第九章 NP完全理论

✓目录

- 易解问题和难解问题
- 理解P类和NP类问题的概念
- 理解NP完全问题的概念
- NP完全问题的近似算法
- 通过实例理解NP完全问题的近似算法

学习NP完全理论的意义

- 今天，NP-complete一词已经成为算法设计者在求解规模大而又复杂困难的问题时所面临的某种难以逾越的深渊的象征。

- 2000年初，美国克雷数学研究所的科学顾问委员会选定了七个“千年大奖问题”，该研究所的董事会决定建立七百万美元的大奖基金，每个“千年大奖问题”的解决都可获得百万美元的奖励。克雷数学研究所“千年大奖问题”的选定，其目的不是为了形成新世纪数学发展的新方向，而是集中在数学家们梦寐以求而期待解决的重大难题上。NP完全问题排在百万美元大奖的首位，足见它的显赫地位和无穷魅力。

- 现在，人们已经认识到，在科学和很多工程技术领域里，常常遇到的许多有重要意义而又没有得到很好解决的难题是NP完全问题。另外，由于人们的新发现，这类问题的数目不断增加。

易解问题和难解问题

- 人们将存在多项式时间算法的问题称为易解问题，将需要在指数时间内解决的问题称为难解问题。
- 目前，已经得到证明的难解问题只有两类：不可判定问题和非决定的难处理问题
- 值得注意的是，通常人们在实际中遇到的那些难解的且有重要实用意义的许多问题都是可判定的（即可求解），且都能用非决定的计算机在多项式时间内求解。不过，人们还不知道，是否能用决定的计算机在多项式时间内求解这些问题，这类问题正是NP完全性理论要研究的主要对象。

易解问题和难解问题

- 不可判定问题（不存在求解算法的问题）
  - 1. 图灵停机问题
    - 编译器用来判定程序是否会进入死循环
    - 反证法

```
Bool Halt( progCode ) { …… }

Void Test( progCode ) {
        while( Halt( progCode ) ) { ; }
}

Halt ( Test );   ?
```

易解问题和难解问题

- 不可判定问题（不存在求解算法的问题）
  - 2.波斯特对应问题（Post correspondence problem）
  - 3.某些形式语言的word problem
  - 4.决定王氏砖块是否能铺满平面
  - 5.判断标记系统是否停机
  - 6.计算某个字符串的柯氏复杂性
  - 7.希尔伯特第十问题：决定不定方程（又称为丢番图方程）的可解答性。
  - 8.决定上下文有关语言会否生成对应字母表的所有字符串；或判断其是否有歧义。
  - 9.两个上下文有关语言能否生成同样的字符串,或一个语言生成另一个语言生成的子集,或是否有某个字符串两个语言都生成。
  - 10.给予一个为初始点的有理坐标是周期性,决定位于basin of attraction是否开集,或是否在平分线函数迭代为两个纲比或三个纲比。
  - 11.决定Λ演算方程是否有正则形式。
  - ……

病毒检测也是不可判定的。

易解问题和难解问题

- 非决定计算机
  - 一种假想计算机，具有同时处理无数个并行的、相互独立的计算序列的能力。
  - 现实世界中的都是决定计算机
  - 未来的量子计算机可能会成为非决定计算机

P类问题

- 定义1：设A是问题 $\prod$ 的一个算法，且每一步只有一个确定的选择，即一个输入只对应一个输出，则称A为确定性算法。

- ★定义2：如果对于某个判定问题 $\prod'$ 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，得到是或否的答案，则该判定问题 $\prod'$ 是一个P类问题（polynomial）。

  - 事实上，所有易解问题都属于P类问题，例如最短路径判定问题就属于P类问题。如何将最短路径转换为判定问题？

  - 判断最短路径是否小于s，s的取值从0到所有路径和，s按最短路径间隔递增。

NP类问题

- 定义3：设A是问题 Π 的一个算法，如果以如下两阶段工作，则称为不确定算法。
  - 非确定的猜测阶段：
    - 用任意方法猜测一个可能解，这个解可以用多项式时间输出。
  - 确定的验证阶段：
    - 用一个确定算法判定这个解是否正确。

- ★定义4：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$O(n^k)$的时间运行一个不确定算法，得到是或否的答案，则该判定问题是一个NP类问题。
  - 从定义4可以看出，NP类问题是一类能够用不确定算法在多项式时间内求解的判定问题。对于NP类判定问题，重要的是它必须存在一个确定算法，能够以多项式时间来验证在猜测阶段所产生的答案。

