9 NP完全问题
NP Complete Problem

本章内容提要

- 易解问题与难解问题
- **P**类问题和**NP**类问题
- **NP**完全问题
- **co-NP**类问题
- **NPI**类问题
- **P**、**NP**、**co-NP**、**NPI**类之间的区别与联系
- **NP**完全问题的计算机处理技术简介

9.1 引言

9.1.1 易解问题与难解问题

- 如果对一个问题∏存在一个算法，时间复杂性为$O(n^k)$，其中n是问题规模，k是一个非负整数，则称问题∏存在多项式时间算法。这类算法在可以接受的时间内实现问题求解， e.g., 排序、串匹配、矩阵相乘。
- 现实世界里还存在很多问题至今没有找到多项式时间算法，计算时间是指数和超指数函数（如2^n和n!），随着问题规模的增长而快速增长。
- 通常将前者看作是易解问题，后者看作难解问题。

9.1.2 易解问题与难解问题的主要区别

在学术界已达成这样的共识：把多项式时间复杂性作为易解问题与难解问题的分界线，主要原因有：

1) 多项式函数与指数函数的增长率有本质差别

问题规模 n	多项式函数					指数函数	
	logn	n	nlogn	n^2	n^3	2^n	n!
1	0	1	0.0	1	1	2	1
10	3.3	10	33.2	100	1000	1024	3628800
20	4.3	20	86.4	400	8000	1048376	2.4E18
50	5.6	50	282.2	2500	125000	1.0E15	3.0E64
100	6.6	100	664.4	10000	1000000	1.3E30	9.3E157

2) 计算机性能的提高对易解问题与难解问题算法的影响

假设求解同一个问题有5个算法A1~A5，时间复杂度T(n)如下表，假定计算机C2的速度是计算机C1的10倍。下表给出了在相同时间内不同算法能处理的问题规模情况：

T(n)	n	n′	变化	n′/n
10n	1000	10000	n′=10n	10
20n	500	5000	n′=10n	10
5nlogn	250	1842	$\sqrt{10}$ n<n′<10n	7.37
$2n^2$	70	223	$\sqrt{10}$ n	3.16
2^n	13	16	n′=n+log10≈n+3	≈1

3) 多项式时间复杂性忽略了系数，不影响易解问题与难解问题的划分

问题规模n	多项式函数			指数函数	
	n^8	$10^8 n$	n^{1000}	1.1^n	$2^{0.01n}$
5	390625	5×10^8	5^{1000}	1.611	1.035
10	10^8	10^9	10^{1000}	2.594	1.072
100	10^{16}	10^{10}	10^{2000}	13780.6	2
1000	10^{24}	10^{11}	10^{3000}	2.47×10^{41}	1024

观察结论：$n\leq100$时，（不自然的）多项式函数值大于指数函数值，但n充分大时，指数函数仍然超过多项式函数。

9.2 P类问题和NP类问题

这个划分标准是基于对所谓判定问题的求解方式。

先看看什么是判定问题。事实上，实际应用中的大部分问题问题可以很容易转化为相应的判定问题，如：

- 排序问题 ⇒ 给定一个实数数组，是否可以按非降序排列?
- 图着色问题：给定无向连通图G=(V,E)，求最小色数k，使得任意相邻顶点具有不同的着色 ⇒

  给定无向连通图G=(V,E)和正整数k，是否可以用k种颜色..... ?

确定性算法与P类问题

对判定问题求解，可以采用确定性算法

- 定义9.1(确定性算法): 设A是求解问题∏的一个算法，如果在算法的整个执行过程中，每一步只有一个确定的选择，则称算法A是确定性算法。
  - 特点：对同一输入实例，运行算法A，所得结果是一样的。
- 定义9.2(P类问题): 如果对于某个判定问题∏，存在一个非负整数k，对于输入规模为n的实例，能够以$O(n^k)$的时间运行一个确定性算法，得到yes或no的答案，则称该判定问题∏是一个P(Polynomial)类问题。
  - 事实上，所有易解问题都是P类问题。

