第八章 NP与计算的难解性

李 翔

内容

- 多项式时间归约
- 使用“零件”归约：SAT
- 有效证书和NP的定义
- NP完全问题
- 排序、划分、图着色、数值问题
- 难问题的部分分类

NP与计算的难解性

- 贪心算法　　区间调度：O(n log n).
- 分治策略　　FFT　　O(n log n).
- 动态规划　　编辑距离　　$O(n^2)$

最开始我们把**多项式时间**作为效率的工作概念

面对一些困难的问题，我们即不知道这些问题存在多项式时间算法，也不能证明问题不存在多项式时间算法

这里将会对“困难”问题提出一个清晰的概念：在计算上实际上是难的，虽然我们不能证明它---NP完全

8.1 多项式时间归约

- 哪些问题在实际中存在可行解?
- 可行性定义： [Cobham 1964, Edmonds 1965, Rabin 1966] 存在多项式时间的算法.

Yes	Probably no
Shortest path	Longest path
Matching	3D-matching
Min cut	Max cut
2-SAT	3-SAT
Planar 4-color	Planar 3-color
Bipartite vertex cover	Vertex cover
Primality testing	Factoring

多项式时间归约

对问题按照相对难度做一个分类

- 比较
- 如何形式的表达“问题X至少像问题Y一样的难”？
- 在计算模型中假设X可以在多项式时间内求解。假设有一个能解问题X实例的**黑盒子**，如果写下X实例的输入，那么可以一步返回正确答案。

多项式时间归约

- 如果对于问题Y:
  - 能够用多项式个标准的计算步骤;
  - 加上多项式次调用解问题X的黑盒子来解问题Y

那么记作$Y \le_p X$;

读作 “Y多项式时间可归约到X”;或 “X至少像Y一样的难(相对于多项式时间)”

多项式时间归约

- 定理8.1 假设$Y \le_p X$，如果X能在多项式时间内求解，则Y也能在多项式时间内求解。

- 定理8.2 假设$Y \le_p X$，如果Y不能在多项式时间内解决，则X不能在多项式时间内解决。

多项式时间归约

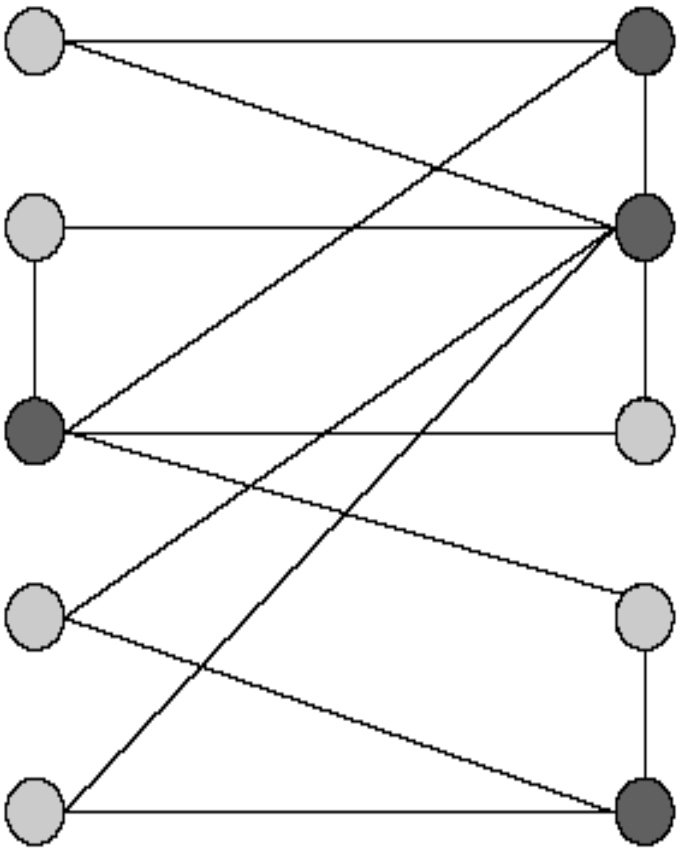
- 归约的基本策略
  - 简单等价归约.
  - 从特殊情形归约.
  - 使用零件归约.

