# 《计算复杂性理论》
# 第14讲 NP完全问题

段会川
山东师范大学信息科学与工程学院
2015年12月

## 目录





## 概述 P343

在前面的章节中，介绍了算法分析的一些工具和方法；对一些不同类型的问题，讨论了几种典型的算法设计技术；对一些特定的算法进行了描述，并分析了它们的时间复杂性。此外，也说明了如果 $\Pi$ 是任意一个问题，对 $\Pi$ 存在着一个算法，其时间复杂性是 $O(n^k)$（其中，$n$ 是输入规模，$k$ 是非负整数），就认为存在着一个解问题 $\Pi$ 的多项式时间算法。多项式时间算法是一种有效的算法。在现实世界中，有很多问题存在多项式时间算法。但是，有更大量的问题，它们的时间复杂性是以指数函数或排列函数来衡量的，即具有 $O(2^n)$ 以及 $O(n!)$ 的时间复杂性。这一类问题，其计算时间随着输入规模的增长而快速增长，即使是对中等规模的输入，其计算时间也是以世纪来衡量的。因此，通常把存在多项式时间算法的问题，称为易解的问题；而把那些指数时间算法或排列时间算法的问题，称为难解的问题。对于后面这一类问题，人们一直在寻找具有多项式时间的算法。虽然还不能给出使其获得多项式时间的方法，但是却可以证明这些问题之中，有很多问题在计算上是相关的。对这些存在着计算上相关的问题，如果其中之一可以用多项式时间来求解，那么其他所有同类问题也可以用多项式时间来求解；如果其中之一肯定不存在多项式时间算法，那么对与之同类的其他问题，也肯定不会找到多项式时间算法。于是，在这一章，从计算的观点看来，不是意图去找出求解它们的算法，而是着眼于表明它们在计算复杂性之间存在着什么样的关系。



## 概述 P343

在讨论 *NP* 完全问题时，经常考虑的是判定问题。因为判定问题可以容易地表达为语言的识别问题，从而方便地在图灵机上进行求解。实际上，有很多问题都可以作为判定问题来求解。例如，排序问题的判定问题可叙述为：给定一个整数数组，它们是否可以按非降顺序排序；图着色的判定问题可叙述为：给定无向图 $G=(V,E)$，是否可用 $k$ 种颜色为 $V$ 中的每一个顶点分配一种颜色，使得不会有两个相邻顶点具有同一种颜色。

有两类问题：一类是判定问题，另一类是优化问题。判定问题的解只牵涉到两种情况：yes 或 no，优化问题则牵涉到极值问题。但是，可以容易地把判定问题转换为优化问题。例如，图着色的优化问题为：求解为图 $G=(V,E)$ 着色，使相邻两个顶点不会有相同颜色时所需要的最少颜色数目。如果令图 $G$ 的顶点个数为 $n$，彩色数是 $num$，并假定存在着一个图着色判定问题的多项式时间算法 coloring：

```
BOOL coloring(GRAPH G,int n,int num)
```

那么，就可以用下面的方法，利用算法 coloring 来解图着色的优化问题。



## 概述 P343

那么，就可以用下面的方法，利用算法 coloring 来解图着色的优化问题。

```
void chromatic_number(GRAPH G,int n,int &num)
{
    int high,low;
    high = n;
    low = 1;
    while (low<=high) {
        num = (low + high) / 2;
        if (coloring(G,n,num))
            low = mid + 1;
        else
            high = mid -1;
    }
    num = high;
}
```

这相当于一个二叉检索算法，很显然，它只要对算法 coloring 调用 $O(\log n)$ 次，就能找出为图着色的最优彩色数。根据假定，算法 coloring 是一个多项式时间算法，因此该算法也是一个多项式时间算法。这就实现了把图着色的判定问题，转换为图着色的优化问题。

正因为如此，在下面讨论 *NP* 完全问题时，主要以判定问题来进行讨论。



## 目录

## 12.1 P类和NP类问题 P344

**定义 12.1** $A$ 是问题 $\Pi$ 的一个算法。如果在处理问题 $\Pi$ 的实例时，在算法的整个执行过程中，每一步只有一个确定的选择，就说算法 $A$ 是确定性的算法。

前面所讨论的算法，基本上都是确定性的算法，算法执行的每一个步骤，都有一个确定的选择。如果重新用同一输入实例运行该算法，所得到的结果严格一致。

**定义 12.2** 如果对某个判定问题 $\Pi$，存在着一个非负整数 $k$，对输入规模为 $n$ 的实例，能够以 $O(n^k)$ 的时间运行一个确定性的算法，得到 yes 或 no 的答案，则该判定问题 $\Pi$ 是一个 $P$ 类判定问题。

