算法概论

第七讲：$\mathcal{NP}$-完全问题简介

薛健

Last Modified: 2019.1.12

主要内容

1 基本概念

基本问题

- 回顾：算法复杂度的渐近阶

	解题时间 (假定单位时间为微秒)			
输入规模 (n)	$33n$	$460n\lg n$	$13n^2$	2^n
10	0.00033sec.	0.015sec.	0.0013sec.	0.001sec.
100	0.0033sec.	0.3sec.	0.13sec.	4×10^{16}yr.
1,000	0.033sec.	4.5sec.	13sec.	
10,000	0.33sec.	61sec.	22min.	
100,000	3.3sec.	13min.	1.5days	

- 到目前为止我们所接触的问题都可以找到时间复杂度在 $O(n^c)$ 的算法来解决
- 是否存在这样的问题：解决该类问题的算法复杂度 (**问题的复杂度**) 在 $O(c^n)$ 或更高？

判定问题

- 涉及到上述疑问的一般都是优化问题 (optimization problem)，或更确切地讲是**组合优化问题** (combinatorial optimization problem)
- 这类问题可以重新描述成某个**判定问题** (decision problem)：有“是 (yes)”或“否 (no)”两种可能回答的问题，其表述一般包含两个部分
  1. **实例描述** (instance description) 部分：定义输入中所应包含的信息
  2. **问题** (question) 部分：提出需要判定“是”或“否”的问题，问题中包含实例描述中定义的变量
- 判定问题根据给定输入和问题的正确答案给出“是”或“否”的回答，因此判定问题可抽象为从所有可能输入的集合到 {yes, no} 的映射

问题举例

Definition 1.1 (图着色和着色数). 图 $G = (V, E)$ 的**着色** (coloring) 是指映射 $C : V \to S$，其中 S 是一个颜色的有限集合，并且如果 $vw \in E$，则 $C(v) \neq C(w)$，即：相邻顶点颜色不同；图 G 的**着色数** (chromatic number) 是对 G 进行着色所需的最少颜色数，记作 $\chi(G)$。

Problem 1.2 (图着色问题). 对给定无向图 $G = (V, E)$ 进行着色

优化问题：给定图 G，确定其着色数 $\chi(G)$，并给出对应的着色方案。

判定问题：给定图 G 和正整数 k，是否存在 G 的一种着色最多使用 k 种颜色？*(若存在，称 G 是 k-*__可着色的__ *$(k$-colorable$)$)*。

Problem 1.3 (哈密尔顿环路和哈密尔顿路径问题). **哈密尔顿环路/路径** *(Hamiltonian cycle / path)* 是无向图中通过所有顶点恰一次的简单环路/路径。

判定问题：给定无向图 G 中是否存在哈密尔顿环路 *(路径)*？

Problem 1.4 (货郎担问题). **货郎担问题** *(Traveling Salesperson Problem, TSP)* 也可以称作**最短周游问题** *(minimum tour problem)*，是著名的易于描述而难以解决的问题之一：某推销员要到其商品销售区域的各个城市推销商品，为节约开销提高效率，他需要从其所在地出发经过每个城市一次然后回到原所在地，且所花开销最少。

优化问题：给定带权完全图，寻找其中具有最小权值的哈密尔顿环路。

判定问题：给定带权完全图和数 k，图中是否存在一条权值小于等于 k 的哈密尔顿环路？

Problem 1.5 (子集和问题 (subset sum)). 给定一个正整数 C 和 n 个大小分别为正整数 $s_1, \cdots, s_n$ 的对象

优化问题：从 n 个对象的全集中选出包含对象数最多的一个子集，使得其中对象的和至多为 C。

判定问题：是否存在一个包含 k 个对象的子集，其中对象大小的和不大于 C？

Problem 1.6 (背包问题 (knapsack)). 有一个容量为 C 的背包和 n 个大小分别为 $s_1, \cdots, s_n$、价值分别为 $p_1, \cdots, p_n$ 的物品

优化问题：寻找总价值最大且能放进背包中的物品子集。

判定问题：给定正数 k，是否存在某个物品子集能够全部放入背包，且总价值不低于 k？

可满足性问题 (Satisfiability Problem)