多项式时间归约

独立集

- 在图G=(V,E)中，如果顶点集合S $\subseteq$ V 中的任意两点之间没有边，则称S是**独立**的。
- **独立集问题：**
- 给定图G和数k,问G包含大小至少为k的独立集吗？

多项式时间归约



存在大小至少为6的独立集吗？

Yes

存在大小至少为7的独立集吗？

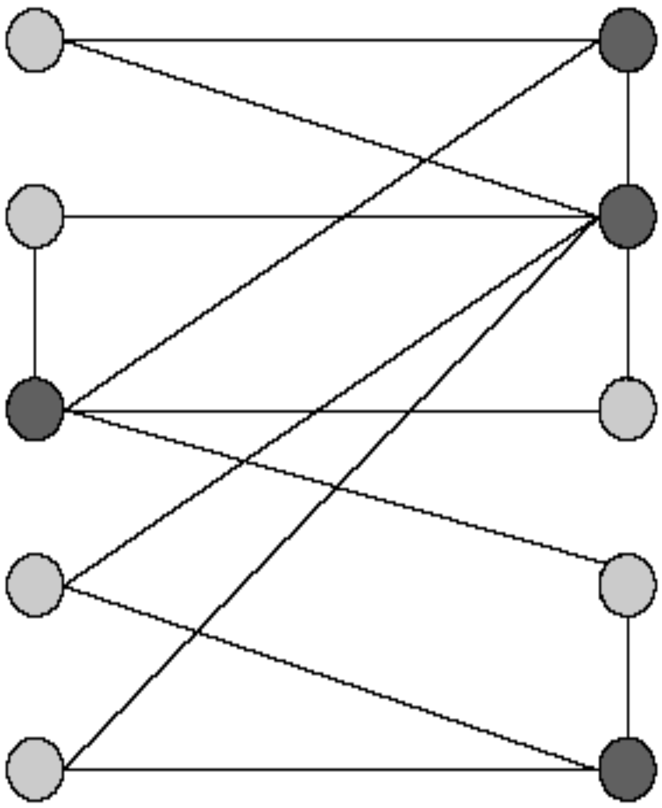
No

多项式时间归约

现在考虑另外一个基本图论问题：顶点覆盖

- 给定图G=(V,E),如果每一条边e ∈ E至少有一个端点在S中，则称S是一个**顶点覆盖**。
- 顶点覆盖问题：
- 给定图G和数k,问G是否包含大小至多为k的顶点覆盖？

