ 和 $(v_i, v_{i+1}) \in E$, $1 \le i \le n$？其中 $n = |V|$。

定理 4 哈密顿回路 (HC) 是 NP—完备的。

证明：顶点覆盖 ⟶ 哈密顿回路.

推论：TSP 是 NP—完备的。

证明：HC ⟶ TSP.

5. 划分

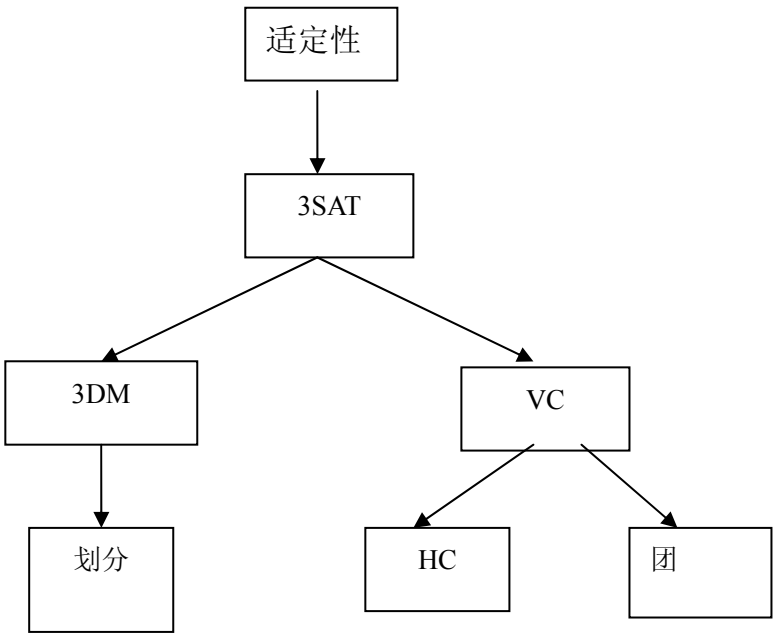
实例：有穷集合 A 以及每一个元素 $a \in A$ 的“大小” $s(a) \in Z^+$。

问：是否有子集 $A' \subseteq A$ 使得 $\sum_{a \in A'} s(a) = \sum_{a \in A - A'} s(a)$？

对于含有数字参数——如长度、重量、成本、生产能力等的划分问题，特别有用。

定理 5 划分 是 NP—完备的。

证明：3DM ⟶ 划分。



这六个问题广泛流传的一个原因是它们都被列在 Karp1972 给出的那张最原始的表中。在那张表中有 21 个 NP－完备问题。

10.9 Co-NP 类问题

哈密顿圈问题

已知图 $G=(V,E)$，G 是哈密顿的吗？（即 G 有一条哈密顿圈吗？）——哈密顿圈问题是 NP—完备的，当然首先是 NP 的。

哈密顿圈问题的余

已知图 $G=(V,E)$，G 是非哈密顿的吗？称为**哈密顿圈问题的余。**

问题：哈密顿圈问题的余是否也是 NP 的？
回答：大概不是（除非 NP＝co-NP）。

注：要证明一个图 $G=(V,E)$ 是非哈密顿的，至今唯一的办法是列举 $G=(V,E)$ 的包含图的所有顶点的所有圈——这是一个指数长的证明。

定义：设问题 A 的实例的编码符号串集合是 $\sum$，如果问题 $\overline{A}$ 的实例的编码符号串集合也是 $\sum$，且问题 $\overline{A}$ 的实例恰好不是问题 A 的实例，则称问题 $\overline{A}$ 是问题 A 的余。

注：（1）严格地，问题 A 的余 $\overline{A}$ 不是问题 A 的补。为什么？

（2）在不影响问题的复杂性的前提下，问题 A 的余 $\overline{A}$ 也认为是问题 A 的补。

TSP 的余：

已知整数 n，$n\times n$ 整数矩阵 $D=[d_{ij}]_{n\times n}$ 和整数 L，问所有的循环排列（环游）τ 都有

$$\sum_{j=1}^{n} d_{j\tau(j)} > L \text{ 吗？}$$

证明方法：列出所有的环游，验证是否成立。没有其它的方法。

P 问题的余

连通行问题的余：

已知图 $G=(V,E)$，问它是不连通的吗？

显然：在多项式时间内解决连通性问题的搜索算法可以用来解决它的余。所以，连通行问题的余也是 P 问题。

定理：如果问题 A 是 P 的，则问题 A 的余 $\overline{A}$ 也是 P 的。

证明：因为问题 A 是 P 的，则存在多项式算法求解问题 A 的任何一个实例。求解问题 A 的多项式算法可以同时求解问题 A 的余 $\overline{A}$，只不过先前回答是的时候，现

在回答否，先前回答否的时候，现在回答是。

注：同样的论证不能用于证明任何 NP 问题的余也是 NP 的。
事实上，哈密顿圈问题的余和 TSP 的余都不是 NP 的。
定义：co-NP 类是所有这种问题的类，它们是 NP 问题的余。
猜想：NP≠co-NP。
定理：如果一个 NP－完备问题的余是 NP 的，则 NP＝co-NP。

所以，在所有的 NP 问题中，NP－完备问题是这样一些问题，它们的余可能不是 NP 的。反之，如果一个 NP 问题的余也是 NP 的，就证明该问题不是 NP－完备的。