非确定性算法与NP类问题

- 定义9.3(非确定性算法):设A是求解问题∏的一个算法，如果算法A以如下猜测+验证的方式工作，称算法A为非确定性(nondeterminism)算法：
- 猜测阶段：对问题的输入实例产生一个任意字串y，在算法的每次运行，y可能不同，因此猜测是以非确定的形式工作。这个工作一般可以在线性时间内完成。
- 验证阶段：在这个阶段，用一个确定性算法验证两件事：首先验证猜测的y是否是合适的形式，若不是，则算法停下并回答no；若是合适形式，则继续检查它是否是问题x的解，如果确实是x的解，则停下并回答yes，否则停下并回答no。要求验证阶段在多项式时间内完成。

注意对非确定性算法输出yes/no的理解：

- 若输出no，并不意味着不存在一个满足要求的解，因为猜测可能不正确；若输出yes，则意味着对于该判定问题的某一输入实例，至少存在一个满足要求的解。

NP类问题

- 定义9.4(NP类问题): 如果对于判定问题∏，存在一个非负整数k，对于输入规模为n的实例，能够以$O(n^k)$的时间运行一个非确定性算法，得到yes/no的答案，则该判断问题∏是一个NP(nondeterministic polynomial)类问题。

※注意：NP类问题是对于判定问题定义的，事实上，可以在多项式时间内应用非确定性算法解决的所有问题都属于NP类问题。

关于P与NP关系的初步思考
--从字面含义

1) 若问题∏属于P类，则存在一个多项式时间的确定性算法，对它进行判定或求解；显然，也可以构造一个多项式时间的非确定性算法，来验证解的正确性，因此，问题也属NP类。因此，显然有

$$P \subseteq NP$$

2) 若问题∏属于NP类，则存在一个多项式时间的非确定性算法，来猜测并验证它的解；但不一定能构造一个多项式时间的确定性算法，来对它进行求解或判定。

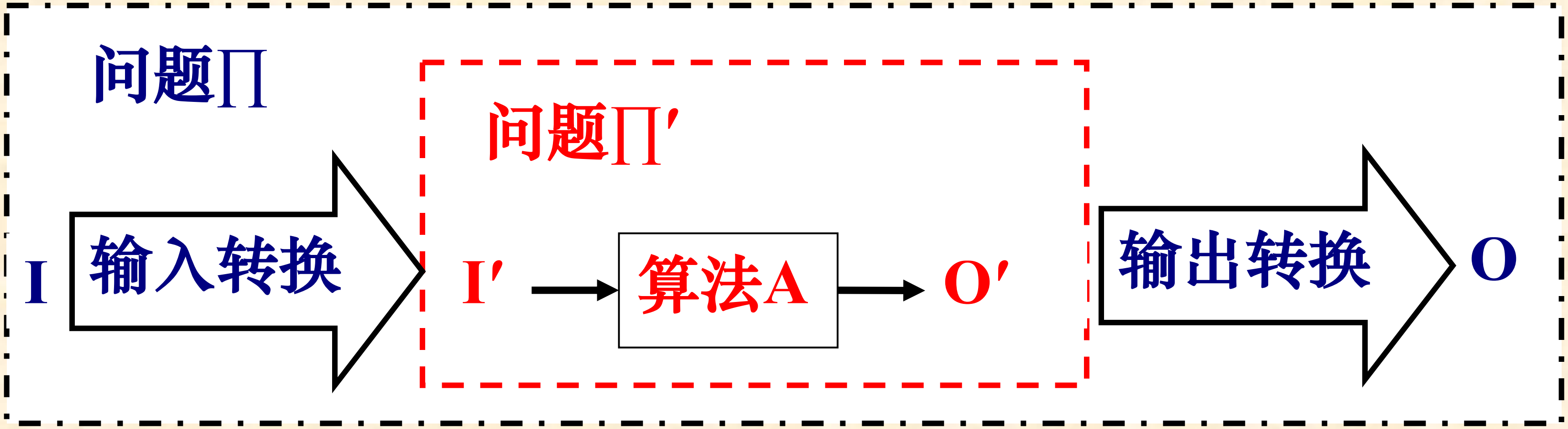
- 因此，人们猜测$P \neq NP$，但是否成立，至今未得到证明。

9.3 NP完全问题

- **NP**完全问题是**NP类问题的子类**，一个具有特殊性质与特殊意义的子类。

问题变换

- **NP**类问题在最坏情况下的时间复杂性一般都是快速增长的指数函数。我们希望能够在**NP**类问题内部找到一种方法，比较两个问题的复杂性。
- 比较两个问题的计算复杂性的方法是问题变换。