多项式时间归约



存在大小至多为4的顶点覆盖吗？

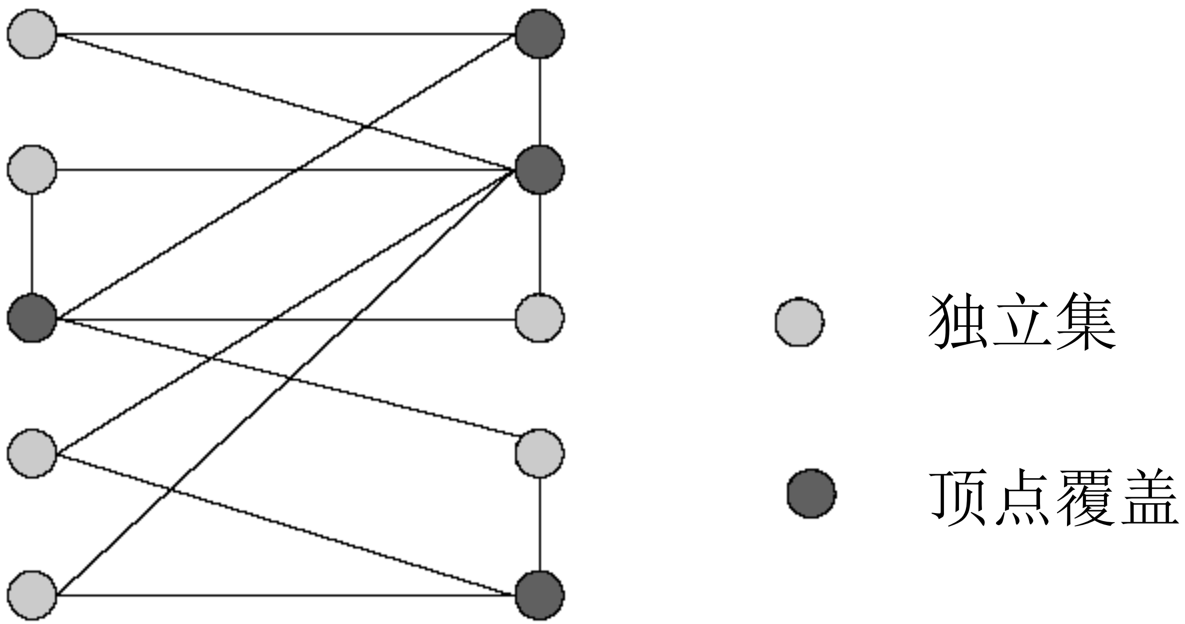
Yes

存在大小至多为3的顶点覆盖吗？

No

多项式时间归约

- 看起来顶点覆盖问题，独立集问题可能存在一定联系



多项式时间归约

- 引理8.3 设G=(V,E)是一个图， $S \subseteq V$，那么S是一个独立集当且仅当它的补V-S是一个顶点覆盖。
- 证明： $\Rightarrow$

首先，设S是一个独立集。考虑任一条边e=(u,v). 因为S是独立集，u和v必有一个在V-S中，于是每一条边至少有一个端点在V-S中。所以V-S是一个顶点覆盖。

多项式时间归约

- $\Leftarrow$
- 设V-S是一个顶点覆盖。考虑S中的任意两个顶点u和v, 如果他们之间有一条边e, 那么e的两个端点都不在V-S中，与假设V-S是一个顶点覆盖矛盾。因此，S中的任意两个顶点之间都没有边，S是一个独立集。

多项式时间归约

- 命题.
- VERTEX-COVER $\equiv_P$ INDEPENDENT-SET

- 下面将介绍从特殊情形归约到一般情形的方法

多项式时间归约

- 集合覆盖：试图用一组较小的集合覆盖一个任意的对象集合
- 应用实例
  - m 个可供利用的软件；
  - 集合U由整个系统必须具备的n个性能组成；
  - 第i个软件包含性能集合$S_i \subseteq U$.
  - 目标：用尽可能少数的软件包含所需具备的n个性能。