- 一个命题 (布尔) 变量 (propositional (boolean) variable) 可被赋予 true 或 false 两种值，命题变量 v 的否定 $\bar{v}$ 值为 true 当且仅当 v 的值为 false
- 一个命题公式 (formula) 是由命题变量、命题常量或者由布尔运算符及其运算对象 (命题公式) 所构成的表达式
- **合取范式** (Conjunctive Normal Form, CNF) 是命题公式的一种常见表达方式，一个命题公式的合取范式由以布尔“与”操作符 ($\wedge$) 连接的一个子句序列构成，其中，每个**子句** (clause) 由以布尔“或”操作符 ($\vee$) 连接的一个命题变量 (或其否定变量) 序列构成。如：

$$(p \vee q \vee s) \wedge (\bar{q} \vee r) \wedge (\bar{p} \vee r) \wedge (\bar{r} \vee s) \wedge (\bar{p} \vee \bar{s} \vee \bar{q})$$

可满足性问题 (cont.)

- 对一个给定命题变量集合的每个元素赋予布尔值称为一个**真值指派** (truth assignment)，若对一个合取范式进行的真值指派使整个范式的值为 true，则称该真值指派**满足** (satisfy) 这个合取范式
- 显然，一个合取范式被满足当且仅当其每个子句为 true，而一个子句为 true 当且仅当至少一个命题变量 (或否定变量) 为 true

Problem 1.7 (可满足性问题). 给定一个合取范式，是否存在一个可满足该范式的真值指派？

▷ 该问题被称为**合取范式可满足性问题** *(CNF-satisfiability, CNF-SAT)* 或直接叫做**可满足性问题**，该问题在 $\mathcal{NP}$-完全问题的研究中扮演了非常重要的角色

2 $\mathcal{P}$ 问题和 $\mathcal{NP}$ 问题

$\mathcal{P}$ 问题

Definition 2.1 (以多项式为界 (polynomially bounded))**.** 以多项式为界的算法是指其最坏情况复杂度上界是其输入规模的多项式函数。

Definition 2.2 ($\mathcal{P}$ 问题)**.** $\mathcal{P}$ 问题是指以多项式为界的判定问题，即该问题可以在多项式时间内给出答案。

- 以多项式为判别界限的理由：
  - 并非所有的 $\mathcal{P}$ 问题有高效的算法来解决，但可以说不属于 $\mathcal{P}$ 类的问题必定时间复杂度较高甚至实际上无法解决
  - 多项式具有某种“闭包”性质，即多个多项式的加、乘、复合运算结果仍然是多项式，这意味着由多个多项式复杂度的简单算法合成的复杂算法，其复杂度仍然是以多项式为界的
  - 以多项式为界与具体的计算模型和实现环境无关

非确定算法

- 判定问题的输入和其一个可能的解决方案 (certificate) 可以用一个有限集中的符号所构成的串来描述，对于特定问题，每个符号的含义需要一组约定 (conventions) 来定义，这组约定称为问题的编码 (encoding)，则检查一个解决方案是否符合问题描述只需检查其对应符号串是否符合编码规则 (是否有意义) 和问题的判定准则。

Definition 2.3 (非确定算法 (nondeterministic algorithm))**.** 一个**非确定算法**包含两个阶段和一个输出步骤：

1. **“猜测** (guessing)**”阶段：**在给定存储区域写出一个完整符号串 s；
2. **确定“验证”** (deterministic “verifying”) **阶段：**执行一个确定的子过程，根据输入 *input* 和解决方案 s (或忽略 s) 返回结果 true 或 false (或进入无限循环)，即：检查 s 在当前输入下是否是判定问题的一个肯定 (yes) 答案
3. **输出步骤：**如果验证阶段返回结果为 true，则算法输出 yes，否则算法无输出

非确定算法 (cont.)