从上面的定义可以看到，$P$ 类判定问题是由具有多项式时间的确定性算法来解的判定问题组成的，因此用 $P$（Polynomial）来表征这类问题。例如，下面的一些判定问题便属于 $P$ 类判定问题。

- 最短路径判定问题 SHORTEST PATH：给定有向赋权图 $G=(V,E)$（权为正整数）、正整数 $k$ 及两个顶点 $s,t \in V$，是否存在着一条由 $s$ 到 $t$、长度至多为 $k$ 的路径。
- 可排序的判定问题 SORT：给定 $n$ 个元素的数组，是否可以按非降顺序排序。



## 12.1 P类和NP类问题 P344

如果把判定问题的提法改变一下，例如把可排序的判定问题的提法改为：给定 $n$ 个元素的数组，是否不可以按非降顺序排序。把这个问题称为不可排序的判定问题 NOT_SORT，则称不可排序的判定问题是可排序的判定问题的补。因此，最短路径判定问题的补是：给定有向赋权图 $G=(V,E)$（权为正整数）、正整数 $k$ 及两个顶点 $s,t \in V$，是否不存在一条由 $s$ 到 $t$、长度至多为 $k$ 的路径。

**定义 12.3** 令 $C$ 是一类问题，如果对 $C$ 中的任何问题 $\Pi \in C$，$\Pi$ 的补也在 $C$ 中，则称 $C$ 类问题在补集下封闭。

**定理 12.1** $P$ 类问题在补集下是封闭的。

**定义 12.4** 令 $\Pi$ 和 $\Pi'$ 是两个判定问题，如果存在一个具有如下性能的确定性算法 $A$，可以用多项式的时间，把问题 $\Pi'$ 的实例 $I'$ 转换为问题 $\Pi$ 的实例 $I$，使得 $I'$ 的答案为 yes，当且仅当 $I$ 的答案是 yes，就说 $\Pi'$ 以多项式时间归约于 $\Pi$，记为 $\Pi' \propto_p \Pi$。

**定理 12.2** $\Pi$ 和 $\Pi'$ 是两个判定问题，如果 $\Pi \in P$，并且 $\Pi' \propto_p \Pi$，则 $\Pi' \in P$。



## 12.1 P类和NP类问题 P345

### 12.1.2 *NP* 类问题

如果有些问题存在着以多项式时间运行的非确定性算法，则这些问题属于 $NP$ 类问题。问题 $\Pi$ 的非确定性算法是由两个阶段组成的：推测阶段和验证阶段。在推测阶段，它对规模为 $n$ 的输入实例 $x$，产生一个输出结果 $y$。这个输出可能是相应输入实例 $x$ 的解，也可能不是，甚至它的形式也不是所希望的解的正确形式。如果再一次运行这个非确定性算法，得到的结果可能和以前得到的结果不一致。但是，它能够以多项式时间 $O(n^i)$（其中，$i$ 是一个非负整数）来输出这个结果。在很多问题中，这一阶段可以按线性时间来完成。

在验证阶段，用一个确定性的算法来验证两件事情：首先，它检查上一阶段所产生的输出 $y$ 是否具有正确的形式。如果不具有正确的形式，这个算法就以答案 no 结束；如果 $y$ 具有正确的形式，则这个算法继续检查 $y$ 是否是问题的输入实例 $x$ 的解。如果它确实是问题实例 $x$ 的解，则以答案 yes 结束；否则，以答案 no 结束。同样，这一阶段的运行时间，也能够以多项式时间 $O(n^j)$（其中，$j$ 也是一个非负整数）来完成。



