第八章　NP-完全问题

§1　关于问题及算法的描述

为了应用算法复杂性理论，首先要对问题、问题的一般描述、计算模型、算法、算法的复杂性给出严格的定义。但在给出精确定义之前，我们先回顾一下有关概念的粗略解释。

所谓一个**问题**(problem)是指一个有待回答、通常含有几个取值还未确定的自由变量的一个一般性提问(question)。它由两部分构成：一是对其关于参数的一般性描述；二是对该问题的答案所应满足的某些特性的说明。而一个问题的某个**实例**则可通过指定问题中所有参数的具体取值来得到。以下用Π表示某个问题，用I表示其实例。

旅行商问题的参数是由所需访问城市的一个有限集合$C=\{C_1,C_1,\cdots,C_m\}$和C中每对城市C_i,C_j之间的距离$d(C_i,C_j)$所组成。它的一个解是对所给城市的一个排序

$$C_{\pi(1)},C_{\pi(2)},\cdots,C_{\pi(m)}$$

使得该排序极小化下面表达式（目标函数）的值

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

旅行商问题的一个**实例**是通过指定城市的数目，并指定每两个城市之间的具体距离而得到的。例如：

$$C=\{C_1,C_2,C_3,C_4\},\quad \begin{array}{l} d(C_1,C_2)=10,d(C_1,C_3)=5,d(C_1,C_4)=9,\\ d(C_2,C_3)=6,d(C_2,C_4)=9,d(C_3,C_4)=3 \end{array}$$

就是旅行商问题的一个实例，这个实例的一个最优解是排序C_1,C_3,C_4,C_2，因为四个城市的这个排序所对应旅行路线是所有可能环游路线中长度最小的，其长度为27。

目前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)；二是把问题转化为判定问题(decision problem)。实际中几乎所有问题都可直接或间接地转述为判定问题。

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与答案为“是”的例子所构成的子集$Y_\Pi\subset D_\Pi$来刻画。不过，为了反映实际问题所具有的特性，通常所采用的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”表示这一部分；第二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“问”。因此，一个具体例子属于D_Π当且仅当它可通过用具有特定性质的某些对象替代一般性例子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”。

例 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$d(C_i,C_j)\in Z^+$以及一个界$B\in Z^+$。

问 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，存在C的一个排序$C_{\pi(1)},C_{\pi(2)},\cdots,C_{\pi(m)}$，使得

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

这是一个将优化问题转化成判定问题的例子。一般地，一个优化问题可以看作是其所有实例的集合，而每一个实例为一个元素对(F,c)，其中F是可行解集，c是目标函数。一个最优化问题可以提出下述三种模式：

- ✓ 最优化模式：求最优解；
- ✓ 求值模式：求出最优值；
- ✓ 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$f\in F$，使得$c(f)\le L$？

显然，在求解最优值不太困难的假设下，上述三种模式的每一种都不比前一种困难。一般而且比较现实的假设是：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。在这种情况下，用二分搜索(或叫折半搜索)技术证明，只要判定模式存在多项式时间算法，求值模式也存在多项式时间算法。

所谓**算法**(algorithm)是指用来求解某一问题的、带有一般性的一步一步的过程。它是用来描述可在许多计算机上实现任一计算流程的抽象形式，其一般性可以超越任何具体实现时的细节。

注意，复杂性理论中对算法的定义与我们通常理解的具体算法(即用某种计算机语言编写的、可在某一特定计算机上实现的计算机程序)有所不同。不过，将算法想象为某个具体的计算机程序，在许多情况下可以帮助我们理解有关概念和理论。

附：一个算法的严格定义直到1936年才出现，丘奇(Alonzo Church)和图灵(Alan Turing)分别在他们的文章中给出。丘奇使用称为$\lambda-$演算的记号系统来定义算法，图灵用机器(图灵机)来定义算法，后来证明两者是等价的。此前，希尔伯特的第10问题就是要设计一个算法来测试多元多项式是否有整数根。不过他不用“算法”这个词，而是用一句短语:“通过有限次运算就可以决定的过程”。我们这里采用图灵的定义,即借用图灵机计算模型来给出算法的精确定义。

到目前为止，关于算法的描述大致有三种不同的程次：一是形式描述，即详尽的写出图灵机的状态、转移函数等等，这是最底层也最详细的描述；二是实现描述，使用日常语言来描述图灵机的运行，如怎样移动读写头，怎样在带上存储数据等，但是不给出状态和转移函数的细节；三是高层的描述，它也使用日常语言来描述算法，但是忽略实现的模型，这种描述不需要提及机器是怎样管理它的带子或读写头的。

称一个算法求解一个问题Π，是指该算法可以应用到Π的任一实例I，并保证能找到该实例的一个解。一个算法的有效性可以用执行该算法所需要的各种计算资源的量来度量。最典型也是最主要的两个资源就是所需要的时间和内存空间。但时间需求常常是决定某一特定算法在实际中是否真正有用和有效的决定性因素。

应该注意到，由于计算机速度和指定系统的不同，同一算法所需时间的多少随着计算机的不同可能有很大差别。为使算法分析具有一般性和在实际中有用，所给时间的度量不应该依赖于具体计算机，即是说，如果它们用不同的编程语言来描述，或在不同的计算机上运行，好的算法仍然保持为好的。另一点需要注意的是，即使同一算法和同一计算机，当用它来求解某一问题的不同例子时，由于有关参数取值的变化，使得所运行时间也有很大差别。对于前一个问题的解决，人们提出了一些简单但又能反映绝大多数计算机的基本运作原理的计算模型，如各种形式的图灵(Turing)机、随机存储机(RAM)等，然后基于这些计算模型来研究算法。对于第二个问题的解决，人们引进了问题例子大小(size)的概念。所谓一个问题例子的大小是指***为描述或表示它而需要的信息量***。只要表示的合适，可望例子的大小的值应该与求解的难易程度成正比。并称相应的表示法为编码策略(encoding scheme)。

事实上，作为输入提供给计算机的对任一问题例子的描述可以看作是从某一有限字母表中选取所需的字符而构成的有限长字符串。通常称该字母表中的字符为码，而由其中之字符如何组成描述问题例子的字符串的方法则称为编码策略。合理的编码策略应该具有可解码性(decodability)和简洁性(conciseness)。一种典型的方法就是利用所谓的结构化字符串，通过递归、复合的方式来给出所考虑问题的合理编码策略。

给定一个问题Π，假设已经找到描述问题一般性例子的一个合理编码策略e，则对Π的任一实例I，称其依编码策略 e 所得的描述相应问题实例的字符串中所含有字符的个数为其输入长度，并将该输入长度的具体值作为例子I的大小的正式度量。例如，若用字母表

$$\{C,[,],/,0,1,2,3,4,5,6,7,8,9\}$$

中的字符表示旅行商问题的例子，则前面所给该问题的具体例子可以用字符串

$$"C[1]C[2]C[3]C[4]//10/5/9//6/9//3"$$

来描述，其相应的输入长度为 32。我们说旅行商问题的这个例子的大小为 32。

虽然所有的判定问题均可用统一的术语来描述，但从数学上讲还是不够严密，不宜于给出算法的严格定义，也不利于算法复杂性研究。为了克服这些不足，我们引进“语言”这一术语，它与判定问题有着很自然的联系。

设Σ是一个字符集，用Σ^*表示由Σ中的字符组成的所有有限长字符串的集合。Σ^*的任何非空子集L都称为Σ上的一个语言(language)。例如$\{01,001,111,1010110\}$就是字符集$\{0,1\}$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。一旦选定了某个字符集Σ，则对于任一个判定问题Π及其编码策略e，Π和e将会把Σ^*中的所有字符串划分为三部分：那些不是Π的例子的编码、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。后一类编码正是与Π通过e来联系的语言。定义：

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

如果一个结论对语言$L[\Pi,e]$成立，则说它在编码策略e下对问题Π成立（计算复杂性理论所直接考虑的是对语言或字符串集的分析）。

对于任何两个合理的编码策略e和e’，问题Π的某个性质要么对$L[\Pi,e]$和$L[\Pi,e']$均成立，要么对二者均不成立。因此，可以直接说某个性质对Π成立或不成立，也常常将$L[\Pi,e]$简记为$L[\Pi]$。但是，这样的省略却失去了对输入长度的具体确定办法，而我们还需要象这样的一个参变量以便能确切表述复杂性的概念。为此，以后总是隐含假定：

对每个判定问题Π，均有一个不依赖于编码方式的函数

$$\text{Length：}\ D_\Pi \to Z^+$$

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。这里，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 和 q，使得如果 $\mathrm{I}\in D_{\Pi}$ 且 x 为 I 在 e 下的编码，则有

$$\mathrm{Length[I]}\le p(|x|)\text{ 且 }|x|\le q(\mathrm{Length[I]})$$

这里，$|x|$表示字符串 x 的长度。易知，Π的任何两个合理的编码策略将导出多项式相关的输入长度。例如，对于旅行商问题，对应的判定问题的任一例子 I，可以定义

$$\mathrm{Length[I]}=m+\lceil \mathrm{lb}B\rceil+\max\left\{\lceil \mathrm{lb}d(C_i,C_j)\rceil \mid C_i,C_j\in C\right\}$$