多项式时间变换

- 定义9.6(多项式时间变换):若在$O(\tau(n))$时间内完成上述输入/输出转换，则称问题∏以$\tau(n)$时间变换到问题∏′，记为

$$\prod \propto_{\tau(n)} \prod'$$

- 其中，n为问题规模；若在多项式时间内完成上述转换，则称问题∏以多项式时间变换到问题∏′，记为

$$\prod \propto_{poly} \prod'$$

举例：多项式时间变换

排序问题的算法A，输入为$x_1, x_2,..., x_n$, 输出为非降序序列$x_{i1}\le x_{i2} \le ... \le x_{in}$；

配对问题∏：输入两组数$X=(x_1, x_2,..., x_n,)$和$Y=(y_1, y_2,..., y_n,)$，输出是两组元素的非降序依次配对。

求解配对问题，需要进行三个变换

- 将配对问题的输入X,Y变成排序问题的两个输入$I_{1'}, I_{2'}$；
- 应用算法A对$I_{1'}, I_{2'}$分别排序，得到两个排序输出$O_{1'}, O_{2'}$；
- 将两个排序输出$O_{1'}, O_{2'}$转换成配对问题的输出O。

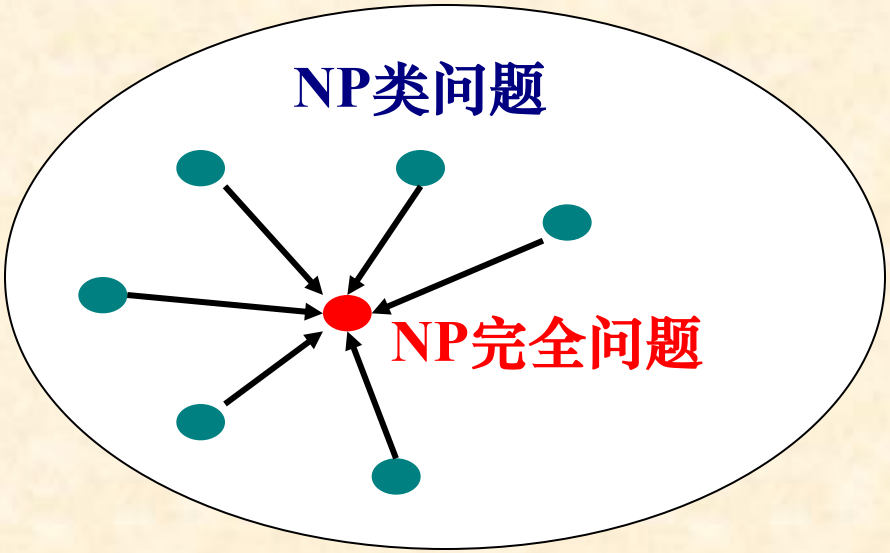
这些操作可以在多项式时间内完成。

多项式时间变换的性质

- **传递性**：设∏、∏’和∏’’是3个判定问题，若$\prod \propto_{\tau(n)} \prod'$，且$\prod' \propto_{\tau(n)} \prod''$，则$\prod \propto_{\tau(n)} \prod''$。