## 12.1 P类和NP类问题 P346

**例 12.1** 货郎担的判定问题：给定 $n$ 个城市、正常数 $k$ 及城市之间的费用矩阵 $C$，判定是否存在一条经过所有城市一次且仅一次，最后返回初始出发城市且费用小于常数 $k$ 的回路。假定 $A$ 是求解货郎担判定问题的算法。首先，$A$ 用非确定性的算法，在多项式时间内推测存在着这样一条回路，假定它是问题的解。然后，用确定性的算法，在多项式时间内检查这条回路是否正好经过每个城市一次，并返回初始出发城市。如果答案为 yes，则继续检查这条回路的费用是否小于常数 $k$。如果答案仍为 yes，则算法 $A$ 输出 yes，否则输出 no。因此，$A$ 是求解货郎担判定问题的非确定性算法。显然，算法 $A$ 输出 no，并不意味着不存在一条所要求的回路，因为算法的推测可能是不正确的。另一方面，对所有的实例 $I$，算法 $A$ 输出 yes，当且仅当在实例 $I$ 中，至少存在一条所要求的回路。

因此，如果 $A$ 是问题 $\Pi$ 的一个非确定性算法，$A$ 接受问题 $\Pi$ 的实例 $I$，当且仅当对输入实例 $I$ 存在着一个推测，从这个推侧可以得出答案 yes，并且在它的某一次验证阶段的运行中，能够得到答案 yes，则 $A$ 接受 $I$。但是，如果算法的答案为 no，并不意味算法 $A$ 不接受 $I$，因为算法的推测可能是不正确的。



## 12.1 P类和NP类问题 P346

非确定性算法的运行时间，是推测阶段和验证阶段的运行时间的和。若推测阶段的运行时间为 $O(n^i)$，验证阶段的运行时间为 $O(n^j)$，则对某个非负整数 $k$，$k=\max(i,\ j)$，非确定性算法的运行时间为 $O(n^i)+O(n^j)=O(n^k)$。这样一来，可以对 $NP$ 类问题作如下的定义。

**定义 12.5** 如果对某个判定问题 $\Pi$，存在着一个非负整数 $k$，对输入规模为 $n$ 的实例，能够以 $O(n^k)$ 的时间运行一个非确定性的算法，得到 yes 或 no 的答案，则该判定问题 $\Pi$ 是一个 $NP$ 类判定问题。

从上面的定义可以看到，$NP$ 类判定问题是由具有多项式时间的非确定性算法来解的判定问题组成的，因此用 $NP$（Nondeterministic Polynomial）来表征这类问题。对于 $NP$ 类判定问题，重要的是它必须存在一个确定性的算法，能够以多项式的时间来检查和验证在推测阶段所产生的答案。



## 12.1 P类和NP类问题 P346

**例 12.2** 上述解货郎担判定问题 TRAVELING SALESMAN 的算法 $A$：显然，$A$ 可在推测阶段用多项式时间推测出一条回路，并假定它是问题的解；在验证阶段用一个多项式时间的确定性算法，检查所推测的回路是否恰好每个城市经过一次，如果是，再进一步判断这条回路的长度是否小于或等于 $l$，如果是，答案为 yes，否则答案为 no。显然，存在着一个多项式时间的确定性算法，来对推测阶段所作出的推测进行检查和验证。因此，货郎担判定问题是 $NP$ 类判定问题。

**例 12.3** $m$ 团问题 CLIQUE：给定无向图 $G=(V,E)$、正整数 $m$，判定 $V$ 中是否存在 $m$ 个顶点，使得它们的导出子图构成一个 $K_m$ 完全图。

可以这样来为 $m$ 团问题构造非确定性算法：在推测阶段用多项式时间对顶点集生成一组 $m$ 个顶点的子集，假定它是问题的解；然后，在验证阶段用一个多项式时间的确定性算法，验证这个子集的导出子图是否构成一个 $K_m$ 完全图。如果是，答案为 yes；否则，答案为 no。显然，存在着这样的多项式时间的确定性算法，来对前面的推测进行检查和验证。

因此，$m$ 团问题是 $NP$ 类判定问题。

## 12.1 P类和NP类问题 P347

如上所述，$P$ 类问题和 $NP$ 类问题的主要差别如下。

- ◆ $P$ 类问题可以用多项式时间的确定性算法来进行判定或求解。
- ◆ $NP$ 类问题可以用多项式时间的确定性算法来检查和验证它的解。

如果问题 $\Pi$ 属于 $P$ 类，则存在一个多项式时间的确定性算法，来对它进行判定或求解。显然，对这样的问题 $\Pi$，也可以构造一个多项式时间的确定性算法，来验证它的解的正确性。因此，$\Pi$ 也属于 $NP$ 类问题。由此，$\Pi \in P$，必然有 $\Pi \in NP$。所以，$P \subseteq NP$。