这里，lb$x$表示 $\log_2 x$。

在后面的陈述中，会交替地使用(判定)问题、语言这两个术语而不加区分。

§2　图灵机与确定性算法

为了算法复杂性分析具有普适性，即一个算法的效能与具体的计算机无关，我们需要选定一种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。**图灵机本质上是**一个具有存储载体（通常用一个有无限多个方格线性带表示）的、按照具体指令可完成向左或右移动、放置标记、抹去标记以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**。在原理上，与我们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。

首先选择确定性单带图灵机(deterministic one-tape Turing machine)，简称确定图灵机，记为 DTM。一个 DTM 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格的线性带所组成。

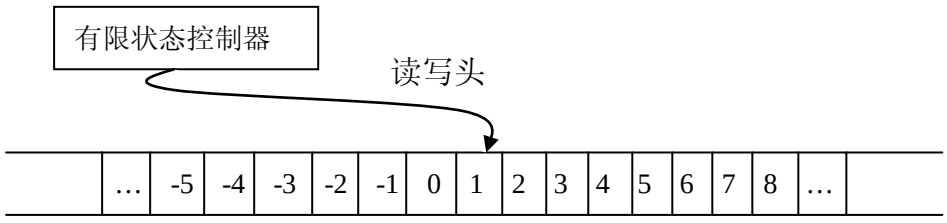

图 8-2-1 单带确定性图灵机示意

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，它应包含输入字符表子$\Sigma \subset \Gamma$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$\delta : (Q - \{q_Y, q_N\}) \times \Gamma \to Q \times \Gamma \times \{l, r\}$

$$(q, s) \mapsto (q', s', \Delta)$$

表示了状态变化、字符变化及读写头移动方向，而且是唯一的。

一个 DTM 程序运行很简单。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存放在带格 1 到带格$|x|$中。所有其它的带格均存放空白字符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”。

若当前状态为$q \in Q - \{q_Y, q_N\}$，且$s \in \Gamma$为读写头当前扫描带格中的字符，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，并写上字符s'；同时，如果$\Delta = l$，则读写头左移一格；如果$\Delta = r$，则读写头右移一格。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准备，除非已处于停机状态。

可见，当前状态、带格的内容以及读写头所在的位置是图灵机的要素，这三者的整体称为一个**格局**，图灵机的运行就是根据转移函数发出的指令从一个格局转移到另一个格局。

有了 DTM 这个计算模型以及定义于它上面的程序概念，就可以给出算法及其复杂性函数的严格定义。设 M 是一个 DTM 程序，输入字符表为Σ。我们说程序 M 接受字符串$x \in \Sigma^*$，如果它作用于x时停机于状态q_Y，并称

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

为程序 M 所识别的语言。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（不是该问题的实例）。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（在有限步内）停机时，才称其为一个**算法**（实际是判定器）。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 的计算时间定义为该程序从开始直至进入停机状态为止所运行的步数。由此可以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，其时间复杂性函数 $T_M : Z^+ \to Z^+$ 定义为：

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

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。我们称

$$P = \{L \mid \text{存在一个多项式时间 } DTM \text{ 程序 } M, \text{ 使得 } L = L_M\}$$

为 P 语言类。如果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。

§3　NP 类问题

不难看出，上面定义的 P 类语言只能用来描述那些存在有效算法（多项式时间）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。其中有一大类这样的问题，虽然不知道求解它们的有效算法，但是，一旦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(称为多项式时间可验证性)。这类问题的分析和描述需要借助另一类图灵机作为计算模型。

非确定性单带图灵机(non-deterministic one-tape Turing machine)，简记为 NDTM，是一种假想的机器。通常有两种方式描述它：**多值模型**和**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).带中字符集 Γ，使得 $\Sigma \subset \Gamma$，且 $b \in \Gamma - \Sigma$；

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

不同之处在于

(c). 多值转移函数 $\overline{\delta}:(Q\setminus\{q_Y,q_N\})\times\Gamma\to 2^{Q\times\Gamma\times\{l,r\}}$,

$$(q,s)\mapsto S\subseteq Q\times\Gamma\times\{l,r\}$$

确定性图灵机在任一状态下只能做一种运算，而非确定性图灵机可以被想象为在同一时刻能够独立、并行地完成多种运算（表现在转移函数的多值性），这显然不现实。

通过允许作不受限制的并行计算可以对不确定性算法做出明确的解释。每当要作某种选择时，算法就好像给自己复制了若干副本，每种可能的选择有一个副本，于是，许多副本同时被执行。第一个获得成功的副本将引起对其它副本计算的结束。如果一个副本获得不成功的完成则只该副本终止。

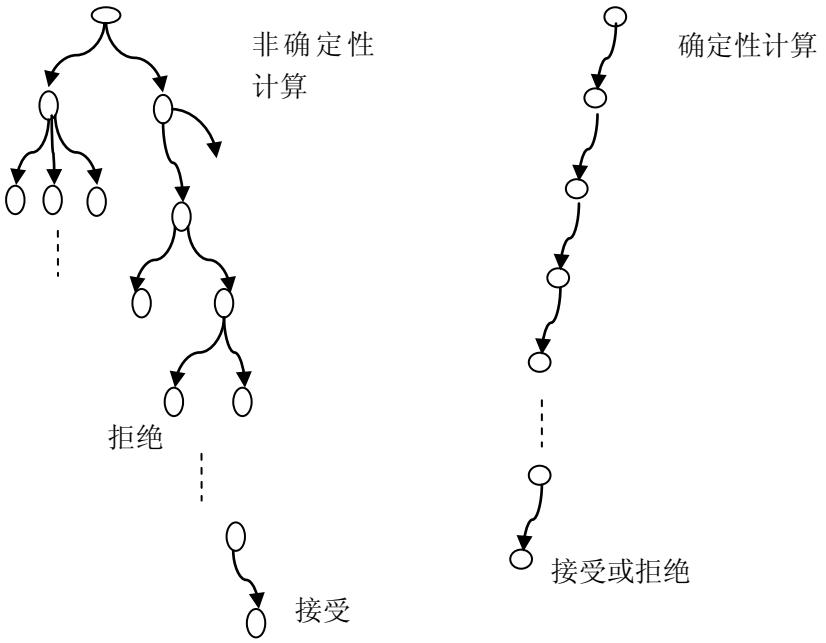

图 8-3-1 与确定性算法的比较

不确定算法举例：

程序 8-3-1 不确定性排序算法

```
pro NSort(A,n)
  //对 n 个正整数排序
  integer A(n),B(n),n,i,j;
  B:=0 //对 B 进行初始化
  for i to n do
     j:=choice(1..n);
     if B[j]≠0 then failure;  end{if};
     B[j]:=A[i];
  end{for}
  for i to n-1 do //验证 B 的次序
```

```
      if B[i]>B[i+1] then failure;  end{if};
    end{for}
    print(B);
    success;
  end{NSort}
```

程序 8-3-2　0/1 背包判定问题的不确定算法

```
proc NPack(P,W,n,M,R,X)
  integer P(n),W(n),R,X(n),n,M,i;
  X:=0 //对 X 进行初始化
  for i to n do
     X[i]:=choice(0,1);
  end{for};
  if  ∑_{1≤i≤n} w(i)X(i) > M or ∑_{1≤i≤n} P(i)X(i) < R  then failure;
  else success;
  end{if};
end{NPack}
```

“**猜想模块模型**”是另一种更形象、直观的解释方法。可将 NDTM 描述成：除多了一个猜想模块（guessing module）外，NDTM 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进行写操作的猜想头，它提供写下猜想的方法，仅此而已。

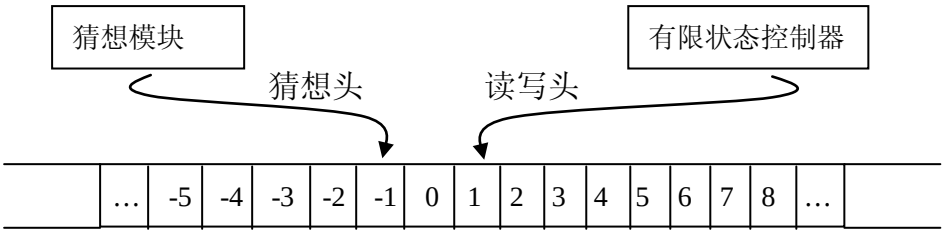

图 8-3-2　NDTM 猜想模块模型示意图

基于这一模型，一个 NDTM 程序可以类同于一个 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

$$\delta:(Q\setminus\{q_Y,q_N\})\times\Gamma\to Q\times\Gamma\times\{l,r\}$$

但对于一个输入$x\in\Sigma^*$，NDTM 程序的计算过程却与 DTM 程序的计算过程不同，它把计算过程分为两个阶段，即**猜想阶段**、**检验阶段**。