P类问题和NP类问题的关系

- **P类—易解问题**
  - 可以用多项式时间的确定性算法来进行判定或求解。
- **NP类—易判定问题**
  - 可以用多项式时间的不确定性算法来进行判定或求解，关键是存在一个确定算法，能够以多项式的时间来验证在猜测阶段所产生的答案。
- **观点**
  - $P \subseteq NP$
  - NP规模远大于P，所以$NP \not\subset P$，$P \neq NP$
  - 但是没人能证明NP中的哪个问题不属于P，也没人能够为NP中的任何问题找到一个多项式的解法。 **P=NP ？**

P类问题和NP类问题的关系

- **NP问题是难解问题的一个子集，也就是说不是所有的难解问题都是NP问题。**
  - 纯排列、组合问题
  - 汉诺塔问题
  - ……

问题变换与计算复杂性归约

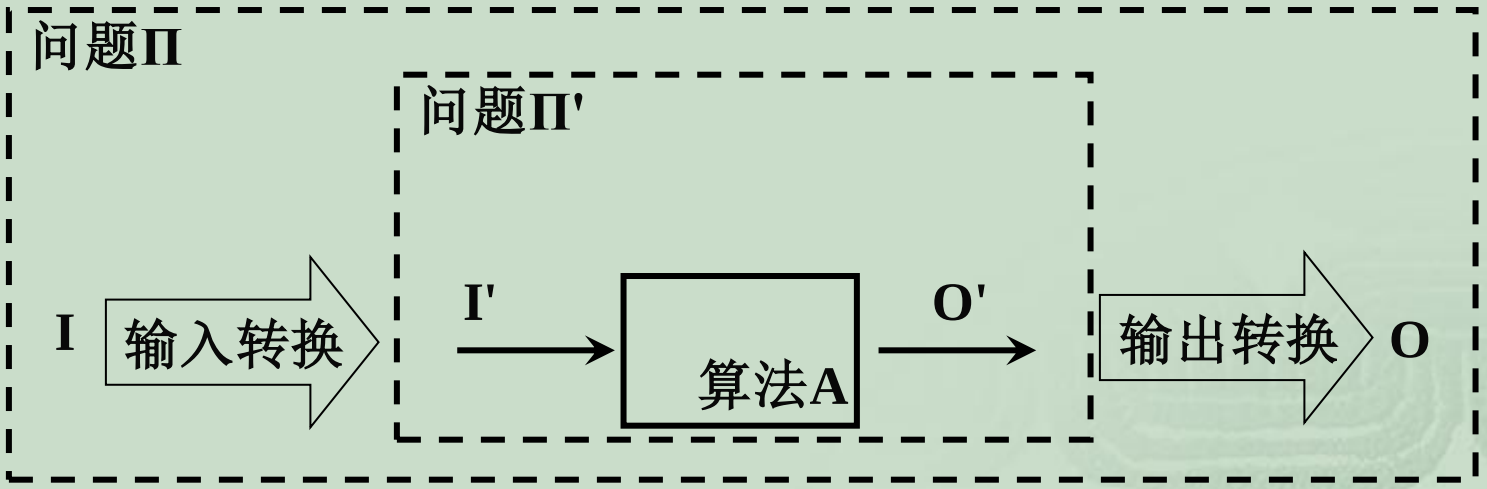
- 假设问题Π'存在一个算法A，对于问题Π'的输入实例I'，算法A求解问题Π'得到一个输出O'，另外一个问题Π的输入实例是I，对应于输入I，问题Π有一个输出O，则问题Π变换到问题Π'是一个三步的过程：

  1. 输入转换：把问题Π的输入I转换为问题Π'的适当输入I'；

  2. 问题求解：对问题Π'应用算法A产生一个输出O '；

  3. 输出转换：把问题Π '的输出O'转换为问题Π 对应于输入I的正确输出。

问题变换的一般过程



❖问题变换的主要目的不是给出解决一个问题的算法，而是给出通过另一个问题理解一个问题的计算时间上下限的一种方式。

例：配对问题到排序问题的变换

排序问题——输入I'是一组整数$X=(x_1, x_2, \ldots, x_n)$，输出O'是这组整数的一个排列$x_{i1} \leqslant x_{i2} \leqslant \ldots \leqslant x_{in}$。

配对问题——输入I是两组整数$X=(x_1, x_2, \ldots, x_n)$和$Y=(y_1, y_2, \ldots, y_n)$，输出O是两组整数的元素配对，即$X$中的最小值与$Y$中的最小值配对，$X$中的次小值与$Y$中的次小值配对，依此类推。