反之，如果问题 $\Pi$ 属于 $NP$ 类问题，只能说明存在一个多项式时间的确定性算法来检查和验证它的解，但是不一定能够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，$\Pi$ 不一定属于 $P$ 类问题。于是，人们猜测 $NP \neq P$。但是，这个不等式是成立还是不成立，至今还没有得到证明。



## 目录





## 12.2 NP完全问题 P347

$NP$ 完全问题是 $NP$ 判定问题中的一个子类。对这个子类中的一个问题，如果能够证明用多项式时间的确定性算法来进行求解或判定，那么 $NP$ 中的所有问题都可以通过多项式的确定性算法来进行求解或判定。因此，如果对这个子类中的任何一个问题，能够找到或者能够证明存在着一个多项式时间的确定性算法，那么就有可能证明 $NP = P$。

**定义 12.6** 令 $\Pi$ 是一个判定问题，如果对 $NP$ 中的每一个问题 $\Pi' \in NP$，有 $\Pi' \propto_p \Pi$，就说判定问题 $\Pi$ 是一个 $NP$ 难题。

**定义 12.7** 令 $\Pi$ 是一个判定问题，如果：

（1）$\Pi \in NP$；

（2）对 $NP$ 中的所有问题 $\Pi' \in NP$，都有 $\Pi' \propto_p \Pi$；

则称判定问题 $\Pi$ 是 $NP$ 完全的。

因此，如果 $\Pi$ 是 $NP$ 完全问题，而 $\Pi'$ 是 $NP$ 难题，那么它们之间的差别在于 $\Pi$ 必定在 $NP$ 类中，而 $\Pi'$ 不一定在 $NP$ 类中。有时把 $NP$ 完全问题记为 $NPC$。

**定理 12.3** 令 $\Pi$、$\Pi'$ 和 $\Pi''$ 是 3 个判定问题，若满足 $\Pi'' \propto_p \Pi'$ 及 $\Pi' \propto_p \Pi$，则有 $\Pi'' \propto_p \Pi$。



## 12.2 NP完全问题 P348

**定理 12.3** 令 $\Pi$、$\Pi'$ 和 $\Pi''$ 是 3 个判定问题，若满足 $\Pi'' \propto_p \Pi'$ 及 $\Pi' \propto_p \Pi$，则有 $\Pi'' \propto_p \Pi$。

这个定理表明：归约关系 $\propto_p$ 是传递的。

**定理 12.4** 令 $\Pi$ 和 $\Pi'$ 是 $NP$ 中的两个问题，使得 $\Pi' \propto_p \Pi$。如果 $\Pi'$ 是 $NP$ 完全的，则 $\Pi$ 也是 $NP$ 完全的。

**例 12.4** 已知哈密尔顿回路问题 HAMILTONIAN CYCLE 是一个 $NP$ 完全问题，证明货郎担问题 TRAVELING SALESMAN 也是一个 $NP$ 完全问题。



## 12.2 NP完全问题 P349

### 12.2.2 几个典型的 *NP* 完全问题

下面讨论几个著名的 $NP$ 完全问题。

1. 可满足性问题（SATISFIABILITY）

设布尔表达式 $f$ 是一个合取范式（conjunction normal form，CNF），它是由若干个析取子句的合取构成的；而这些析取子句又是由若干个文字的析取组成；文字则是布尔变元或布尔变元的否定。把前者称为正文字，后者称为负文字。例如，$x$ 是布尔变元，则 $x$ 是正文字，$x$ 的否定 $\neg x$ 是负文字。负文字有时也表达为 $\overline{x}$。下面的例子是一个合取范式：

$$f = (x_2 \vee x_3 \vee x_5) \wedge (x_1 \vee x_3 \vee \overline{x}_4 \vee x_5) \wedge (\overline{x}_2 \vee \overline{x}_3 \vee x_4)$$

如果对其相应的布尔变量赋值，使 $f$ 的真值为真，就说布尔表达式 $f$ 是可满足的。例如，在上式中，只要使 $x_1$、$x_4$ 和 $x_5$ 为真，则表达式 $f$ 为真。因此，这个式子是可满足的。