NP完全问题

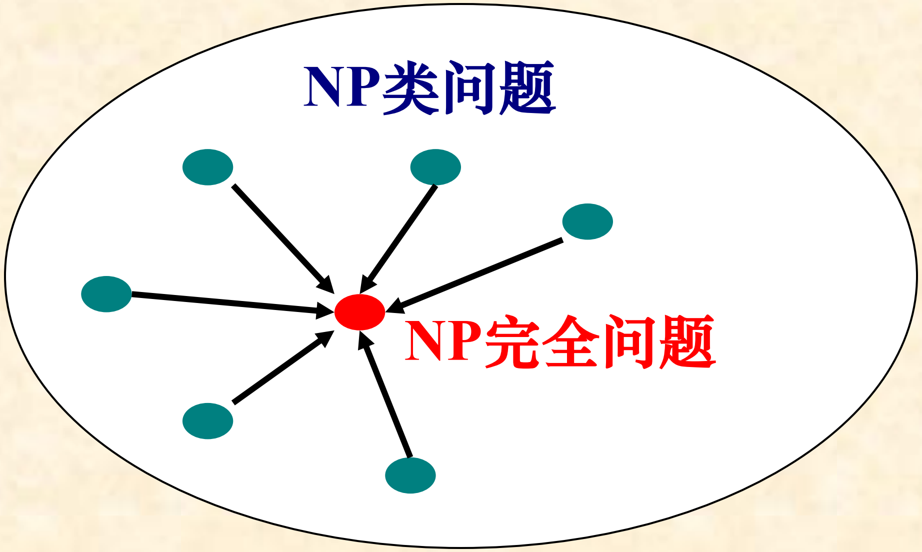
NP完全问题是一类具备如下特殊性质的NP类问题



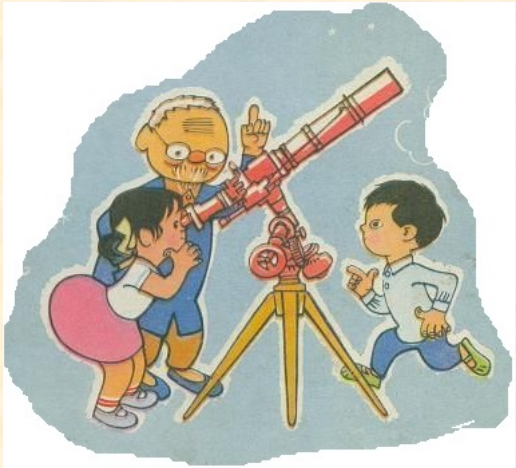
- ∏（该问题本身）就是一个NP类问题
- 每一个NP类问题都可以通过多项式时间化为∏

NP完全问题的定义

- 定义9.7(NP完全问题): 令∏是一个判定问题，如果问题∏属于NP类问题，并且对NP类问题中的每一个问题∏′，都有$\Pi' \propto_{poly} \Pi$，则称判定问题∏是一个NP完全问题(NP complete problem, NPC)。

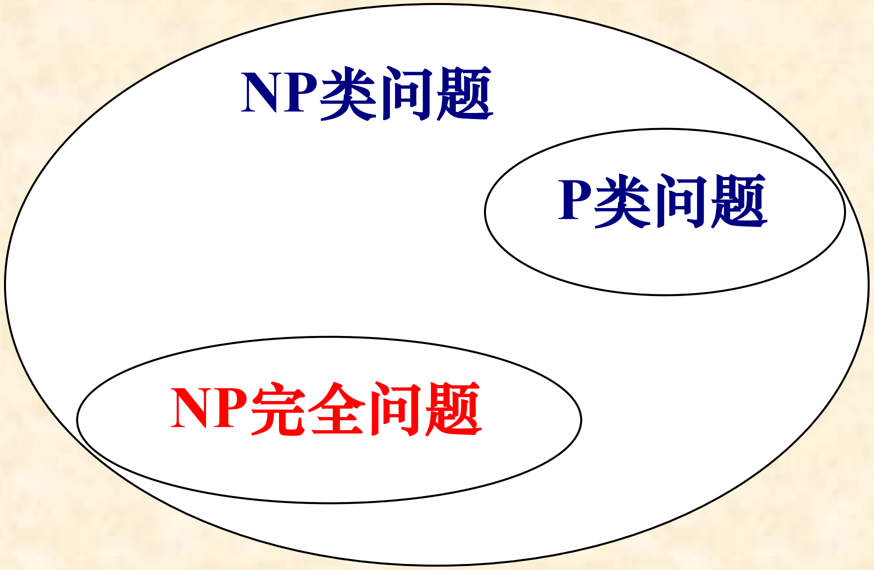


对“NP完全问题”的评述



- NP完全问题是NP类问题中最难的一类问题，至今已经发现了几千个，但一个也没有找到多项式时间算法。
- 如果某一个NP完全问题能在多项式时间内解决，则每一个NP完全问题都能在多项式时间内解决。
- 这些问题也许存在多项式时间算法，因为计算机科学是相对新生的科学，肯定还有新的算法设计技术有待发现；
- 这些问题也许不存在多项式时间算法，但目前缺乏足够的依据来证明这一点。