第一阶段为猜想阶段。开始时，输入字符串 x 被写在由格 1 到格 $|x|$ 的带上，其余带格为空白字符。读写头将扫描带格 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：要么在被扫描的带格中写下 Γ 中的某一个字符并左移一格；要么停止。若为停止，猜想模块即变为不起作用的，而有限状态控制器变为起作用的并处于状态 q_0。猜想模块是否保持起作用，以及起作用时该从 Γ 中选择哪个字符写在带格中，均由猜想模块以某种完全任意的方式决定。

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，计算机就在该 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到两个停机状态之一时，计算就会停止。若停机为 q_Y，则说是一个可接受的计算，否则（包括永不停机），说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 所识别的语言为

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

一个 NDTM 程序 M 接受 $x \in L_M$ 的时间定义为：在 M 对于 x 的所有可接受计算中，程序从一开始直到停机状态 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 $T_M : Z^+ \to Z^+$ 则定义为：

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

若存在多项式 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。并称

$$NP = \{ L | \text{存在一个多项式时间 } N\text{DTM 程序 M，使得 } L_M = L \}$$

为 NP 语言类。称判定问题 Π 在编码策略 e 下属于 NP，若 $L[\Pi, e] \in NP$。与 P 类语言时的讨论一样，只要编码策略是合理的，就可以简单地称问题 Π 属于 NP。

因为任何现有的计算模型都可以通过加上一个类似于 NDTM 中的带有只写头

的猜想模块来扩充，然后相对于所得的模型重新描述上述的正式定义，而且所有如此得到的计算模型在多项式时间内可相互转换的意义下将是等价的。因此，也没有必要特别提及 NDTM 模型，我们将简单地说“**多项式时间不确定性算法**”，并将 NP 类语言与所有可用多项式时间不确定性算法求解的判定问题等同看待。

例子：考虑无向图的团问题：

例：给定一个有 n 个顶点的无向图 $G=(V,E)$ 和一个整数 k 。

问：G 是否包含一个具有 k 个顶点的完全子图（团）？

如果用邻接矩阵表示图 G，用二进制串表示整数 k，则团问题的一个实例可用长度为 $m=n^2+\log k+1$ 的二进制串表示。团问题可表示为语言

$$CLIQUE=\{w\#v \mid w,v\in\{0,1\}^*,\ w\text{为邻接矩阵的图}G\text{有一个}k\text{顶点的团},\ v\text{是}k\text{的二进制表示}\}$$

一个接受语言 *CLIQUE* 的非确定性算法可以设计如下：

第一阶段将输入串 $w\#v$ 进行分解，并计算出 $n=\sqrt{|w|}$，以及用 v 表示的整数 k。若输入不具有形式 $w\#v$ 或 $|w|$ 不是一个平方数，则拒绝输入。这阶段可在 $O(n)$ 内完成。

第二阶段，非确定性地选择 V 的一个 k 元子集 $V'\subseteq V$，并用一个位向量 A[1..n]表示：

A[i]=1 当且仅当 $i\in V'$，因而 A 中恰有 k 个 1.

程序 8-3-3 非确定性选择算法

```
integer j,m;
j:=0;
for i to n do
   m:=choice(0,1);
   case:
     m=0: A[i]:=0;
     m=1; A[i]:=1; j:=j+1;
   end{case}
end{for}
if j≠k then failure; end{if}
```

第三阶段，确定性地检查 V' 的团性质。若 V' 是一个团则接受，否则拒绝。

这可以在$O(k^2)$时间内完成。因此整个算法的时间复杂性为$O(n+k^2)=O(n^2)=O(m)$。

如果图$G=(V,E)$不包含一个k团，则算法的第二阶段产生的任何k元子集V'不具有团性质。因此算法没有导致接受状态的计算路径。反之，若图G含有一个k团V'，则算法的第二阶段中有一个计算路径产生V'，使得在算法的第三阶段导致接受状态。

注意到，算法的第二阶段是非确定性的且耗时为$O(n)$。整个算法的计算时间复杂性主要取决于第三阶段的验证算法，即给定图G的一个k团猜测V'，验证它是否是一个团。若验证部分可在多项式时间内完成，则整个非确定性算法具有多项式时间复杂性。这一特征具有一般性，为此我们定义验证算法：

验证算法定义为具有两个自变量的算法 A，其中一个自变量是通常的输入串 X，另一个自变量是一个称为“证书”的二进制串 Y。如果对任意串 X∈L，存在一个证书 Y 并且 A 可以用 Y 来证明 X∈L，则算法 A 就验证了语言 L. 称

$$VP=\left\{\begin{array}{l}L\mid L\in\Sigma^*,\Sigma\text{为一个有限字符集，存在一个多项式 }p\text{和}\\ \quad\text{多项式时间验证算法}A(X,Y),\text{使得对于任意}X\in\Sigma^*,\\ \quad X\in L\text{当且仅当存在}Y\in\Sigma^*,|Y|\le p(|X|)\text{且}A(X,Y)=1\end{array}\right\}$$

为多项式时间可验证语言类。

定理 1：$VP=NP$。

证明：先证明$VP\subseteq NP$。对于任意$L\in VP$，设p是一个多项式且 A 是一个多项式时间验证算法，则下面的非确定性算法接受语言L：

1） 对于输入X，非确定性地产生一个字符串$Y\in\Sigma^*$；

2） 当 A(X,Y)=1 时接受 X.

该算法的第一步与团问题的第二阶段非确定性算法一样，至多在$O(|X|)$时间内完成。第二步的计算时间是|X|和|Y|的多项式，而$|Y|\le p(|X|)$。因此它也是|X|的多项式。整个算法可多项式时间内完成。至此$L\in NP$。

反之，设$L\in NP$，$L\in\Sigma^*$，且非确定性图灵机 M 在多项式时间p内接受语言L。设 M 在任何情况下只有不超过d个的下一动作选择，则对于输入串 X，M

的任一动作序列可用$\{0,1,\cdots,d-1\}$的长度不超过$p(|X|)$的字符串来编码。不失一般性，设$|\Sigma|\geq d$。验证算法A(X,Y)用于验证“Y是M上关于输入X的一条接受计算路径的编码”。即当Y是这样一个编码时A(X,Y)=1. A(X,Y)显然可在多项式时间内确定性地进行验证，且

$$L=\{X\mid \text{存在}Y\text{使得}|Y|\leq p(|X|)\text{且}A(X,Y)=1\}$$

因此$L\in VP$。　　证毕

定理2 若$\Pi\in NP$，则存在一个多项式p，使得Π可以用一个时间复杂性为$O(2^{p(n)})$的确定性算法来求解。

证明：设A为求解Π的一个多项式时间不确定性算法，其时间复杂性由一个多项式$q(n)$来界定。不失一般性，设q可在多项式时间内被估值。因此，对于长度为n的每个被接受的输入，必然存在字符集Γ上长度至多为$q(n)$的某个猜想字符串，使算法A的检验阶段在不多于$q(n)$步内回答“是”。若$|\Gamma|=k$，则所需要考虑的可能猜测的数目最多为$k^{q(n)}$。对于一个长度为n的给定输入，通过应用算法A的确定性检验阶段到相应的$k^{q(n)}$多个可能猜测字符串中的每一个，直到停止或运行$q(n)$步，我们可以肯定地发现A对于该输入是否有一个可接受计算。如果A在该时间界内遇到一个导致可接受计算的猜测串，则该模拟过程回答“是”；否则回答“非”。这显然形成了一个求解Π的确定性算法，而且该算法的时间复杂度将以$q(n)k^{q(n)}$为上界，其等价于$O(2^{p(n)})$。　　证毕

定理2的证明指出，在非确定性图灵机上时间复杂性为$q(n)$的判定问题与在确定性图灵机上时间复杂性为$q(n)k^{q(n)}$的问题相当，因此，直观上我们有理由认为多项式时间不确定性算法要比多项式时间确定性算法的速度快得多，能够在多项式时间内求解后者所不能够求解的许多其它问题。由此及许多其它理由，在目前已知知识背景下，人们普遍认为P是NP的真子集。关于这方面的研究基本上有两条线路：

1）　证明NP类中的某些问题是难解的，从而得到$P\neq NP$。但是这同原问题的难度几乎相当，也许只有建立一套全新的数学论证方法才有希望解决。

2）　考虑NP类中问题之间的关系，从中找到一些具有特定性质的、与P

中问题有显著不同的问题。沿此路线人们已经证明了在 NP 类中存在一个称为 NP－完全的子类，并由此发展出一套著名的 NP－完全理论。而证明一个问题是 NP－完全的通常被认为一个告诉我们应该放弃寻找、设计求解该问题的有效算法（多项式时间算法）的强有力证明。

§4 NP 完全问题

研究 P=NP 的问题有两条基本思路：

1．证明 NP 类中的某些问题是难解的，从而得到 NP≠P。但是要证明这一点几乎同证明 P=NP 一样困难。

2．考察 NP 类中问题之间的关系，从中找到一些具有特殊性质的问题。沿着这一路线人们已经证明了在 NP 类中存在被称为 NP-完全的子类，简称 NPC 问题，并由此发展了一套著名的 NP 完全理论。

本节简要介绍 NP-完全性理论。为此，首先引入各语言之间的多项式变换的概念。

定义 1 所谓从一个语言 $L_1 \subseteq \Sigma_1^*$ 到另一个语言 $L_2 \subseteq \Sigma_2^*$ 的多项式变换是指满足下面两个条件的函数 $f:\Sigma_1^* \to \Sigma_2^*$，