## 12.2 NP完全问题 P349

可满足性问题的提法是：

判定问题：SATISFIABILITY
输入：CNF 布尔表达式 f
问题：对布尔表达式 f 中的布尔变量赋值，是否可使 f 的真值为真

**定理 12.5** 可满足性问题 SATISFIABILITY 是 $NP$ 完全的。

定理 12.5 称为 Cook 定理。在定理的证明中，如何用合取范式的形式构造一个布尔表达式 $f$，来模拟算法 $A$ 对实例 $I$ 的计算，留待后面叙述。这个定理具有很重要的作用，因为它给出了第 1 个 $NP$ 完全问题，使得对任何问题 $\Pi$，只要能够证明 $\Pi \in NP$，并且 SATISFIABILITY $\propto_p \Pi$，那么 $\Pi$ 就是 $NP$ 完全的。所以，以 SATISFIABILITY 的 $NP$ 完全性为基础，很快又证明了很多其他的 $NP$ 完全问题，逐渐地产生了一棵以 SATISFIABILITY 为根的 $NP$ 完全树。

## 12.2 NP完全问题 P349

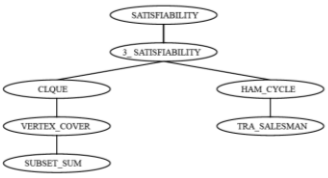
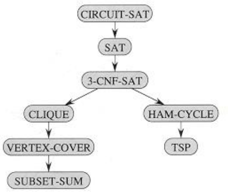



图 12.1 部分的 *NP* 完全问题树






## 12.2 NP完全问题 P350-1

2. 三元可满足性问题（3_SATISFIABILITY）

在合取范式中，如果每个析取子句恰好由 3 个文字组成，则称为三元合取范式或三元 CNF 范式。三元合取范式的可满足性问题 3_SATISFIABILITY 的提法是：

判定问题：3_SATISFIABILITY
输入：三元合取范式 f
问题：对布尔表达式 f 中的布尔变量赋值，是否可使 f 的真值为真

3. 图的着色问题（COLORING）

给定无向图 $G=(V,E)$，用 $k$ 种颜色为 $V$ 中的每一个顶点分配一种颜色，使得不会有两个相邻顶点具有同一种颜色。此问题称为图的着色问题 COLORING。图着色问题的提法是：

判定问题：COLORING
输入：无向图 G=(V,E)，正整数 k≥1
问题：是否可用 k 种颜色为图 G 着色



## 12.2 NP完全问题 P352-3

4. 团问题（CLIQUE）

给定一个无向图 $G=(V,E)$ 和一个正整数 $k$，$G$ 中具有 $k$ 个顶点的完全子图，称为 $G$ 的大小为 $k$ 的团。则团判定问题的提法是：

判定问题：CLIQUE
输入：无向图 G，正整数 k
问题：G 中是否包含有大小为 k 的团

5. 顶点覆盖问题（VERTEX COVER）

给定一个无向图 $G=(V,E)$ 和一个正整数 $k$，若存在 $V'\subseteq V$，$|V'|=k$，使得对任意的 $(u,v)\in E$，都有 $u\in V'$ 或 $v\in V'$，则称 $V'$ 为图 $G$ 的一个大小为 $k$ 的顶点覆盖。顶点覆盖问题的提法是：

判定问题：VERTEX COVER
输入：无向图 G=(V,E)，正整数 k
问题：G 中是否存在一个大小为 k 的顶点覆盖



## 12.2 NP完全问题 P355

下面是另外一些 *NP* 完全问题。

（1）三着色问题 3_COLORING：给定无向图 $G=(V,E)$，是否可用 3 种颜色来为图 $G$ 着色，使得图中不会有两个邻接顶点具有同一种颜色。

（2）独立集问题 INDEPENDENT SET：给定无向图 $G=(V,E)$，是否存在一个大小为 $k$ 的独立集 $S$。其中，$S\subseteq V$。若 $S$ 中任意两个顶点都不互相邻接，则称 $S$ 是图 $G$ 的独立集。