Algorithm `NondetA`(String *input*)

```
1 String s ← GetCertif();
2 checkOK ← VerifyA(input, s);
3 if checkOK then 输出 “yes” ;
```

- 算法的总执行步数为 `GetCertif` 和 `VerifyA` 的执行步数之和
- 对同一输入 x，上述算法执行结果可能不一样 (依赖于 s)
- 判定问题对输入 x 的回答为 yes 当且仅当上述算法某次执行 (针对某个 s) 给出了 yes 输出，否则回答为 no，即对任意 s，算法执行无输出结果

$\mathcal{NP}$ 问题

Definition 2.4. $\mathcal{NP}$ **问题**是指存在以多项式为界的非确定算法的判定问题。$\mathcal{NP}$ 取自 “**N**ondeterministic **P**olynomially bounded”。

Theorem 2.5. 图着色问题、哈密尔顿环路/路径问题、货郎担问题、子集和问题、背包问题、可满足性问题都是 $\mathcal{NP}$ 问题。

Theorem 2.6. $\mathcal{P} \subseteq \mathcal{NP}$

Proof. 所有 $\mathcal{P}$ 问题的确定算法 (deterministic algorithm，即普通意义上的算法) 都可以看作一种特殊的非确定算法，即将其作为非确定算法第二阶段的 `VerifyA`，若其输出为 yes，则 `VerifyA` 返回 true，否则返回 false，而第一阶段的 `GetCertif` 什么都不做，这样只要确定算法以多项式为界，则其对应的非确定算法也以多项式为界。 □

$\mathcal{P} \stackrel{?}{=} \mathcal{NP}$

- $\mathcal{P} = \mathcal{NP}$ 还是 $\mathcal{P} \subset \mathcal{NP}$，即：是否有这样的问题，它存在以多项式为界的非确定算法，但无法在多项式时间内由确定算法解决
- 对于一个 $\mathcal{NP}$ 问题，如果我们取遍所有可能的解决方案 s 则可以对判定问题给出确定的 yes 或 no 回答，但可能的解决方案数目很大，若 `GetCertif` 给出的解决方案所对应的符号串长度至多为多项式 $p(n)$，而符号集包含的符号数为 c，则可能的解决方案数目将达到 $c^{p(n)}$！虽然有时候我们可以采用某些技巧使得不需要检查所有可能的解决方案就可以得到结果，但找到这样的技巧并不是一件容易的事情
- 因此，普遍认为 $\mathcal{NP}$ 应该是一个比 $\mathcal{P}$ 大得多的集合，但是迄今为止仍然没有找到一个 $\mathcal{NP}$ 问题可以证明其不是 $\mathcal{P}$ 问题，即存在大量的问题其多项式复杂度的算法未知，而已知的该问题的复杂度下界又不高于多项式
- $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$ 仍然是一个开放的问题。

注意输入规模

- 要判定一个问题是否是 $\mathcal{P}$ 问题，需特别留心其输入规模
- 例如：给定一个正整数 n，判定其是否是质数，我们有一个非常简洁的算法：

Algorithm `IsPrime`(n)

```
1 r ← yes;
2 for i ← 2 to n − 1 do
3 |   if (n mod i) = 0 then r ← no; break;
4 end
5 return r;
```

- $(n \mod i)$ 可以在 $O(\log^2(n))$ 时间内完成，则该算法复杂度似乎不超过 $O(n^2)$
- 质数判定问题是否是 $\mathcal{P}$ 问题？
- 对大整数 n 其输入规模是 $h \in \Theta(\log_b n)$，该算法的复杂度为 b^h！

实际上质数判定问题是 $\mathcal{P}$ 问题，有学者在 2002 年提出了质数判定的多项式复杂度算法，即著名的 AKS 质数判定算法。

AKS 质数判定算法 (Agrawal-Kayal-Saxena primality test) 由三位印度坎普尔理工学院 (Indian Institute of Technology Kanpur) 的计算机科学家 Manindra Agrawal，Neeraj Kayal 和 Nitin Saxena 在 2002 年 8 月提出，发表在名为 “PRIMES is in $\mathcal{P}$ ” 的论文中，他们由于这项工作获得了很多奖，包括 2006 年的哥德尔奖 (Gödel Prize) 和 2006 年的 Fulkerson 奖 (Fulkerson Prize)。