(1) 存在计算 f 的一个多项式时间 DTM 程序；

(2) 对于所有的 $x \in \Sigma_1^*$ 有：$x \in L_1$ 当且仅当 $f(x) \in L_2$。

用 $L_1 \propto L_2$ 表示存在一个从语言 L_1 到语言 L_2 的多项式变换。相应地，对于判定问题 Π_1, Π_2，设 e_1 和 e_2 是相应的编码策略。若 $L[\Pi_1, e_1] \propto L[\Pi_2, e_2]$，则记为 $\Pi_1 \propto \Pi_2$。

也可以从问题的角度来叙述：由判定问题 Π_1 到判定问题 Π_2 的多项式变换是满足下列条件的函数 $f:D_{\Pi_1} \to D_{\Pi_2}$，

(1) f 可由一个多项式时间的确定性算法来计算；

(2) 对于所有的 $I \in D_{\Pi_1}$ 有：$I \in Y_{\Pi_1}$ 当且仅当 $f(I) \in Y_{\Pi_2}$。

定义 2 称一个语言 L（判定问题 Π）为 NP-完全的（NPC），如果

$$L \in NP\ (\Pi \in NP)\ ,$$

且对于所有别的语言$L' \in NP$（判定问题$\Pi' \in NP$）均有

$$L' \propto L\ (\Pi' \propto \Pi)\ 。$$

按照定义 2，要证明问题Π是 NP 完全的，需要证明所有的 NP 问题均能够经多项式变换变成Π。这几乎是很难做到的。如果 NP-完全问题比较多，我们也不能对每一个这样的问题都这样验证。为此我们讨论一些 NPC 问题的有用的性质。

性质 1 如果$L' \propto L$，则$L \in P$意味着$L' \in P$。

性质 2 如果$L' \propto L$而且$L \propto L''$，则$L' \propto L''$。

由性质 1，2，不能推出下列结论，

定理 2 设Π是 NP 完全的，如果$P \neq NP$，则$\Pi \in NP \setminus P$。

定理 3 如果$L', L \in NP, L' \propto L$，则$L' \in NPC \Rightarrow L \in NPC$.

定理 3 是证明 NP 完全问题的基础。但这需要一个 NPC 问题作为源问题。Cook 首先给出了这样一个问题--可满足性问题。可满足性问题是数理逻辑中一个重要问题，它定义在布尔变量之上。

给定布尔变量集$U = \{u_1, u_2, \cdots, u_m\}$，$U$ 上的一个真值分配是指一个映射$t: U \to \{T, F\}$。U 上的一个子句 c 就是由一些布尔变量（或它们的“非”）通过逻辑“或”连接起来的布尔表达式

$$c = z_1 \vee z_2 \vee \cdots \vee z_k, z_i \in U \cup \bar{U}, i = 1, 2, \cdots, k \qquad (8.4.1)$$

其中，$\bar{U} = \{\bar{u}_1, \bar{u}_2, \cdots, \bar{u}_n\}$。若**存在**对于布尔变量集$U$的一个真值分配，使得该子句取值为真，则说该子句被满足。子句的集合说是可满足的，如果存在U的一个真值分配，使得集合中的每个子句的取值均为真。可满足性问题可陈述如下：

例 给定布尔变量之集U以及U上子句的一个集合 C。

问 是否存在U的一个真值分配，使得 C 是可满足的。

Cook 定理 可满足性问题是 NP-完全问题。

从 Cook 的开创性工作至今，人们已经发现并证明了数千个 NPC 问题（如，0/1 背包问题和 Hamilton 回路问题），总结出证明 NP-完全性的几种方法，并建立了如何分析、进而近似求解 NP-完全问题的方法。以下列出几个典型的 NPC 问题：

✧　**三维匹配问题** 3DM(3 Dimensional Matching)

例： 给定三个互不相交的、均含有 q 个元素的集合 W,X,Y，取 $M \subseteq W \times X \times Y$。

问： M 包含一个匹配吗？即是说，是否存在一个子集 $M' \subseteq M$，使得 $|M'| = q$，且 M' 中任意两个三元组都没有相同的分量。

✧ **三元精确覆盖**问题 X3C(Exact Cover by 3-sets)

例：给定有限集合 $X, |X| = 3q$，以及 X 的三元子集族 C。

问： C 含有 X 的一个精确覆盖吗？即是说，是否存在一个子族 $C' \subseteq C$，使得 X 的每个元素恰好只出现在 C' 的一个三元子集中。

注意到，如果令 $U = W \cup X \cup Y$，$C = \{\{w,x,y\} | w \in W, x \in X, y \in Y\}$，则三元匹配问题就转化为三元精确覆盖问题（若已知道三元匹配问题是 NP-完全问题，那么，三元精确覆盖问题也必是 NP-问题)。

✧ **顶点覆盖**问题 VC（Vertex Cover）

例：给定一个图 G(V,E)和一个正整数 K≤|V|.

问： 是否存在 G 的一个顶点数不超过 K 的覆盖？即是否存在一个顶点子集 $V' \subseteq$ V, $|V'| \leq$ K，使得对于每一条边{u,v}∈E，u 与 v 中至少有一个属于 V'.

✧ **Hamilton 回路**问题 HC（Hamiltonian Circuit）

例：已知一个图 G(V,E)。

问：G 含有一个 Hamilton 回路吗？G 的 Hamilton 回路是指包含图 G 的所有顶点的简单回路(圈)，即是 G 的顶点的一个排序：[$v_1, v_2, \ldots, v_n$]，其中 n=|V|，使得对所有的 i: 1 ≤ i < n, { v_i, v_{i+1}}∈E，{ v_n, v_1}∈E.

✧ **划分问题**

例 已知一个有限集合 A 及对于每个 $a \in A$ 的一个权值 $s(a) \in Z^+$。

问 问是否存在 A 的一个子集 $A' \subseteq A$，使得 $\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)$ ？

✧ **三元可满足性问题** 3SAT

例 给定布尔变量的一个有限集合 U 及定义于其上的子句集 $C = \{c_1, c_2, \cdots, c_m\}$, 其中 $|c_i| = 3, i = 1,2,\cdots,m$

问 是否存在 U 上的一个真赋值，使得 C 中所有的子句均被满足？

§5　证明新问题是 NPC 问题的方法

根据上节的结论，证明一个新问题 Π 是 NPC 问题的一般步骤是：

(a)　证明 $\Pi \in NP$；

(b)　选取一个已知的 NP 完全问题 Π'；

(c)　构造一个从 Π' 到 Π 的变换 f；

(d)　证明 f 为一个多项式变换。

这里一个关键的问题是如何选取“参照物” Π' 和构造多项式变换 f。在实际证明过程中参照物的选择带有一定的经验性，已知的 NPC 问题实例越多越有利。一般情况下，很难写出多项式变换的明确的解析表达式，但是可以给出问题的转换过程，并说明这个转换过程可以通过多项式时间完成。

例 8.5.1 证明三元可满足性问题 3SAT(3-Satisfiability)是 NPC 问题

例：给定布尔变量的一个有限集合 $U=\{u_1,u_2,\cdots,u_n\}$ 及定义于其上的逻辑语句 $C=C_1\wedge C_2\wedge\cdots\wedge C_m$，其中，$|C_i|=3$，$i=1,2,\cdots,m$。

问：是否存在 U 的一个真赋值，使得 C 为真？

证明：我们选取可满足性问题 SAT 作为参照物。设 $U=\{u_1,u_2,\cdots,u_n\}$ 及语句 $C=C_1\wedge C_2\wedge\cdots\wedge C_m$ 构成 SAT 的任一实例。我们构造新的布尔变量集 U' 及定义其上的三元语句 C'，使得 C 可满足当且仅当 C' 可满足。

对每个子句 C_j，设 $C_j=z_1\vee z_2\vee\cdots\vee z_k$，其中 $z_i\in\overline{U}\cup U$，$1\le i\le k$，$\overline{U}=\{\overline{u}_1,\overline{u}_2,\cdots,\overline{u}_n\}$

情形 1：k=1, 令

$$U_j'=\left\{y_j^1,y_j^2\right\}$$

$$C_j'=\left(z_1\vee y_j^1\vee y_j^2\right)\wedge\left(z_1\vee y_j^1\vee \overline{y}_j^2\right)\wedge\left(z_1\vee \overline{y}_j^1\vee \overline{y}_j^2\right)\wedge\left(z_1\vee \overline{y}_j^1\vee \overline{y}_j^2\right);$$

情形 2：k=2，令

$$U_j'=\{y_j^1\},\quad C_j'=(z_1\vee z_2\vee y_j^1)\wedge(z_1\vee z_2\vee \overline{y}_j^1);$$

情形 3：k=3，令

$$U_j'=\{\},\quad C_j'=C_j;$$

情形 4：k>3，令

$$U_j'=\{y_j^i\mid 1\le i\le k-3\}$$

$$C_j'=(z_1\vee z_2\vee y_j^1)\wedge \underset{1\le i\le k-4}{\wedge}(\overline{y}_j^i\vee z_{i+2}\vee y_j^{i+1})\wedge(\overline{y}_j^{k-3}\vee z_{k-1}\vee z_k)$$