多项式时间归约

- 集合覆盖问题
- 给定n个元素的集合U, U的子集S_1, S_2, . . . , S_m 以及数k, 是否存在数目至多为k的子集合，其并集等于U

例子

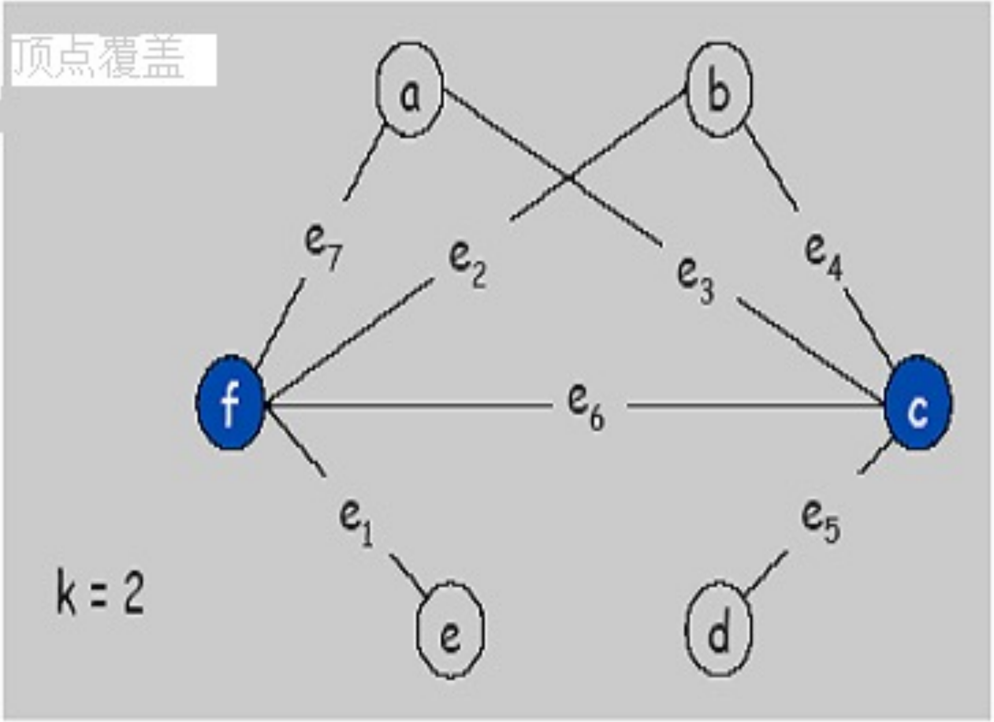
U = { 1, 2, 3, 4, 5, 6, 7 }
k = 2
S_1 = {3, 7}
S_2 = {3, 4, 5, 6}
S_3 = {1}
S_4 = {2, 4}
S_5 = {5}
S_6 = {1, 2, 6, 7}

多项式时间归约

直觉上告诉我们，集合覆盖比顶点覆盖更具一般性

- 定理8.6 顶点覆盖$\leq_P$集合覆盖
- 证明：*我们构造集合覆盖的一个实例,* U=E. 在顶点覆盖中，每次取一个顶点覆盖所有与它关联的边。于是，对每一个顶点$i \in V$, $S_i = \{e \in E : e$ 连接到$i\}$;
  k = k

多项式时间归约



集合覆盖

$U = \{1, 2, 3, 4, 5, 6, 7\}$

$k = 2$

$S_a = \{3, 7\}$　　$S_b = \{2, 4\}$

$S_c = \{3, 4, 5, 6\}$　　$S_d = \{5\}$

$S_e = \{1\}$　　$S_f = \{1, 2, 6, 7\}$

多项式时间归约

- 对于这个构造的实例，U能被至多k个子集合覆盖，当且仅当G有大小至多为k的顶点覆盖。
- 如果$S_{i_1},\ldots,S_{i_l}$是*l*<=*k*个覆盖U的集合，那么G中的每一条边都关联到顶点$i_1,\ldots,i_l$中的一个,$\{i_1,\ldots,i_l\}$是G中大小为*l*的顶点覆盖。反过来亦然。

多项式时间归约

- 注意到集合覆盖是顶点覆盖的自然推广
- 类似的，独立集的自然推广是关于集合的包装问题
- **集合包装**问题：希望把大量集合包装在一起，限制他们中的任意两个都不重叠。
- 例子：设想有n个不能共享的资源集合U,m个软件进程，第i个进程运行需要的资源集为S_i,需要从这些进程中寻找一组进程，使得他们能够同时运行，而且所需资源没有重叠**冲突**。

多项式时间归约

- 问题描述：
- 给定n个元素的集合U, U的子集 $S_1,\ldots S_m$ 以及数k, 问在这些子集中至少有k个两两不相交吗？

定理8.7 独立集$\leq_p$集合包装

使用“零件”归约

- 给定n个布尔变量$x_1,\ldots,x_n$ 的集合X,每个布尔变量可以取值0,1
- 项(Literal):一个布尔变量或者其逆 x_i or $\overline{x_i}$
- 子句(Clause): 项的析取 $C_j = x_1 \vee \overline{x_2} \vee x_3$
- 合取范式(Conjunctive normal form): 一个命题公式Φ 由子句的合取范式构成。

$$\Phi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$$

使用“零件”归约

- 如果赋值v满足每一个子句$C_1,\ldots,C_k$, 也就是合取式 $\phi = c_1 \wedge \cdots \wedge c_k$ 值为1，称v是关于$C_1,\ldots,C_k$的一个满足的赋值，又称这组子句是可**满足**的。