配对问题到排序问题的变换过程

假设存在算法A解决排序问题，则配对问题可以变换到排序问题：

1．把配对问题的输入I转化为排序问题的两个输入I_1'和I_2'；

2．排序这两组整数，即应用算法A对两个输入I_1'和I_2'分别排序得到两个有序序列O_1'和O_2'；

3．把排序问题的输出O1'和O2'转化为配对问题的输出O，这可以通过配对每组整数的第一个元素、第二个元素、……来得到。

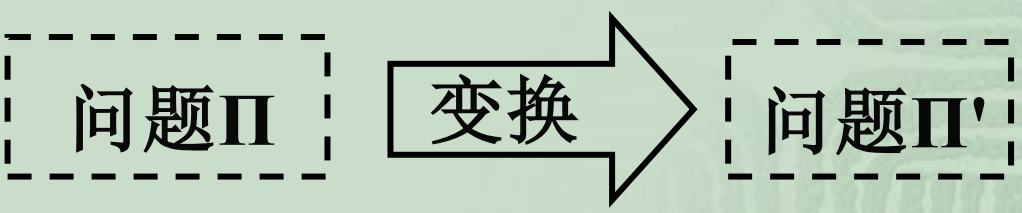
❖ 配对问题到排序问题的变换有助于建立配对问题代价的一个上限，因为上述输入和输出转换都是在多项式时间完成，所以，如果排序问题有多项式时间算法，则配对问题也一定有多项式时间算法。

计算时间下限归约：

若已知问题Π 的计算时间下限是$T(n)$，且问题Π 可$\tau(n)$变换到问题Π '，即$\Pi \propto_{\tau(n)} \Pi'$，则$T(n)-O(\tau(n))$为问题Π '的一个计算时间下限。

计算时间上限归约：

若已知问题Π '的计算时间上限是$T(n)$，且问题Π 可$\tau(n)$变换到问题Π '，即$\Pi \propto_{\tau(n)} \Pi'$，则$T(n)+O(\tau(n))$为问题Π 的一个计算时间上限。

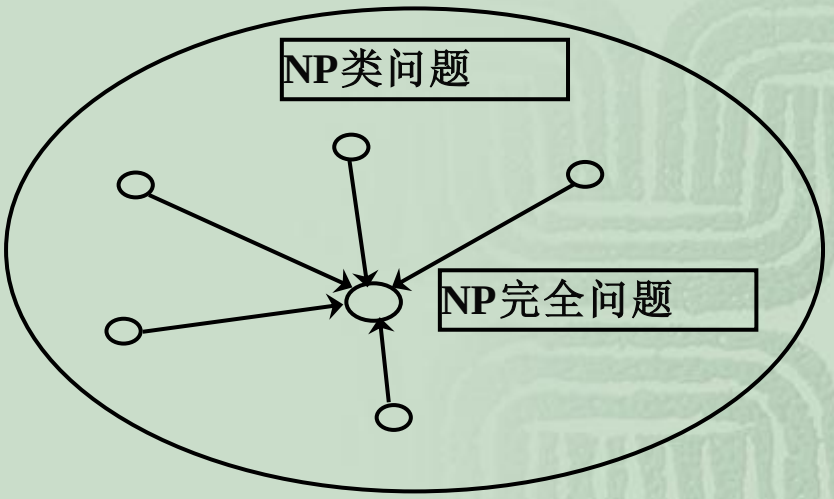


计算时间下限归约：　$T(n)$ 　- - →　$\tau(n)$ 　- - →　$T(n)-O(\tau(n))$

计算时间上限归约：$T(n)+O(\tau(n))$ 　← - -　$\tau(n)$ 　← - -　$T(n)$

NP完全问题（NPC）

- **NP完全问题是NP类问题的一个子类**
- **NP完全问题的特性**
  - 如果一个NPC能在多项式时间内得到解决，那么NP类中的每个问题都可以在多项式时间内得到解决，即P=NP成立！
  - 因为，任何一个NP问题均可以在多项式时间内变换成NPC。