最后令 $U'=U\cup\bigcup_{i=1}^{m}U_i'$，$C'=\underset{1\le i\le m}{\wedge}C_i'$

要证明上述转换的确是一个变换，只需证明：C'可满足当且仅当C可满足。

设$t:U\to\{T,F\}$为满足C的一个真赋值，以下证明t可扩展成满足C'的一个真赋值：$t':U'\to\{T,F\}$。因为$U'\backslash U$中的变量可分解成不同的集合U_j',$1\le j\le m$，而每个U_j'的变量仅出现在属于C_j'的子句中，我们仅需要说明如何将t扩充到各个U_j'即可，且证明在上述四种情形的每一种情形下，C_j'中的所有子句均被满足。

若U_j'属于情形 $k=1$或情形 $k=2$，则C_j'中的字句已被t所满足，从而可任意地扩展它到U_j'，如对所有的$y\in U_j'$，令$t'(y)=T$。

若U_j'是由情形$k=3$所确定的，那么U_j'为空集，而t的赋值已经使$C_j'=C_j$中的单个子句取真值。

若U_j'是由情形$k>3$所确定的，因为t为C的一种可满足性真赋值，必然存在一个最小的整数，使得文字z_l在t之下被赋予真值。若l为 1 或 2，则可对$i:1\le i\le k-3$，令$t'(y_j^i)=F$；若l为$k-1$或k，则对于$i:1\le i\le k-3$，令$t'(y_j^i)=T$；其余情况，令$t'(y_j^i)=T,1\le i\le l-2;$ 。 $t'(y_j^i)=F,l-1\le i\le k-3$

容易证明，这些选择将保证C_j'中所有的子句均被满足，进而C'中的所有子

句也均被赋值t'所满足。

反之，若t'为C'的一个可满足性真赋值，则不难证明t'对于U中变量的限制必形成对C的一个可满足性真赋值。至此，我们证明了C可满足当且仅当C'可满足。

要证明上述变换可在多项式时间内完成，只需注意到C'中三元子句的数目被$m\cdot n$的一个多项式所界定,因为，任一个子句的长度k不超过$2n$。也就是说，3SAT 例子的大小由 SAT 例子的大小的一个多项式函数所界定。由此，根据上述构造方法，不难证明它是一个多项式变换。

例 8.5.2 团问题（CLIQUE）

例：给定一个无向图$G(V,E)$和一个正整数k。

问：G是否包含一个k团？即是否存在$V'\subseteq V,|V'|=k$，且对任意$u,v\in V'$，有$(u,v)\in E$。

可以证明 CLIQUE 是 NP 问题，下面通过 3-SAT∝CLIQUE 来证明 CLIQUE 是 NP 难的，从而说明 CLIQUE 是 NP 完全的。

设$C=C_1\wedge C_2\wedge\cdots\wedge C_k$是一个三元合取范式，其中

$$C_i=c_1^i\vee c_2^i\vee c_3^i,\quad i=1,2,\cdots,k$$

构造图$G(V,E)$如下：对于每个子句C_i定义三个顶点：v_1^i,v_2^i,v_3^i分别与子句中的成分c_1^i,c_2^i,c_3^i对应，以每个子句对应的三个顶点为一部分，构造一个k部图，只要$i\neq j$而且$c_s^i\neq\overline{c}_t^j$（不是互非的），则顶点$v_s^i$与顶点$v_t^j$之间就连一条边（参看图 8-5-1）。

以下证明C可满足当且仅当G有k团。如果C可满足，则存在逻辑变量的一组真值分配，使得$C=T$，因而每个子句都取真值：$C_i=T,\ i=1,2,\cdots,k$。在每个子句中至少有一个成分取真值，这样在 G 的每部分中取一个顶点，这个顶点对应的子句成分是取真值的，我们得到一个子集合$V'\subseteq V$。因为V'中任何两个顶点属于不同部分，而且它们所对应的成分不可能是互为非的（因为都取真值），所以这两个顶点是相连的。因而V'构成一个k团。反之，如果G有一个k团V'，则

V' 中的顶点来源于 k 部图 G 的每一部分恰好一个。对于 $v_s^i \in V'$，v_s^i 等于逻辑变量 u_p 时，则给 u_p 分配真值；如果 v_s^i 等于逻辑变量 u_p 的非 $\overline{u}_p$，则给 u_p 分配假值。这样就给部分逻辑变量分配了值，而且这种分配不会出现矛盾（根据图 G 的定义），其余未提到的变量随意取定值即可。对于这种真值分配，C 一定是满足的，因为每个子句中至少有一个成分取真值。

考察例子 $C=(x_1 \vee \overline{x}_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$

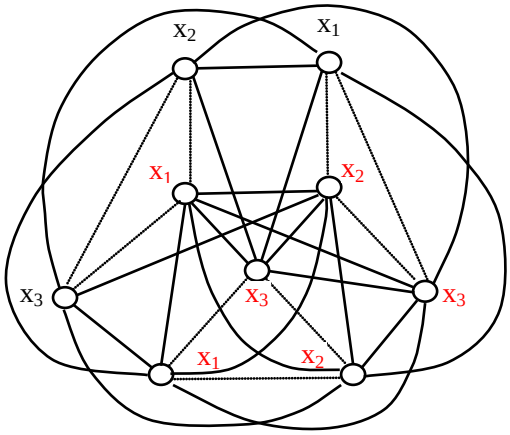

图 8.5.1 一个三元子句对应的图

例 8.5.3 顶点覆盖问题 VERTEX-COVER

例：给定无向图 $G(V,E)$ 和一个正整数 k。

问：G 是否有 k 覆盖，即是否存在 $V' \subseteq V, |V'| = k$，使得对任意 $(u,v) \in E$，有 $u \in V' \text{ or } v \in V'$。

顶点覆盖问题是 NP 问题。我们通过 CLIQUE$\propto$VERTEX-COVER 来证明 VERTEX-COVER 是 NP-完全的。

对于 CLIQUE，设 n 个顶点的图 G(V,E) 有 k 团 $V' \subseteq V$，则 G 的补图 $\overline{G}$ 有 n-k 覆盖 $V'' = V \setminus V'$。反之亦然。

例 8.5.4 精确覆盖问题 XC

例：已知有限集合 S 的子集族 C

问：C 是否包含一个精确覆盖，即是否存在 $C' \subseteq C$，使得 C' 中元素互不相交，且 $\bigcup_{c \in C'} c = S$。

这个问题是 NPC 问题。我们将利用它证明定和子集问题是 NPC 问题。

例 8.5.5　定和子集问题 DSS

例：给定非负整数集合 S，正整数 M

问：是否存在子集$C \subseteq S$, 使得$\sum_{c \in C} c = M$

给定精确覆盖的一个实例：集合$F = \{f_1, \cdots, f_n\}$，其子集族$T = \{T_1, \cdots, T_k\}$，构造定和子集问题的一个实例：$S = \{s_1, \cdots, s_k\}$，其中$s_i = \sum_{1 \le j \le n} \delta_{ij} (k+1)^{j-1}$，其中，当$f_j \in T_i$时$\delta_{ij} = 1$；当$f_j \notin T_i$时$\delta_{ij} = 0$。取$M = \sum_{1 \le j \le n} (k+1)^{j-1} = \left((k+1)^n - 1\right)/k$。当$T$含有$F$的精确覆盖$T'$时，令$S' = \{s_i \mid T_i \in T'\}$，则$\sum_{s_i \in S'} s_i = M$。反之，由$\sum_{s_i \in S'} s_i = M$令$T' = \{T_i \mid s_i \in S'\}$可知$T'$是$F$的精确覆盖。

例 8.5.6 划分问题 PARTS

例：已知一个有限集合A，及对每个$a \in A$赋予的权值$s(a) \in Z^+$

问：是否存在一个子集$A' \subseteq A$，使得$\sum_{s \in A'} s(a) = \sum_{s \in A \setminus A'} s(a)$

以下证明 DSS $\propto$ PARTS。给定非负整数集合$S = \{s_1, \cdots, s_k\}$和正整数M，构造集合$A = \{a_1, \cdots, a_k, a_{k+1}, a_{k+2}\}$，其中

$$a_i = s_i, 1 \le i \le k; \quad a_{k+1} = M + 1, a_{k+2} = 1 - M + \sum_{1 \le i \le k} s_i$$

当且仅当S有一个和数为M的子集时，A有一划分。

例 8.5.7　0/1 背包（判定）问题 0/1 KPS

例：给定一个有限集合U，对于每个$a \in U$，对应一个值$w(a) \in Z^+$和另一个值$v(a) \in Z^+$。另外给定一个约束值$B \in Z^+$和目标$K \in Z^+$

问：是否存在U的一个子集$U' \subseteq U$，使得$\sum_{a \in U'} w(a) \le B$，而且$\sum_{a \in U'} v(a) \ge K$