再探欧几里德算法

Algorithm GCD(a, b)

```
while b ≠ 0 do
    t ← b;
    b ← a mod b;
    a ← t;
end
return a;
```

递归版本：

Algorithm GCD(a, b)

```
if b = 0 then return a ;
else return GCD(b, a mod b) ;
```

计算序列：

$$\begin{aligned} a &= q_0 b + r_0 \\ b &= q_1 r_0 + r_1 \\ r_0 &= q_2 r_1 + r_2 \\ r_1 &= q_3 r_2 + r_3 \\ &\vdots \end{aligned}$$

第 k 步计算：

$$r_{k-2} = q_k r_{k-1} + r_k$$

当 $r_N = 0$ 时停止：

$$\mathrm{GCD}(a, b) = r_{N-1}$$

时间复杂度分析*

- 循环步数：
  - 需要 N 次循环的最小自然数 a, b $(a > b > 0)$ 分别为斐波那契数 F_{N+2} 和 F_{N+1} (Gabriel Lamé 1844 年用数学归纳法证明，计算复杂性理论的发端，斐波那契数的第一个实际应用)
  - 当 b 足够大时，$b \geq F_{N+1} > F_N \approx \phi^N / \sqrt{5}$，$\phi$ 为黄金分割数 $\frac{1+\sqrt{5}}{2}$
  - $N < \log_\phi \sqrt{5} b \approx 4.785 \log_{10} b + 1.6723 < 5h$
- 单步开销：
  - 对大整数来说，用试商的方法求商和余数，其复杂度为 $O(hl)$，h 是除数的位数，l 是商的位数
  - 设 $r_0, r_1, \cdots, r_{N-1}$ 的位数分别为 $h_0, h_1, \cdots, h_{N-1}$
  - 第 k 步的开销为 $O(h_{k-1}(h_{k-2} - h_{k-1} + 1))$
  - N 步的开销总和为

$$\begin{aligned} O(\sum_{i<N} h_{i-1}(h_{i-2} - h_{i-1} + 1)) &\subseteq O(h \sum_{i<N} (h_{i-2} - h_{i-1} + 1)) \\ &\subseteq O(h(h + N)) \subseteq O(h^2) = O(\log^2 b) \end{aligned}$$

*Refers to "*The Art of Computer Programming - Volume 2: Seminumerical Algorithms (3rd Edition)*" by Donald E. Knuth.

大整数的质因数分解

- 求最大公约数也可以通过分别对两个整数进行质因数分解来求
- **整数的质因数分解** (integer factorization 或 prime factorization)：将合数分解为一系列质数的积
- 当待分解的整数 n 很大时，尚无公开的有效算法
- 目前公认的最好算法是 GNFS (General Number Field Sieve) 算法，但其复杂度仍为指数级：

$$O\left(\exp\left(\left(\tfrac{64}{9}\log n\right)^{\frac{1}{3}}(\log\log n)^{\frac{2}{3}}\right)\right)$$

即：

$$O\left(\exp\left(\left(\tfrac{64}{9}h\right)^{\frac{1}{3}}(\log h)^{\frac{2}{3}}\right)\right)$$

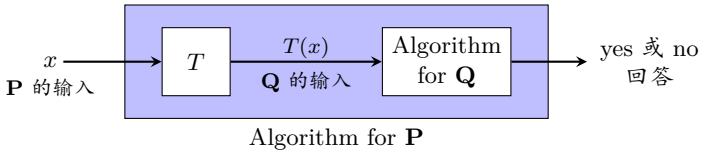
$h=\log n$ 为 n 的位数

- 多数加密协议的安全性均建立在“大整数分解的困难性”基础上

3 $\mathcal{NP}$-完全问题

问题的归约 (Reduction)