例子：$(\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$ 是否可满足？

Yes: x_1 = 1(true), x_2 = 1(true), x_3 = 0(false).

使用“零件”归约

- 可满足性问题：(SAT)
- 给定变量集合 $X=\{x_1,\ldots,x_n\}$ 上的一组子句 $C_1,\ldots,C_k$, 或者是对于 $\phi=c_1\wedge\cdots\wedge c_k$ (CNF), 问存在满足的真值赋值吗？

3-SAT:三元可满足性：

要求每一个子句的长为3

使用“零件”归约

- 可满足性和三元可满足性是基本的组合搜索问题，以非常梗概的方式包含一个难计算问题的基本要素，必须作出n个独立的决定满足一组约束。

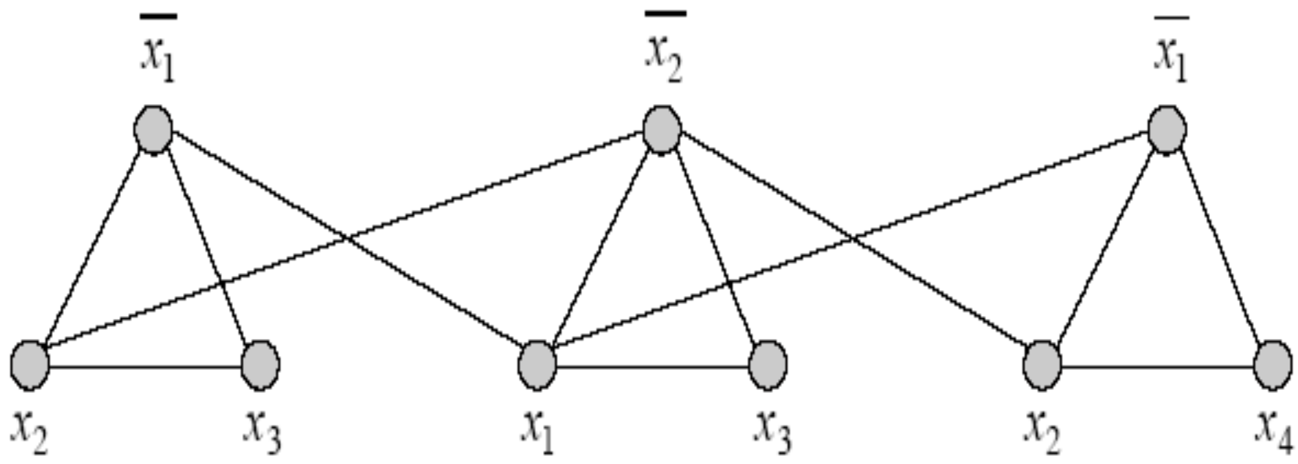
- 定理8.8 3-SAT $\leq_P$ 独立集.

使用“零件”归约

- 证明：给定3-SAT的一个实例Φ，我们构造独立集 (G, k)的一个实例，使得存在一个大小为k的独立集当且仅当Φ 是可满足的.
- 构造:
  - G 包含3个顶点对应到一个子句，每一个顶点代表一个项.
  - 连接一个三角形(对应到一个子句)的三个顶点(项).
  - 把每一个项与它的否定项连接起来.

使用“零件”归约

G



k = 3

$$\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$$

使用“零件”归约

- 规约的传递性

- 定理8.9 如果$X \le_p Y$ 且 $Y \le_p Z$, 则$X \le_p Z$.