多项式变换技术

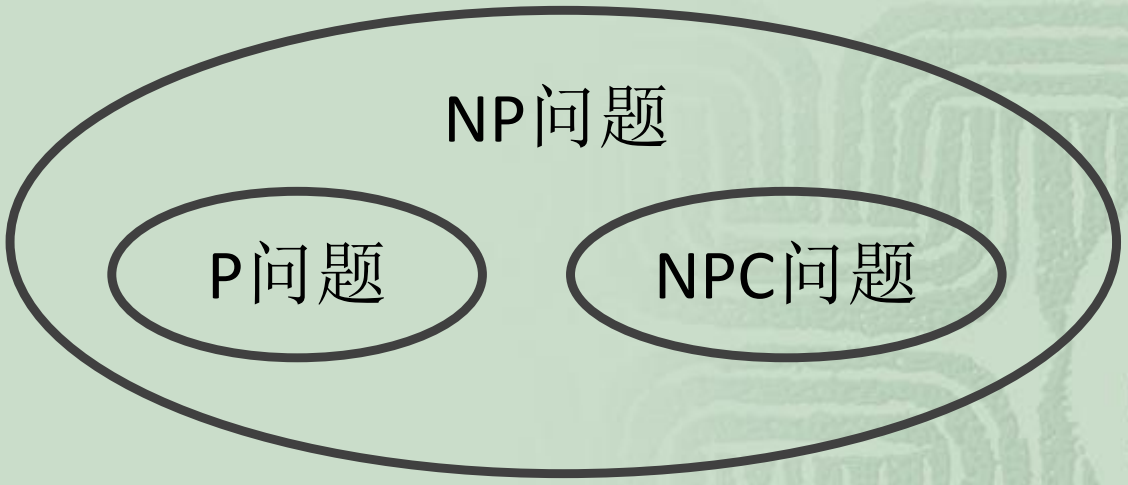
- 如果问题Q_1的任何一个实例，都能在多项式时间内转化成问题Q_2的相应实例，从而使得问题Q_1的解可以在多项式时间内利用问题Q_2相应实例的解求出，称问题Q_1是可多项式变换为问题Q_2的，记为$Q_1 \propto_p Q_2$，其中p表示在多项式时间内完成输入和输出的转换。
- 多项式变换关系是可传递的。

- ★定义5：令 Π 是一个判定问题，如果：
  - (1) $\Pi \in$ NP，即问题属于NP类问题
  - (2) 对NP中的所有问题 Π' 都有 $\Pi' \propto_p \Pi$ 则称该判定问题是一个NP完全问题，简记为NPC。

P＝NP？

广义上说，**P**类问题是可以用确定性算法在多项式时间内求解的一类问题，**NP**类问题是可以用非确定性算法在多项式时间猜测并验证的一类问题，而且P⊆NP。

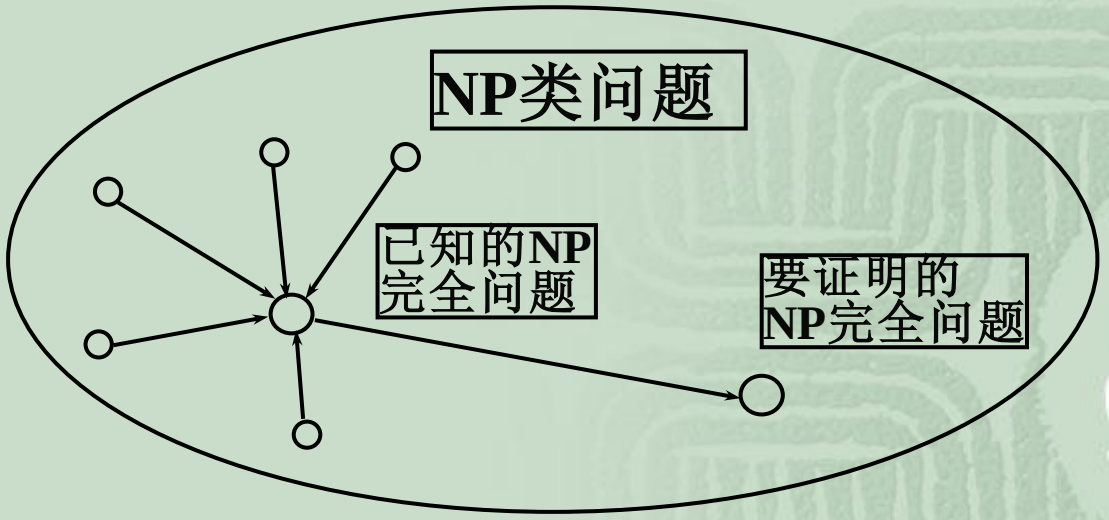
目前人们猜测**P≠NP**，则**P**类问题、**NP**类问题、**NP**完全问题之间的关系如下：



证明一个判定问题Π 是*NP*完全问题

Setp1: 证明问题Π属于NP类问题，也就是说，可以在多项式时间以确定性算法验证一个任意生成的串，以确定它是不是问题的一个解；

Step2: 证明NP类问题中的每一个问题都能在多项式时间变换为问题Π 。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变换为问题Π 。



典型的NP完全问题

- **Cook 定理：布尔表达式的可满足性问题（SAT）**
  - Cook在他1971年的论文中指出：所谓的合取范式可满足性问题就是NP完全问题 。
  - 这是第一个获得“NP完全问题”称号的问题。

- **SAT问题**
  - 一个由布尔变量（只取0或1 ）构成的逻辑表达式，运算只有“与”“或”“非”，如：$x \vee y \wedge \neg z$。判断这个表达式能等于1吗？如果能，x，y，z分别等于什 么。表达式等于1，说明这个表达式被满足了。
  - 证明：略（图灵机证明）