- $\mathcal{NP}$-完全问题 ($\mathcal{NP}$-complete problem) 用来描述 $\mathcal{NP}$ 问题中最难的那一类问题，一旦发现一个 $\mathcal{NP}$-完全问题存在以多项式为界的算法，则可以断定每一个 $\mathcal{NP}$ 问题都存在以多项式为界的算法 ($\Rightarrow \mathcal{P}=\mathcal{NP}$)
- 前面所举的问题实例实际上都属于 $\mathcal{NP}$-完全问题
- $\mathcal{NP}$-完全问题正式的定义采用多项式归约的方式给出：已有问题 **Q** 的算法，要解决问题 **P**，若存在某种转换函数 T，能够将问题 **P** 的任一输入 x 转换为问题 **Q** 的输入 $T(x)$，则我们就得到了解决问题 **P** 的算法，对输入 x 的输出为 yes 当且仅当 **Q** 的算法对输入 $T(x)$ 的输出为 yes



多项式归约

Definition 3.1. 设 T 是判定问题 **P** 的输入集合到判定问题 **Q** 的输入集合的函数，如果 T 满足下列条件，则称 T 为**多项式归约** (polynomial reduction)：

1. T 的计算时间以多项式为界；
2. 对每个输入 x，若 x 对 **P** 的输出为 yes，则 $T(x)$ 对 **Q** 的输出也为 yes；
3. 对每个输入 x，若 x 对 **P** 的输出为 no，则 $T(x)$ 对 **Q** 的输出也为 no；或者等价地：对每个输入 x，若 $T(x)$ 对 **Q** 的输出为 yes，则 x 对 **P** 的输出也为 yes；

如果存在这样的 T，则称问题 **P** **可多项式归约到** (polynomially reducible) 到 **Q**，记作：$\mathbf{P}\leq_P\mathbf{Q}$。

Theorem 3.2. 如果 $\mathbf{P}\leq_P\mathbf{Q}$，并且 **Q** 属于 $\mathcal{P}$ 问题，则 **P** 也属于 $\mathcal{P}$ 问题。

$\mathcal{NP}$-完全问题

Definition 3.3 ($\mathcal{NP}$-难问题和 $\mathcal{NP}$-完全问题)**.** 若任一 $\mathcal{NP}$ 问题 $\mathbf{P}$ 可多项式归约到问题 $\mathbf{Q}$，则称问题 $\mathbf{Q}$ 为 $\mathcal{NP}$-**难问题** ($\mathcal{NP}$-hard problem)；若同时问题 $\mathbf{Q}$ 也属于 $\mathcal{NP}$ 问题，则称问题 $\mathbf{Q}$ 为 $\mathcal{NP}$-**完全问题** ($\mathcal{NP}$-complete problem)。

Theorem 3.4. 若存在某个 $\mathcal{NP}$-完全问题属于 $\mathcal{P}$ 问题，则 $\mathcal{P} = \mathcal{NP}$。

- 要证明某一个判定问题 $\mathbf{Q}$ 属于 $\mathcal{NP}$-完全问题，首先要证明它是 $\mathcal{NP}$-难问题，即所有的 $\mathcal{NP}$ 问题 (包括未知的) 都可多项式归约到 $\mathbf{Q}$，*Mission Impossible!*
- 多项式归约具有传递性，这意味着只要有一个问题被证明属于 $\mathcal{NP}$-完全问题，对其他问题的判断就简单了

寻找 $\mathcal{NP}$-完全问题

Theorem 3.5 (Cook 定理)**.** 合取范式可满足性问题是 $\mathcal{NP}$-完全问题。

[1] Stephen A. Cook The Complexity of Theorem-Proving Procedures Proceedings of the third annual ACM symposium on Theory of computing, pages 151–158, 1971

Theorem 3.6. 图着色问题、哈密尔顿环路/路径问题、货郎担问题、子集和问题、背包问题都是 $\mathcal{NP}$-完全问题。

- 证明判定问题 $\mathbf{Q} \in \mathcal{NP}$ 是 $\mathcal{NP}$-完全问题：
  1. 找到一个合适的 $\mathcal{NP}$-完全问题 $\mathbf{P}$，则 $\forall \mathbf{R} \in \mathcal{NP}$，$\mathbf{R} \leq_P \mathbf{P}$
  2. 证明 $\mathbf{P} \leq_P \mathbf{Q}$
  3. 则 $\forall \mathbf{R} \in \mathcal{NP}$，$\mathbf{R} \leq_P \mathbf{Q}$，即 $\mathbf{Q}$ 也是 $\mathcal{NP}$-完全问题