以下证明：PARTS $\propto$ KPS。给定一个划分问题的实例：有限集合A，及对每个$a \in A$的一个权值$s(a) \in Z^+$。构造一个 0/1 背包问题实例：$U = A, \forall a \in U, w(a) = v(a) = s(a)$，定义$w(a) = v(a) = s(a)$。再令$B = K = \frac{1}{2} \sum_{a \in A} s(a)$。如果$A$有一个划分$A'$，则$\sum_{a \in A'} s(a) = K = B$，因而$\sum_{a \in A'} w(a) = B$，而且$\sum_{a \in A'} v(a) = K$，

取$U'=A'$即是 0/1 背包问题上述实例的解。反之，若$U'\subseteq U$是 0/1 背包问题上述实例的解，则$\sum_{a\in U'}w(a)\le B$，而且$\sum_{a\in U'}v(a)\ge K=B$，因而$\sum_{a\in U'}v(a)=B=\frac{1}{2}\sum_{a\in A}v(a)$，于是取$A'=U'$即得划分问题上述实例的解。

例 8.5.8 区间排序问题（Sequencing within intervals）

例：给定任务的有限集合W，对于每个任务$w\in W$，相应有一个开始时间$r(w)$和终止时间$d(w)$以及工作时间$l(w)$，其中$r(w)\in Z^+\cup\{0\}$，$d(w),l(w)\in Z^+$。

问：是否存在任务集的一个可行时间排序表，即是否存在函数$\sigma:W\to Z^+\cup\{0\}$，满足下面两个条件：

a) 对每个$w\in W$，有$\sigma(w)\ge r(w)$,且$\sigma(w)+l(w)\le d(w)$；

b) 若$w'\in W,w'\ne w$，则$\sigma(w')+l(w')\le\sigma(w)$或$\sigma(w')\ge\sigma(w)+l(w)$。

证明：选取划分问题作为“参照物”：有限集合A以及对每个$a\in A$的权值$s(a)\in Z^+$。现在由划分的一般性实例构造区间排序问题的一般性实例。令$B=\sum_{a\in A}s(a)$。将$a\in A$用一项任务w_a来置换，并令其满足$r(w_a)=0$, $d(w_a)=B+1$, $l(w_a)=s(a)$。再引进一个附加任务$\overline{w}$，其满足$r(\overline{w})=\lceil B/2\rceil$, $d(\overline{w})=\lceil(B+1)/2\rceil$, $l(\overline{w})=1$。这一构造过程显然可以在多项式时间内完成。以下证明：当且仅当上述划分问题的例子回答为是时所构造的区间排序问题例子的回答才为是。

附加任务$\overline{w}$对于可行时间排序表的限制表现在两个方面。首先，它确保B为奇数时不可能构造出一可行排序，因为此时由$r(\overline{w})=d(\overline{w})$，而$l(\overline{w})=1$，故$\overline{w}$不可能被排序。同时，$B$为奇数表明所对应的划分问题例子的回答必为非。故以下不妨设B为偶数，但这时附加任务$\overline{w}$所起的第二个限制就表现出来了。因为B是偶数，所以$r(\overline{w})=B/2, d(\overline{w})=r(\overline{w})+1$，并由可行排序的第一个要求知，可行排序必然使$\sigma(\overline{w})=B/2$,这样就限制了余下的任务只能安排被任务$\overline{w}$分离的两个

时间段内，且每个时间段的长度为$B/2$，如下图所示

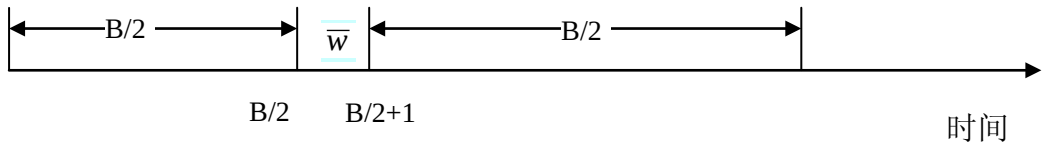

图 8.5.2 区间排序示意图

因此，排序问题就被转化为把余下任务划分为两个子集的问题，其中一个子集中所选的任务都被安排在任务$\overline{w}$之前的时间段内完成，而另一个子集中的任务均被安排在任务$\overline{w}$之后的时间段内完成。由于两个时间段的总的时间长度恰好等于除了$\overline{w}$以外的所有任务总的工作时间，故两个时间段恰好被排满。然而要做到这一点的充要条件是存在一个子集$A'\subseteq A$，使得

$$\sum_{a\in A'} s(a) = B/2 = \sum_{a\in A\setminus A'} s(a)。$$

因此，对划分问题的回答为是当且仅当对相应的区间排序问题存在一个可行时间排序，即对它的回答也为是。至此，证明了划分问题可多项式变换到区间排序问题。

例 8.5.9 有向哈密顿圈问题 DHC

例：给定有向图$G(V,E)$

问：G 是否有一个有向 Hamilton 圈，即长度为$n=|V|$，而且恰好经过每个顶点一次，然后回到起始顶点。

DHC 是 NP 问题。我们通过 CNF-可满足性 $\propto$ DHC 证明 DHC 是 NP 完全的。设$C=C_1\wedge C_2\wedge\cdots\wedge C_k$，$n$个逻辑变量：$x_1,x_2,\cdots,x_{n.}$。画一个有$r$行和$2n$列的数组，第$i$行表示子句$C_i$。对每个$j$作两个相邻的列，前一列代表$x_j$，后一列代表$\overline{x}_j$，$j=1,2,\cdots,n$。若$x_j$是$C_i$的成分，则在对应$C_i$行与$x_j$列交叉处画一个⊗；若$\overline{x}_j$是$C_i$的成分，则在对应$C_i$行与$\overline{x}_j$列交叉处画一个⊗。在$x_j$和$\overline{x}_j$两列之间引入两个顶点：　u_j在列的顶端，v_j在列的底部。对于每个j，从v_j向上到u_j画两条（由边组成的）链，一条把x_j列上的⊗连起来，另一条把$\overline{x}_j$列的⊗连起来。然后再画边

(u_j, v_{j+1})， $j=1,2,\cdots,n-1$。在每一行C_i的右端引一个方框 $\boxed{i}$，并画边$(u_r, \boxed{i})$和$(\boxed{i}, v_1)$，再画边$(\boxed{i}, \boxed{i+})$，$i=1,2,\cdots,r$。如下图

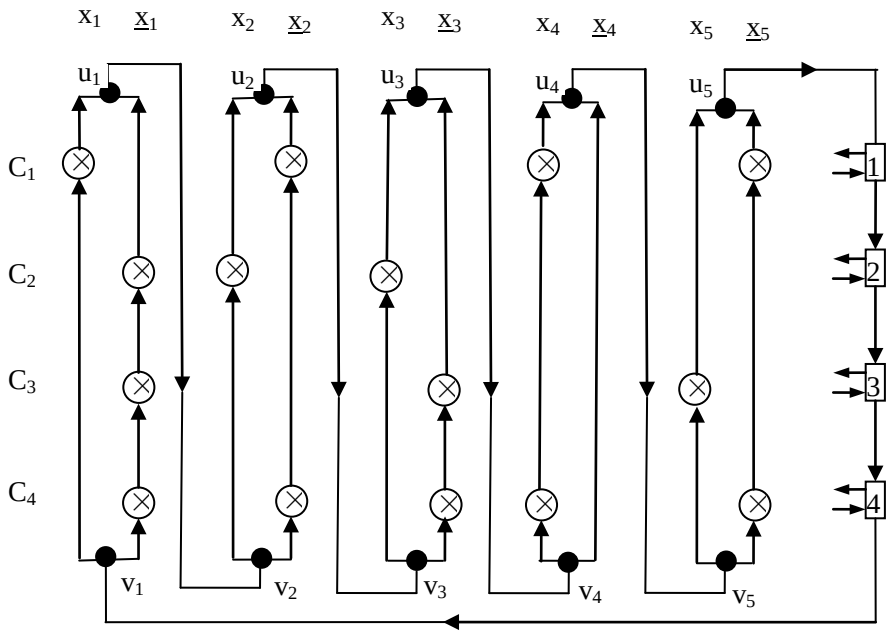

图 8.5.3 Hamilton 问题

此外，还要将图中的每一个⊗用如下左边的子图替代，$\boxed{i}$用如下右边的子图替代(见图 4.3)。这里，A_i是入口顶点，B_i是出口顶点，每条边$(\boxed{i}, \boxed{i+})$代之以(B_i, A_{i+1})，边$(u_r, \boxed{i})$和$(\boxed{i}, v_1)$分别代之以(u_r, A_1)和(B_r, v_1). 而$R_{is} \to \otimes \to R_{i\,s+1}$表示从$R_{is}$进入 ⊗ 的$w_1$顶点，$R_{i\,s+1}$从 ⊗ 的$w_3$顶点引出。至此，有向图$G$构造完毕(见图 8.5.3)。