典型的NP完全问题

- 几个典型的NP完全问题。

（1）图着色问题COLORING

（2）路径问题LONG-PATH

（3）顶点覆盖问题VERTEX-COVER

（4）子集和问题SUBSET-SUM

（5）哈密尔顿回路问题HAM-CYCLE

（6）旅行商问题TSP

（7）装箱问题BIN-PACKING

（8）Windows自带游戏：扫雷、当空接龙、蜘蛛牌……

（9）……

NP完全问题的计算机处理

1. 采用先进的算法设计技术
2. 充分利用限制条件
3. 近似算法
4. 概率算法
5. 并行计算
6. 智能算法

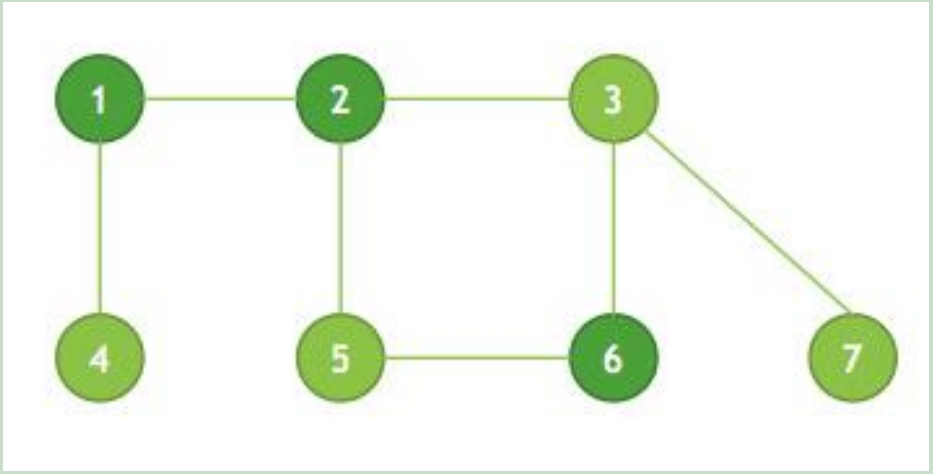
NP完全问题的近似解法

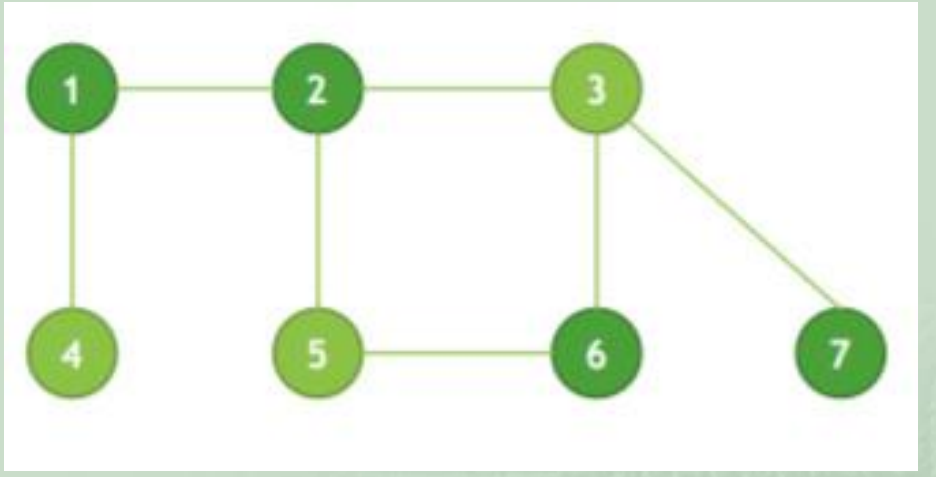
- 一般来说，近似算法所适用的问题是最优化问题，即要求在满足约束条件的前提下，使某个目标函数值达到最大或者最小。对于一个规模为n的问题，近似算法应该满足下面两个基本的要求：
- (1) 时间复杂性：要求算法能在n的多项式时间内完成。
- (2) 解的近似度：算法的近似解应满足一定的精度。
  - 通常，用来衡量精度的标准有近似比和相对误差。