P类问题、NP类问题、NP完全问题的关系



关于P与NP关系的再思考
--从深层意义

至今没有人能证明是否P ≠ NP。

- 若P=NP，说明NP类中所有问题（包括NP完全问题）都具有多项式时间算法；

- 若P ≠ NP，说明NP类中除P类之外的所有问题（包括NP完全问题）都不存在多项式时间算法。

无论哪种答案，都将为算法设计提供重要的指导和依据。

目前人们普遍猜测：P ≠ NP

最基本的NP完全问题

- **1971年，美国的Cook证明了Cook定理：布尔表达式的可满足性(SAT)问题是NP完全的。**

可满足性问题即合取范式的可满足性问题，来源于许多实际的逻辑推理的应用。合取范式形如$A_1 \wedge A_2 \wedge \ldots \wedge A_n$，其中子句$A_i$ $(1 \le i \le n)$形如：$a_1 \vee a_2 \vee \ldots \vee a_k$, 其中$a_j$称为文字，为布尔变量。SAT问题是指：是否存在一组对所有布尔变量的赋值，使得整个合取范式为真? 例如

$$f = (x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_3 \vee x_4 \vee \bar{x}_5) \wedge (x_1 \vee \bar{x}_3 \vee x_4)$$

当x_1和x_3都为真、其余文字任意赋值时，f值为真。

其他NP完全问题

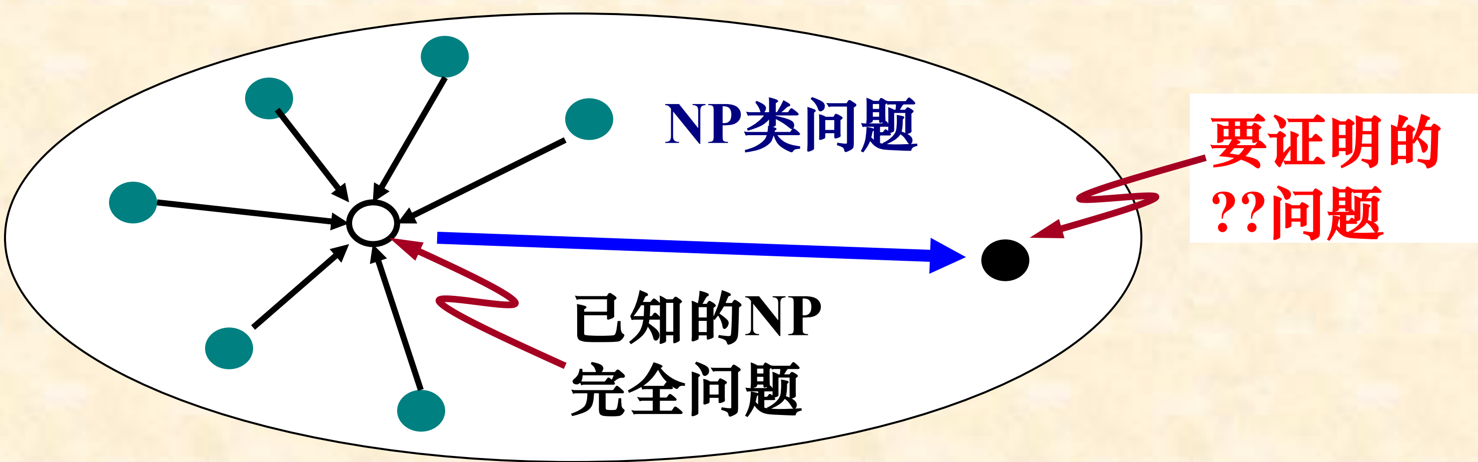
- 可满足性(SAT)问题是第一个被证明的NP完全问题。1972年，Karp证明了十几个问题都是NP完全的。这些NP完全问题的证明思想和技巧，以及利用他们证明的几千个NP完全问题，极大地丰富了NP完全理论。

- 已证明的NP完全问题：

  SAT问题、最大团问题、图着色问题、哈密顿回路问题、旅行商问题、背包问题、最长路径问题、扫雷游戏…

如何证明一个问题是NP完全的？

- 根据NP完全问题的定义（满足的两个性质），显然地，证明需要分两个步骤进行：
  - 证明问题∏是NP类问题；即可以在多项式时间内以确定性算法验证一个任意生成的串，以确定它是否为问题的一个解。
  - 证明NP类问题中的每一个问题都能在多项式时间内变换为问题∏。由于多项式问题变换具有传递性，所以，只需证明一个已知的NP完全问题能够在多项式时间内变换为问题∏.