证明实例

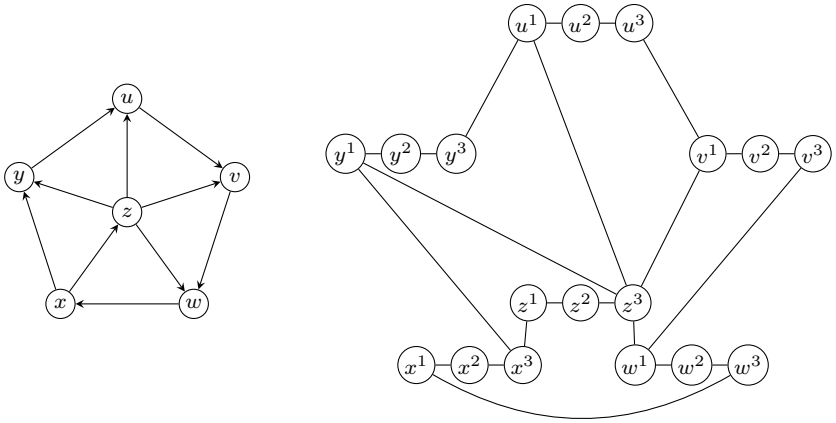
Theorem 3.7. 已知有向图的哈密尔顿环路问题是 $\mathcal{NP}$-完全问题，证明无向图的哈密尔顿环路问题也是 $\mathcal{NP}$-完全问题。

Proof. 显然，无向哈密尔顿环路问题属于 $\mathcal{NP}$ 问题，因此只要证明有向图的哈密尔顿环路问题可多项式规约到无向图的哈密尔顿环路问题 (已知有向图的哈密尔顿问题是 $\mathcal{NP}$-完全问题)。

令 $G = (V, E)$ 是包含 n 个顶点的有向图，现将其转换到无向图 $G' = (V', E')$：对每个 $v \in V$，V' 中包含 3 个顶点 v^1, v^2, v^3 与之对应，E' 中包含两条无向边 v^1v^2, v^2v^3 与之对应，对 E 中的每条边 vw，E' 中包含无向边 v^3w^1 与之对应，显然，该转换所需时间以多项式为界，若 $|V| = n, |E| = m$ 则 $|V'| = 3n, |E'| = 2n + m$。

若 G 中有一条有向哈密尔顿环路 $v_1, \cdots, v_n$，则 $v_1^1, v_1^2, v_1^3, v_2^1, v_2^2, v_2^3, \cdots, v_n^1, v_n^2, v_n^3$ 是 G' 中的一条 (无向) 哈密尔顿环路；另一方面，若 G' 中存在一条哈密尔顿环路，由于对每组顶点 v^1, v^2, v^3，v^2 只与 v^1 和 v^3 相连，因此在环路上必按 v^1, v^2, v^3 或 v^3, v^2, v^1 的顺序访问这三个顶点，而 G' 中的其他边仅连接上标为 1 和 3 的顶点，因此该环路上所有的三顶点组要么都按 $1, 2, 3$ 的顺序排列，要么都按 $3, 2, 1$ 的顺序排列，不妨设该环路为 $v_{i_1}^1, v_{i_1}^2, v_{i_1}^3, \cdots, v_{i_n}^1, v_{i_n}^2, v_{i_n}^3$，则 $v_{i_1}, \cdots, v_{i_n}$ 是 G 中的一条有向哈密尔顿环路。所以 G 包含有向哈密尔顿环路当且仅当 G' 包含无向哈密尔顿环路。 □