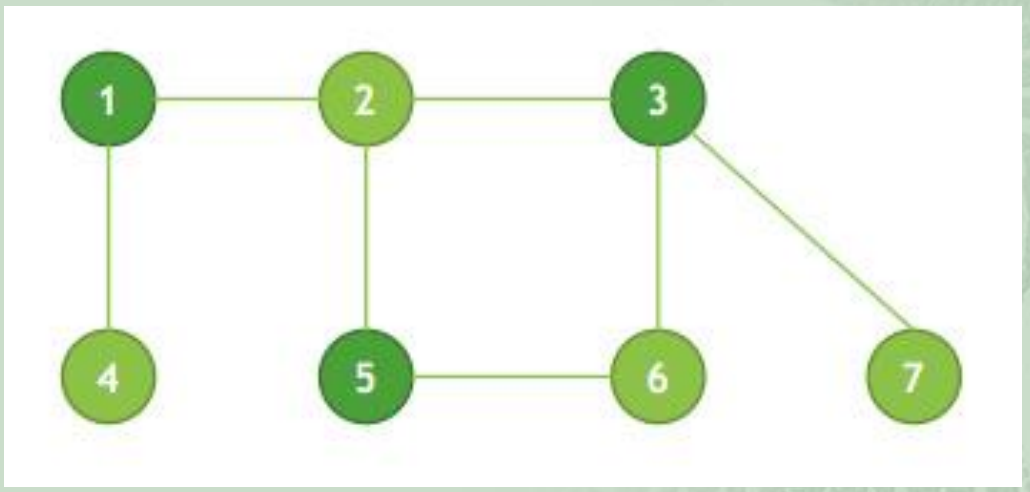
NP完全问题的近似解法

- 近似比：
  - 设最优解为C，一个近似解为c
    - 则最大化问题的近似比：ρ(n)=C/c
    - 最小化问题的近似比：ρ(n)=c/C
- 相对误差：
  - $\lambda = \left|\frac{c-C}{C}\right|$

顶点覆盖问题





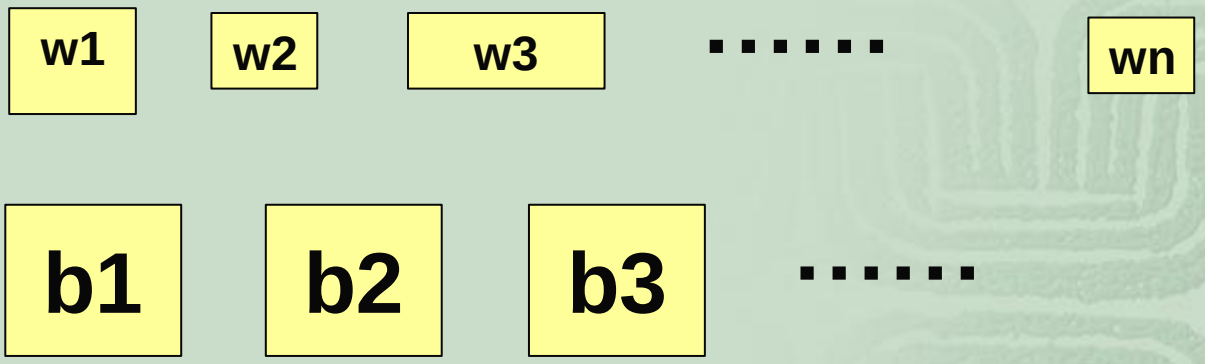


顶点覆盖问题

- 近似算法：
  - 无向图G=(V,E)。集合Cset来存储目标顶点，初值为空。
  - 边集E1=E，然后不断从E1中选取一条边(u,v)，端点u和v加入Cset中，并将E1中与顶点u和v相邻接的所有边删去，直至Cset已覆盖E1中所有边，即E1为空时算法停止。
  - 最后Cset是G的一个顶点覆盖，由于每次把尽量多的相邻边从边集E1中删除，可以期望Cset中的顶点数尽量少，但不能保证Cset中的顶点数最少。
- 顶点覆盖问题的近似算法的性能比为2。

装箱问题

- 设有n个物品w1，w2，…，wn和若干个体积均为C的箱子b1，b2，…,bk,…。n个物品的体积分别为s1，s2，…,sn且有si≤C(1≤i≤n)。要求把所有物品分别装入箱子且物品不能分割，使得占用箱子数最少的装箱方案。



装箱问题

- 近似算法
  - 首次适宜法（近似比小于2）
  - 最适宜法（近似比小于2）找一个使得剩余空间最小的箱子放进去
  - 首次适宜降序法
  - 最适应降序法

旅行商问题TSP

- 问题描述：给定一个无向带权图G=(V,E)，对每一个边(u,v)∈E，都有一个非负的常数费用c(u,v)>0，求G中费用最小的哈密尔顿回路。
- 两类TSP问题
  - 欧几里德TSP：如果图G中的顶点在一个平面上，任意两个顶点之间的距离为欧几里德距离。那么，对于图中的任意3个顶点u、v、w∈V，其费用函数具有三角不等式性质：c(u,v)≤c(u,w)+c(w,v)。
  - 一般TSP：把不具有欧几里德性质的TSP问题。

欧几里德TSP问题

- 近似算法：
  - 在图中任选一个顶点u，用Prim算法构造图G的以u为根的最小耗费生成树T，然后用深度优先搜索算法遍历最小耗费生成树T，取得按前序遍历顺序存放的顶点序列L，则L中顺序存放的顶点号即为欧几里德旅行商问题的解。
- 算法的近似比小于2.