证明一个问题是NP完全的

--以最大团问题为例

- 团的概念：团是图G=(V,E)的一个完全子图，该子图（有k个顶点）中任意两个顶点都有1条边相连。
- 团问题：对于给定的无向图G=(V,E)和正整数k，是否存在具有k个顶点的团?

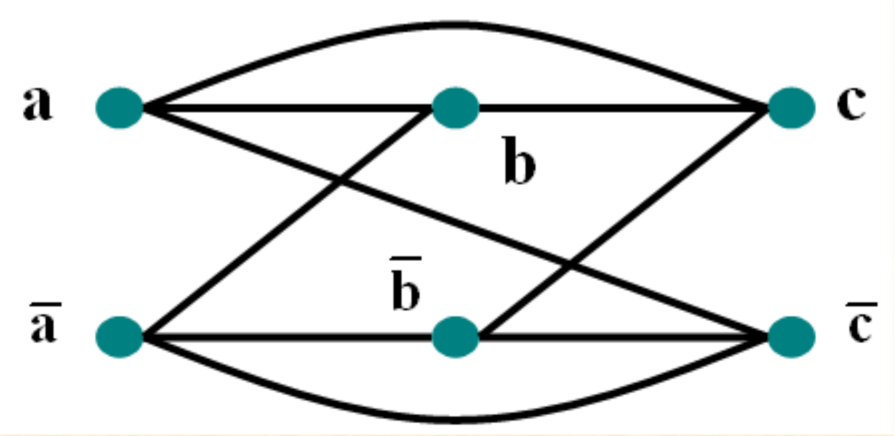
- 下面证明团问题属于NP完全问题，证明分两步：
  - 1) 团问题属于NP类问题
    - 显然，验证图G的一个子图是否构成团只需要多项式时间，所以团问题属于NP类问题。
  - 2) SAT问题$\propto_{poly}$团问题

SAT问题$\propto_{poly}$团问题

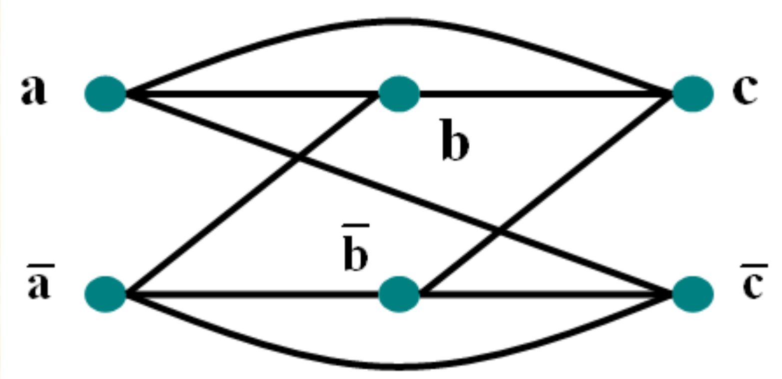
对于任意一个合取范式，按照如下方式构造相应的图G：

例如 $f=(a\vee\bar{b})\wedge(b\vee\bar{c})\wedge(c\vee\bar{a})$

- 图G的每个顶点对应于f中的每个文字，多次出现的重复表示；
- 若图G中两个顶点对应的文字不互补，且不出现在同一子句中，则将其连线。（a-b连线意味着a和b同时为真）



SAT问题$\propto_{poly}$团问题



设f有n个子句，则f可满足

- ⇔ n个子句同时为真
- ⇔每个子句至少1个文字为真 ⇐ 同时为真，则相连
- ⇔G中有n个顶点之间彼此相连
- ⇔G中有n个顶点的团

显然，上述构造图G的方法可在多项式时间内完成，故有：SAT$\propto_{poly}$团问题。

由以上证明可知，团问题是NP完全问题。

NP难问题

难解问题中还有一类问题，虽然也能证明所有的NP类问题可以在多项式时间内变换到问题∏，但并不能证明∏也是NP类问题，所以∏不是NP完全的。但问题∏至少与任意NP类问题有同样的难度，这样的问题称为NP难问题。