若C满足，S 是相应的真值分配。则G的有向 Hamilton 圈可以从v_1开始，行进到u_1，然后到v_2，再u_2，v_3，再u_3，…，u_r. 在由v_i向上行进到u_i时，若x_i在 S 中为真则采用x_i对应那一列；否则沿$\bar{x}_i$对应的列向上行进。这个圈将从u_r行进到A_1，然后经过R_{11}，R_{12}，…，R_{1p}，B_1(注意，这里的$R_{1,1}$实际上应该是$R_{1,j+1}$，而 j 是第一个进到的 ⊗，然后诸$R_{1,k}$按照逆时针循环检查)到A_2，…，最后回到v_1。在任一子图 的R_{is}行进到$R_{i\,s+1}$的过程中，当且仅当某个子图 ⊗ 的顶点还不在v_1到R_{is}的路径上时，则转移到第 i 行的 ⊗ 子图。注意到，若C_i的长度是

i_p，则 至多转移到 i_p-1 个 ⊗ 子图，这是因为在C_i至少已经通过了一个 ⊗ 子图。从而，若C满足，则G有一个有向 Hamilton 圈。

反之，若G有一个有向 Hamilton 圈，则G的有向 Hamilton 圈上的顶点 v_1 开始，由于子图⊗和子图　的结构，这样的有向圈必须向上恰好沿每一对(x_i，$\underline{x}_i$)中一列行进。另外，圈的这部分在每一行必须至少经过一个⊗子图。因此，用于从 v_i 行进到 u_i（i=1,2, … ,n）的那一列定义了一组使C为真的真值分配。

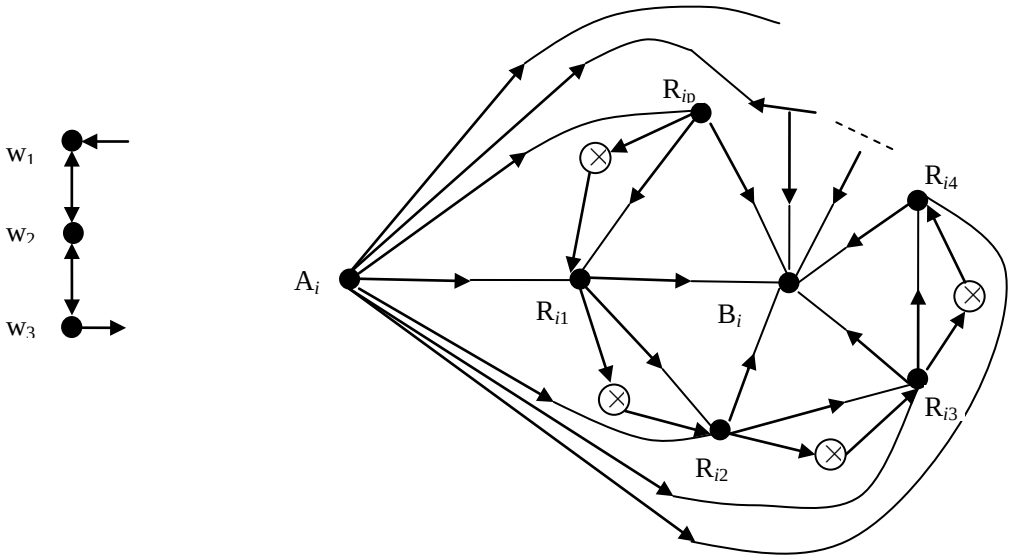

图 8.5.4 Hamilton 圈分解图

例 8.5.10　三元精确覆盖问题 X3C

例：给定有限集合 X, |X|=3q, 以及 X 的三元子集族 C。

问：C 含有 X 的一个精确覆盖吗？即是否存在一个子族$C'\subseteq C$，使得 X 的每个元素恰好只出现C'的一个三元子集中。

这个问题是 NPC 问题。我们将利用它证明 Steiner 树问题是 NPC 问题。该问题可广泛用于描述诸如有关服务设施、厂址的最优设置、某个区域内最廉价交通网、通讯线路的设计等实际问题。

例 8.5.11 Steiner 树问题

例：给定图$G=(V,E)$，对其每条边$e\in E$都有相应的权$w(e)\in Z^+$，另外有 G 的顶点子集$R\subseteq V$，某个界$B\in Z^+$。

问：是否存在 G 的一颗子树 $T=(V_1,E_1)$，使 $R\subseteq V_1\subseteq V, E_1\subseteq E$，而且 $\sum_{e\in E_1} w(e)\le B$？

以下证明 Steiner 树问题是 NPC。选 X3C 问题作为参照物：$X=\{x_1,x_2,\cdots,x_{3q}\}$，三元子集族 $C=\{c_1,c_2,\cdots,c_n\}$，构造 Steiner 树问题的相应例子如下：取 $G=(V,E)$，其中

$$V=\{v_0\}\cup C\cup X,\quad E=\{(v_0,c_i)\mid 1\le i\le n\}\cup\{(c_i,x_j)\mid x_j\in c_i, 1\le i\le n, 1\le j\le 3q\}$$

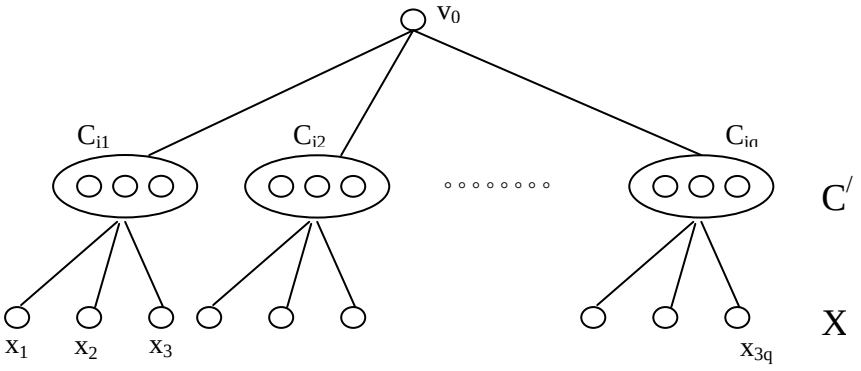

图 8.5.5 Steiner 树问题

令所有边的权值为 1，$R=\{v_0\}\cup X$，$B=4q$。显然这一构造过程可在多项式时间内完成。

如果 $C'\subseteq C$ 为 X 的一个精确覆盖，则取 $T=(V_1,E_1)$，其中

$$V_1=\{v_0\}\cup\{c_i\mid c_i\in C'\}\cup X,$$

$$E=\{(v_0,c_i)\mid c_i\in C'\}\cup\{(c_i,x_j)\mid c_i\in C', x_j\in c_i, 1\le j\le 3q\}$$

由于每个 x_j 恰好出现在 C' 的某个三元子集里，故 T 是连通的无圈图。而且 $|E_1|=4q$，因而是一棵树。注意到 $R\subseteq V_1$，且 $\sum_{e\in E_1} w(e)=|E_1|\le B$，由此知 Steiner 树问题的回答为“是”。

反之，若 $T=(V',E')$ 为 $G=(V,E)$ 的一棵 Steiner 树，$R\subseteq V'$，则每个 $x_j(1\le j\le 3q)$ 都是 T 的顶点。不妨设每个 x_j 都是 T 的叶顶点，因为，否则的话，

顶点x_j的度数大于 1，及存在边$(x_j,c_{i_1}),(x_j,c_{i_2})\in E'\subseteq E$，删去其中一条边，如$(x_j,c_{i_1})$。由于$(v_0,c_{i_1})$，$(v_0,c_{i_2})$不可能都属于$E'$，否则$T$包含有圈，将不属于$E'$的那条边添入$E'$得到另一棵树$T_1$，已知$T_1$是总权值不变的 Steiner 树。再由连通性，每个x_j恰好和某个c_i在T中邻接，令 $C'=\{c_i\mid(c_i,x_j)\in E',1\le j\le 3q\}$，则$C'$显然是$X$的一个精确覆盖。至此，证明了 X3C 问题可多项式变换到 Steiner 树问题。

§6　NP 困难问题

设II_1和II_2是两个判定问题，我们说II_1在多项式时间内可图灵归约为II_2，记做$II_1\propto_T II_2$，如果存在II_1的一个算法A_1，它多次调用求解II_2的算法A_2作为其子程序，而且，若假设每次调用该子程序A_2均需用单位时间，则A_1为一个多项式时间算法。称A_1为从II_1到II_2的多项式归约。

图灵归约也有多项式变换类似的两个性质，特别地，如果判定问题II_1可以归约到II_2，则II_2至少和II_1一样难。图灵归约的定义可不限于判定问题，它可以适用于最优化问题等更广的一类问题。

定义 8.6.1 对于问题II，如果存在一个 NP 完全问题II'，使得$II'\propto_T II$，则称问题II是 NP 困难的（NP-hard）.