集合覆盖问题

- 集合覆盖问题是对常见的组合问题的抽象。
  - 例如，假设有X种颜色的球，有一群人F，每人手中有不同颜色的球若干。希望从F中选出尽可能少的人构成一个组，使得X种颜色的球在这个组里都能看到。

算法设计

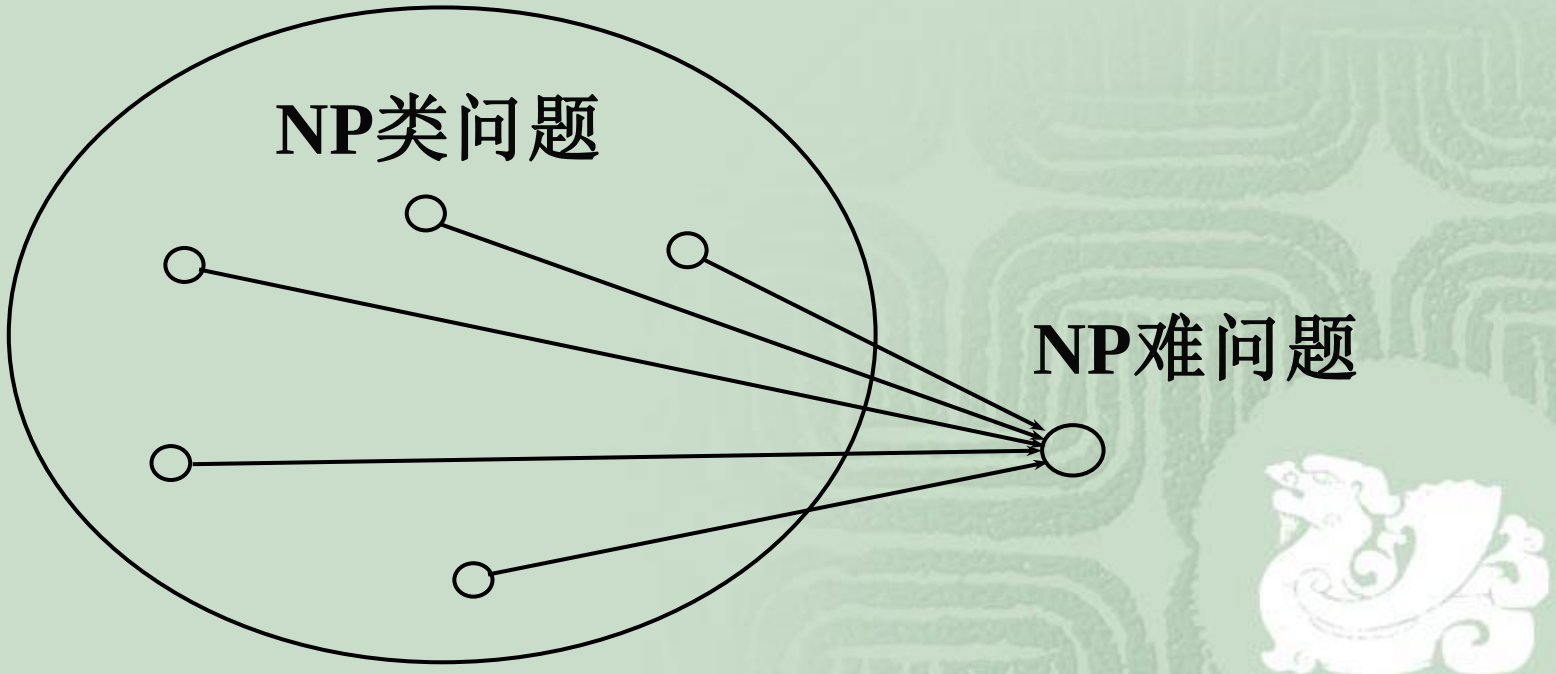
- 令集合U存放每一阶段中尚未被包含的X中的颜色，集合C包含了当前已选择 的人。
- 算法的求解步骤如下：
  - (1) 初始化，令U=X，C为空集；
  - (2) 先从F中选择一个人s，他手中尽可能多的拥有未被包含的颜色；
  - (3) 将U中被s拥有的颜色删去，并将s加入C中；
  - (4) 如果集合U不为空，重复步骤 (2) 和 (3)。否则，算法结束，此时C中包含了覆盖X的F的一个子集族。
- 该算法的近似比为ln|X|+1。