（3）哈密尔顿回路问题 HAMILTONIAN CYCLE：给定无向图 $G=(V,E)$，是否存在一条简单回路，使得每个顶点经过一次且仅一次。

（4）划分问题 PARTITION：给定一个具有 $n$ 个整数的集合 $S$，是否能把 $S$ 划分成两个子集 $S_1$ 和 $S_2$，使得 $S_1$ 中的整数之和等于 $S_2$ 中的整数之和。

（5）子集求和问题 SUBSET SUM：给定整数集 $S$ 和整数 $t$，是否存在 $S$ 的一个子集 $T\subseteq S$，使得 $T$ 中的整数之和为 $t$。

（6）装箱问题 BIN PACKING：给定大小为 $s_1,s_2,\cdots,s_n$ 的物体，箱子的容量为 $C$，以及一个正整数 $k$，是否能够用 $k$ 个箱子来装这 $n$ 个物体。

（7）集合覆盖问题 SET COVER：给定集合 $S$ 和由 $S$ 的子集构成的集类 $A$，以及 1 和 |A|之间的整数 $k$，在 $A$ 中是否存在 $k$ 个元素，它们的广义并为 $S$。

（8）多处理器调度问题 MULTIPROCESSOR SCHEDULING：给定 $m$ 个性能相同的处理器、$n$ 个作业 $J_1,J_2,\cdots,J_n$、每一个作业的运行时间 $t_1,t_2,\cdots,t_n$，以及时间 $T$，是否可以调度这 $m$ 个处理器，使得它们最多在时间 $T$ 里，完成这 $n$ 个作业。



## 12.3 co_NP类和NPI类问题 355-6

**定义 12.8** $co\_NP$ 类问题是由一些 $NP$ 类问题的补组成的。

有些 $NP$ 类问题，它们的补可能不属于 $NP$，由这些 $NP$ 类问题的补组成了 $co\_NP$ 类。

例如，哈密尔顿回路问题 HAMILTONIAN CYCLE 的补是：给定图 $G=(V,E)$，是否不存在一条每个顶点只经过一次且仅一次的回路。这个问题的解，可能需要花费 $(n-1)!$ 时间，对所有 $(n-1)!$ 种可能性进行判断。因此，有理由猜想，可能不存在一个非确定性的算法，可以用多项式的时间，而不是用 $(n-1)!$ 时间来解这个问题。所以，哈密尔顿回路问题的补可能不属于 $NP$，而把它归类于 $co\_NP$。

可满足性问题的补是：给定一个布尔公式 $f$，是否对公式中的 $n$ 个布尔变量的真值赋值，都不能使公式 $f$ 的真值为真，即公式 $f$ 是不可满足的。同样，解可满足性问题的补，可能需要对 $n$ 个布尔变量的 $2^n$ 种可能的赋值进行判断，因此，也可能不存在一个非确定性的算法，可以用多项式的时间，而不是用 $2^n$ 时间来解这个问题。因此，可满足性问题的补可能不属于 $NP$，而把它归类于 $co\ \ NP$。



## 12.3 co_NP类和NPI类问题 356

可满足性问题的补是：给定一个布尔公式 $f$，是否对公式中的 $n$ 个布尔变量的真值赋值，都不能使公式 $f$ 的真值为真，即公式 $f$ 是不可满足的。同样，解可满足性问题的补，可能需要对 $n$ 个布尔变量的 $2^n$ 种可能的赋值进行判断，因此，也可能不存在一个非确定性的算法，可以用多项式的时间，而不是用 $2^n$ 时间来解这个问题。因此，可满足性问题的补可能不属于 $NP$，而把它归类于 $co\_NP$。