由定义 8.6.1， 所有的 NP 类问题都可以多项式归约到任一个 NP 困难问题II，这有时也作为 NP 困难问题的定义。注意，在上述定义中，并不要求$II\in NP$。类似于对 NP 完全问题的讨论方法，不难推出 NP 困难问题的下述性质：

若II是 NP 困难的， 则II不可能在多项式时间内求解，除非 P=NP；

若II是 NP-完全的和 NP 困难的，则其补问题$\overline{II}$必是 NP 困难的。

一个典型的不属于 NP，但是 NP 困难的问题就是第 k 个最重子集问题：

例：已知整数$c_1,c_2,\cdots,c_n,k,L$；

问：存在 k 个不同的子集$S_1,S_2,\cdots,S_k\subseteq\{1,2,\cdots,n\}$，使得对于$i=1,\cdots,k$有

$\sum_{j\in S_i} c_j \geq L$ 吗？

我们已经知道划分问题是NP-完全问题，据此可以证明第k最重子集问题是NP-困难问题。事实上，设$S=\{c_1,c_2,\cdots,c_n\}$是划分问题的某个给定的实例，假设已经有个算法$A[S,k,L]$可以用来求解第k最重子集问题,则可设计出求解划分问题的算法如下：

首先，若$\sum_{i=1}^{n} c_i$为奇数，则立即推出问题的回答为否；若$\sum_{i=1}^{n} c_i$为偶数，当$L=\frac{1}{2}\sum_{i=1}^{n} c_i$为奇数时，用算法$A[S,k,L]$作为子程序，按下列二分搜索技术确定重量至少是$L$的子集的数目$K^*$：

(a) 令 $K_{\min}=0, K_{\max}=2^n$；

(b) 若$K_{\max}-K_{\min}=1$，则置$K^*=K_{\min}$，且停机；

(c) 令 $K=(K_{\max}+K_{\min})/2$，并调用算法$A[S,k,L]$。若回答为“是“，则令$K_{\min}=K$，转(b)；否则，令$K_{\max}=K$，转(b) 。

以上过程通过恰好n次调用算法$A[S,k,L]$，即可找到K^*。至此，只需再调用一次算法即可给出所考虑划分问题例子的答案，即调用$A[S,K^*,L+1]$。若该次调用的答案为“是”，则所有S中重量至少为L的子集也必满足其重至少为$L+1$，因此，S没有重为L的子集，故对划分问题的回答为“否”。相反，若该次调用的回答为“否”，则意味着存在某一个S的子集，使得其重为L，对划分问题的回答为“是”。

由上述迭代过程容易看出，若假设每次调用算法A只需要单位时间，则以上即给出了求解划分问题的一个多项式时间算法。也就是说，我们证明了划分问题可多项式时间归约到第k最重子集问题。

另一个典型的NP困难问题是对称（距离矩阵是对称的）旅行商问题。如果假定已经知道无向图的Hamilton问题是NPC问题，则可以证明对称旅行商问题

是 NP 困难问题如下：

证明：首先，对称旅行商问题不是 NP 的。因为，对其解的任一猜想，要检验它是否是最优的，需要同所有其它的环游比较，这样的环游会有指数多个，因而不可能在多项式时间内完成。

考虑图的 Hamilton 回路问题，已知无向图 $G=(V,E),|V|=n$，构造其对应的旅行商问题如下：

$$d_{ij}=\begin{cases}1, & \text{if } (v_i,v_j)\in E\\ 2, & \text{otherwise}\end{cases}$$

显然，这一变换可以在多项式时间内完成，而且，图 G 有 Hamilton 回路的充分必要条件是上述构建的旅行商问题有解，且其解对应的路程长度为 n。因为，若 G 不含 Hamilton 回路，则这时的旅行商问题的解对应的路程长度至少为 $n+1$。因为已知图的 Hamilton 回路问题是 NP-完全的，且上述变换为多项式变换，故我们证明了对称旅行商问题是 NP-困难问题。

事实上，如果问题 Π 的判定模式是 NPC，则其最优模式一般是 NP-困难的。

习题 八

1．叙述二元可满足性问题，并证明二元可满足性问题是 P 类问题；

2．相遇集问题

例：给定集合 S 的一个子集族 C 和一个正整数 K；

问：S 是否包含子集 S'，$|S'|\le K$，使得 S' 与 C 中的任何一个子集的交均非空？（S' 称为 C 的相遇子集）试判定相遇集问题是 P 类的还是 NP 完全的，并给出你的证明？

3．0/1 整数规划问题

例：给定一个 $m\times n$ 矩阵 A 和一个 m 元整数向量 b；

问：是否存在一个 n 元 0/1 向量 x，使得 $Ax\le b$？

试证明 0/1 整数规划问题是 NP 完全问题。

4.　如何证明精确覆盖问题和三元精确覆盖问题都是 NPC 问题？

5. 独立集问题：

例：对于给定的无向图 $G=(V,E)$ 和正整数 $k\ (\le|V|)$

问：G 是否包含一个于 k 一独立集 V'，即是否存在一个子集 $V'\subseteq V,|V'|=k$，使得 V' 中的任何两个顶点在图中 G 都不相邻。

证明独立集问题都是 NPC 问题（提示：考虑独立集和团的关系：如果V'是图G的团，则V'是G的补图$\tilde{G}$的独立集；反之亦然）。

6. 证明无向图的 Hamilton 圈问题是 NPC 问题。

7. 说明 0/1 背包问题是 NP 困难问题。

8. NP-完全问题一定是 NP 困难问题吗？

提示：对于无向图的 Hamilton 圈问题，可将有向图的（有向）Hamilton 圈问题实例变换成无向图的 Hamilton 问题实例：把已知有向图D的每个顶点u换成欲构造的无向图G的三个顶点：u^i,u^m,u^o，并将顶点u^m与顶点u^i,u^o分别相连。如果在有向图D中有从顶点u到顶点v的有向边（弧），则在无向图G中将顶点u^o与顶点v^i连接一条边。对于独立集问题，

其它问题参考答案：

1．二元可满足性问题 2SAT

例：给定布尔变量的一个有限集合$U=\{u_1,u_2,\cdots,u_n\}$及定义其上的子句$C=\{c_1,c_2,\cdots,c_m\}$，其中$|c_k|=2,k=1,2,\cdots,m$。

问：是否存在U的一个真赋值，使得C中所有的子句均被满足？

证明： 2SAT 是 P－类问题。为叙述方便，采用数理逻辑中的“合取式”表达逻辑命题，于是

$$C=c_1\wedge c_2\wedge\cdots\wedge c_m=\prod_{k=1}^{m}c_k=\prod_{k=1}^{m}(x_k+y_k)$$

其中$c_i\cdot c_j$表示逻辑“与”，x_k+y_k表示逻辑“或”，x_k,y_k是某个u_j或$\bar{u}_i$。

考虑表达式$C=\prod_{k=1}^{m}(x_k+y_k)$，如果有某个$x_k+y_k=u_i+\bar{u}_i$，则在乘积式中可以去掉该子句：$C'=C\setminus(u_i+\bar{u}_i)$，可见$C$与$C'$的可满足性是等价的。所以我们可以假定$C$中不含有形如$u_i+\bar{u}_i$的子句。注意到此时$C$中的子句个数不会超过$n(n-1)$。

如果逻辑变量u_n或它的非$\bar{u}_n$在C的某个子句中出现，我们将C表示成

$$C=G\cdot(u_n+y_1)\cdots(u_n+y_k)(\bar{u}_n+z_1)\cdots(\bar{u}_n+z_h) \qquad (1)$$

其中G是C的一部分子句，而且不出现逻辑变量u_i或它的非$\overline{u}_i$。令

$$C'=G\cdot\prod_{1\le i\le k,1\le j\le h}(y_i+z_j) \qquad (2)$$

(2)式中不再含有变量u_n和它的非$\overline{u}_n$。记$U'=\{u_1,u_2,\cdots,u_{n-1}\}$。如果存在$U$的真赋值，使得$C$满足，在$U'$也一定满足。因为如果$u_n$取真值，则所有的$z_j$必然取真值；$u_n$取价值，所有的$y_i$必然都取真值，不管那中情况，$C'$的乘积部分必然取真值。反之，假设存在$U'$的真赋值，使得$C'$满足。若有某个$y_i$取假值，则所有的$z_j$必然取真值，此时令$u_n$取真值，得到$U$的真赋值，使得$C$满足。若有某个$z_j$取假值，则令$u_n$取假值，得到$U$的真赋值，使得$C$满足。如果所有的$y_i$和$z_j$都取真值，$u_n$取假值得到$U$的真赋值，使得$C$满足。至此我们得到：$C$与$C'$的可满足性是等价的。但是后者涉及的变量数比前者少1，子句数为$m-(k+h)+kh$。但是，我们可以象前面一样简化掉所有形如$u_i+\overline{u}_i$的子句，因而可以假定C'中子句个数不超过$(n-1)(n-2)$。

上述过程可以一直进行到判定只含有一个逻辑变量的逻辑语句的可满足性问题。这需要一个常数时间即可。注意到我们每一步简化都可以在多项式（关于n的）步骤内完成，总共需要至多$n-1$步简化，因而，在多项式时间内可以完成2SAT二满足性问题的判定。即2SAT是P－类问题。　　证毕

2.相遇集问题是NP－完全问题。证明采用限制法：（首先说明相遇集问题是NP－类问题，略）若对所有的$c\in C$，置$|c|=2$，则该问题变成顶点覆盖问题VC，故由VC的NP完全性知相遇集问题也是NP－完全的。

3．证明也是采用限制法，比照0/1背包问题的判定模式是NPC－问题。（注意说明0/1整数规划问题是NP－类问题）。