其它问题

- NP-Hard问题
- Co-NP问题
- Co-NP完全问题
- P-SPACE问题
- P-SPACE完全问题

NP-Hard问题

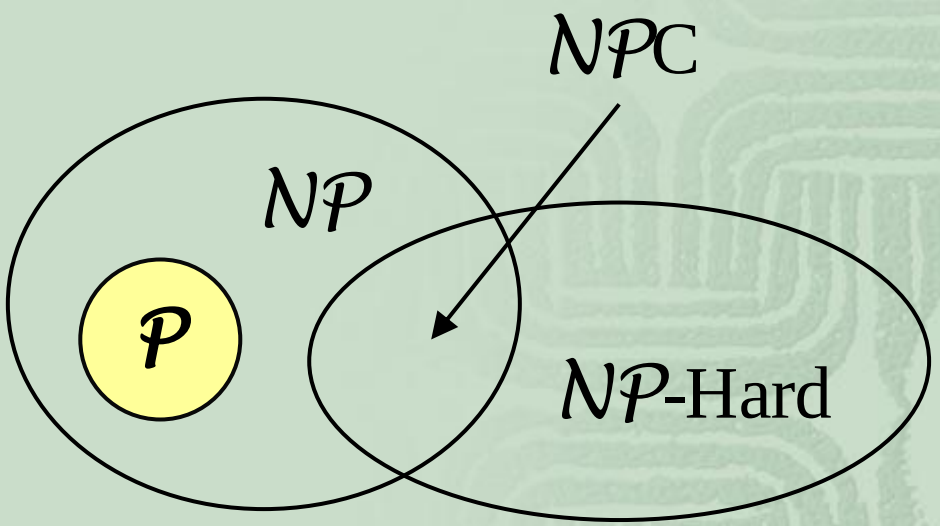
令Π 是一个判定问题，如果对于*NP*类问题中的每一个问题Π '，都有$\Pi' \propto_p \Pi$，则称判定问题Π 是一个**NP难**问题。



NP完全问题和NP难问题的区别：

如果Π 是NP完全问题，Π ’是NP难问题，那么，他们之间的差别在于Π 必定是NP类问题，而Π ’不一定在NP类问题中。

❖ 一般而言，若判定问题属于NP完全问题，则相应的最优化问题属于NP难问题。



Co-NP与Co-P

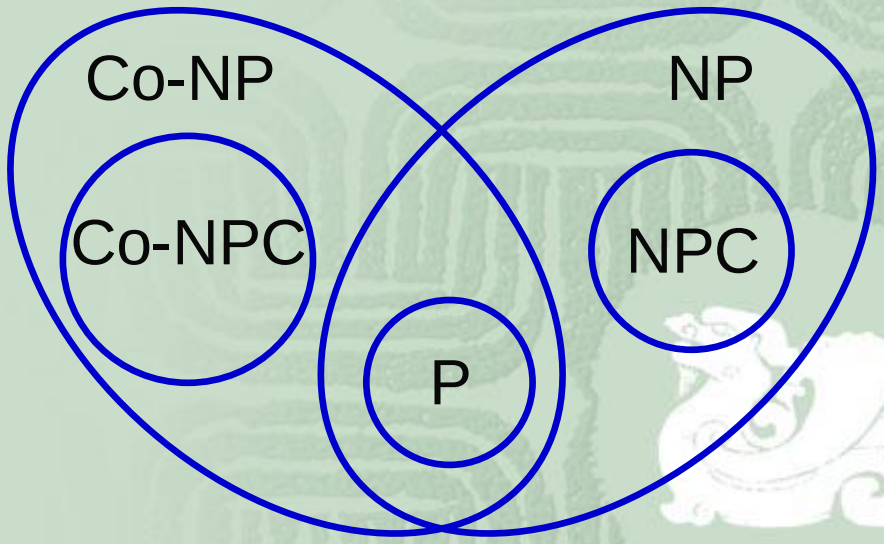
- Co-P ＝ $\{\overline{\Pi} \mid \Pi \in NP\}$;
- Co-NP＝$\{\overline{\Pi} \mid \Pi \in P\}$。

人们猜想： **NP≠Co-NP**

但对于**Co-P**：**P＝Co-P**

三个未解决的问题

- 需要指出，到目前为止，对下面三个问题：
  1. **P＝NP吗？**
  2. **P＝NP ∩ Co-NP吗？**
  3. **Co-NP＝NP吗？**



空间复杂性简介

- 到目前为止，我们只考虑了算法执行或问题求解过程中所需要的一种资源——时间，而空间做为另一种资源也常常很重要。

- P-SPACE定义： 如果问题Π存在一些算法，使得它的运算能被限制在以输入长度大小的多项式为界的空间内进行，则称Π为P空间的问题。

- P-SPACE完全定义： 如果某问题是P-SPACE的，并且所有其它P-SPACE的问题都可多项式变换到它，则称它为P-SPACE完全的。

各类问题间可能的相互关系示意图