由此，人们提出了第 2 个猜想：$co\_NP\neq NP$。

类似于 $NP$ 完全问题，$co\_NP$ 完全问题的定义如下。

**定义 12.9** 令 $\Pi$ 是一个判定问题，如果

（1）$\Pi\in co\_NP$；

（2）对所有的 $\Pi'\in co\_NP$，都有 $\Pi'\propto_p\Pi$；

则问题 $\Pi$ 对 $co\_NP$ 是完全的。

**定理 12.6** 问题 $\Pi$ 是 $NP$ 完全的，当且仅当 $\Pi$ 的补 $\overline{\Pi}$ 对 $co\_NP$ 是完全的。

## 12.3 co_NP类和NPI类问题 357

在 *NP* 中，有些问题的补也有可能是属于 *NP* 的。例如，下面的素数问题 PRIME。

判定问题：PRIME
输入：整数 k≥2
问题：k 是否是一个素数

合数问题 COMPOSITE：

判定问题：COMPOSITE
输入：整数 k≥4
问题：是否存在两个整数 p≥2 及 q≥2，使得 pq = k

则素数问题 PRIME 和合数问题 COMPOSITE 互补，而素数问题和合数问题都是 *NP* 问题，因此，它们的补都在 *NP* 中。素数问题不可能是 *NP* 完全的，否则，它的补 COMPOSITE 对 $co\_NP$ 就是完全的了。如果这样，$co\_NP$ 中的所有问题都可归约于 COMPOSITE，而 COMPOSITE 是 *NP* 问题。这就意味着 $co\_NP$ 中的所有问题，也都是 *NP* 问题。于是，将得出这样的结果：$co\_NP = NP$，而这种可能性是很小的。



## 12.3 co_NP类和NPI类问题 357

由于这两个问题是 *NP* 完全的可能性很小，也因为它们不会属于 *P* 类问题，就把这样的问题叫做 *NPI* 问题，表示为 $NP\_Intermediate$。

本章所讨论的 4 种复杂性问题之间的关系如图 12.4 所示。从图中可以看到，*NPI* 类在补下是封闭的，并且有：$P \subseteq NPI$。

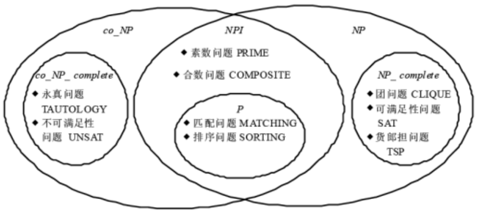
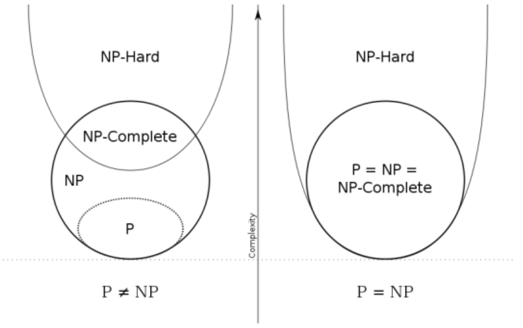



图 12.4 4 种复杂性问题之间的关系



## 问题复杂性类的关系图




目前已知的NPC问题已经有上千个。



## 目录





## 学习要点

1. 算法伪代码描述的一般准则
2. 渐近上界$O$、渐近下界$\Omega$及渐近精确界$\Theta$的定义及极限判断准则
3. 0-1背包问题、TSP问题的定义及其状态空间树
4. 0-1背包问题、TSP问题的穷举算法
5. 合并排序、堆排序、快速排序的伪代码及复杂度推导，基于比较的排序算法复杂度总结
6. 基于比较的排序算法的复杂度上界及其证明
7. Karatsuba乘法算法的伪代码及其复杂度
8. 递推式的主定理求解法及其在求解Karatsuba乘法算法、合并排序算法复杂度中的应用



## 学习要点

9. Dijkstra最短路径算法的伪代码、复杂度及其贪心选择性质和最优子结构性质的证明
10. 0-1背包问题基于价值重量比的贪心算法及其复杂度，举例说明该算法不能保证获得最优解
11. TSP问题的最近邻贪心算法及其复杂度，举例说明该算法不能保证获得最优解
12. 设计DP算法的基本步骤
13. 最长递增子序列问题，描述其最优子结构性质，给出其DP算法并说明其复杂度
14. 两个字符串间的Levenshtein距离，描述其最优子结构性质，给出其DP算法并说明其复杂度

## 学习要点

15. 回溯法求解问题的基本步骤
16. 0-1背包问题和TSP问题回溯法的约束条件和限界条件，伪代码描述，最坏情况下的复杂度
17. 0-1背包问题分支限界法的约束条件和限界条件，运行示例
18. 费用矩阵及其规约，TSP问题分支限界法的约束条件和限界条件，运行示例
19. NP完全问题
    - 判定性问题与优化问题、P类和NP类问题、确定性算法和非确定性算法、多项式规约、NP难问题和NP完全问题、问题复杂性关系图、NP完全问题规约树



# The End

Thanks !