转换图示

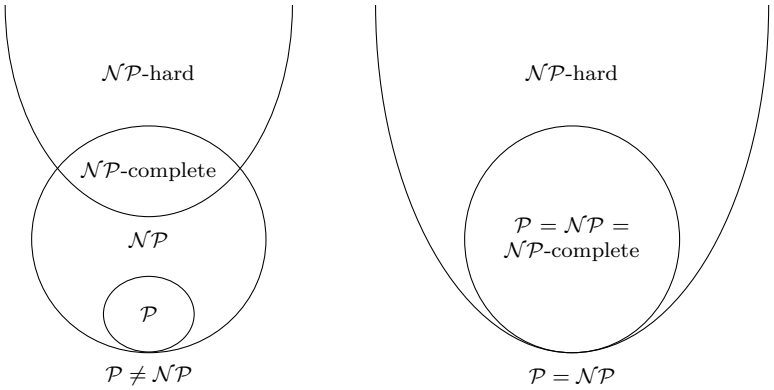


小结

- 在本讲之前，我们所讲的所有算法都以多项式为界，其所解决的问题都属于 $\mathcal{P}$ 问题；
- 本讲所给出的大部分算法问题都属于 $\mathcal{NP}$-完全问题；
- $\mathcal{P}$ 问题和非 $\mathcal{P}$ 问题的界限在于是否存在以多项式为界的 (确定) 算法；
- $\mathcal{NP}$ 问题和非 $\mathcal{NP}$ 问题的界限在于是否存在以多项式为界的非确定算法；
- $\mathcal{P} \subseteq \mathcal{NP}$，但 $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$ 仍然是一个开放的问题；
- $\mathcal{NP}$-完全问题是 $\mathcal{NP}$ 问题中最难的一类问题，其难度体现在所有的 $\mathcal{NP}$ 问题都可以 (多项式) 规约到任一个 $\mathcal{NP}$-完全问题，即只要任意一个 $\mathcal{NP}$-完全问题找到了以多项式为界的确定算法，则所有的 $\mathcal{NP}$ 问题都可以找到以多项式为界的确定算法，从而都属于 $\mathcal{P}$ 问题；

小结 **(cont.)**

- 遗憾的是到目前为止，仍然没有发现一个 $\mathcal{NP}$-完全问题存在以多项式为界的确定算法，同时也无法证明这些问题的复杂度下界高于多项式，而 $\mathcal{NP}$-完全问题却在被大量的发现，鉴于 $\mathcal{NP}$-完全问题的解决难度，多数研究者更倾向于认为 $\mathcal{P} \subset \mathcal{NP}$



(参考：http://en.wikipedia.org/wiki/NP-complete)

更深入的学习

- 一本有关 $\mathcal{NP}$-完全性理论的经典书籍：

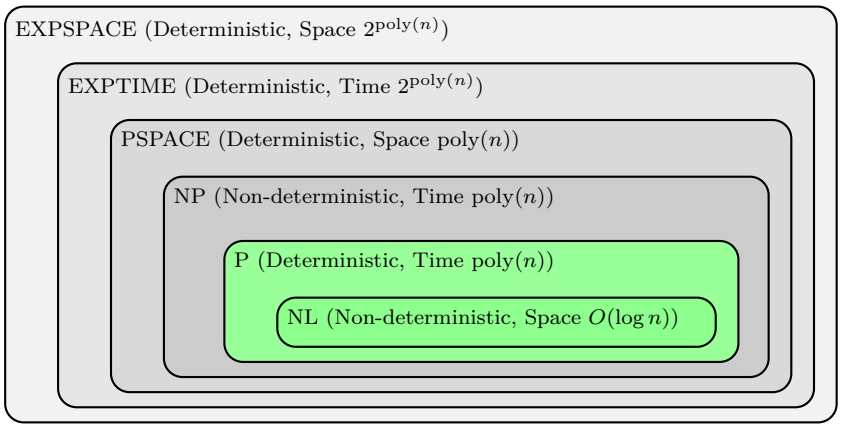
  Michael R. Garey, D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. New York: W.H. Freeman, 1979

- 一份已知的 $\mathcal{NP}$-完全问题清单：

  http://en.wikipedia.org/wiki/List_of_NP-complete_problems

- 一个研究领域：计算复杂性理论 (Computational Complexity Theory)

并非结束的结束



(参考：http://en.wikipedia.org/wiki/PSPACE)

- 到现在为止，我们能用计算机来解决的问题还是非常有限的，在算法领域还有非常广阔的未知空间等待我们去研究和探索！