  证明：把两个算法组合起来

使用“零件”归约

- 简单等价归约：独立集 $\equiv_P$ 顶点覆盖.
- 从特殊情形归约：顶点覆盖 $\le_P$ 集合覆盖.
- 使用“零件”归约：3-SAT $\le_P$ 独立集

于是对于这几个问题的难度我们有了一个比较关系：

3-SAT $\le_P$ 独立集 $\le_P$ 顶点覆盖 $\le_P$ 集合覆盖.

归约—问题描述

- 判别问题．是否存在一个集合大小不超过k的顶点覆盖？
- 寻找问题．寻找最小的顶点覆盖
- 自归约性质．寻找问题 $\le_P$ 判别问题
  - 可以应用到本章中的所有(NP完全)问题.
  - 我们可以重点集中讨论判别问题.

8.3 有效证书和NP的定义

- 一个计算问题的输入被编码成有穷的二进制串s,串s的长度记为|s|.把判定问题X等同于由对它的答案为“Yes”的串组成的集合
- 判定问题的算法A接受输入串s并返回值“yes”或“no”,返回的值记为A(s),如果对所有的串s,A(s)=yes *当且仅当*$s \in X$,则称A解问题X.

NP的定义

- 如果存在多项式p(.)使得对每一个输入串s, 算法A对s的计算在至多O(p|s|)步内终止，则称A有多项式运行时间。
- 根据这一概念，就形成了问题类：**存在多项式时间解法的问题的集合P**

NP的定义

- PRIMES: X = { 2, 3, 5, 7, 11, 13, 17, 23, 29, 31, 37, …. }
- 判定素性问题:
- [Agrawal-Kayal-Saxena, 2002] AKS算法
- Manindra Agrawal, Neeraj Kayal, and Nitin Saxena on August 6, 2002： *PRIMES is in P*.
- The authors received the 2006 Gödel Prize and the 2006 Fulkerson Prize for this work.
- $p(|s|) = |s|^{6+\varepsilon}$

NP的定义

- 一些经典的P类问题

Problem	Description	Algorithm	Yes	No
MULTIPLE	Is x a multiple of y?	Grade school division	51, 17	51, 16
RELPRIME	Are x and y relatively prime?	Euclid (300 BCE)	34, 39	34, 51
PRIMES	Is x prime?	AKS (2002)	53	51
EDIT-DISTANCE	Is the edit distance between x and y less than 5?	Dynamic programming	niether neither	acgggt tttta
LSOLVE	Is there a vector x that satisfies Ax = b?	Gauss-Edmonds elimination	$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 4 & -2 \\ 0 & 3 & 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 2 \\ 36 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

NP的定义

- 直觉上告诉我们，求解问题是困难的，但是**验证**一个问题的解相对要容易

- 问题X的“验证算法”具有不同于求解问题的算法结构，为了验证一个解，需要输入串s以及另外的“**证书**”串t,这个证书t包含s是X的“yes”实例的证据。

NP的定义

- 如果：
- B是有两个输入变量s,t的多项式时间算法
- 存在多项式p使得对每一个输入串s, s∈X当且仅当存在串t使得$|t| \le p(|s|)$ 且B(s,t)=yes.
- 则称B是问题X的有效验证程序。

t是证书

NP的定义

- 根据验证算法的性质，可以定义新的问题类别
- NP(Nondeterministic polynomial time)是所有存在有效验证程序的问题的集合

NP的定义

- 定理8.10 P$\subseteq$NP
- 证明：考虑问题X $\in$P,意味着存在一个解X的多项式时间算法A。

如下设计B,对于输入(s,t),验证程序直接返回值A(s).

可以看到，如果串s $\in$X,对于每一个长度不超过p(|s|)的t, 满足B(s,t)=yes…

可见B是有效验证程序。

NP的定义

- 例子：对于三元可满足性问题
- 证书t?
- 对所有变量的真值赋值
- 验证程序B?
- 计算ϕ在这个赋值下的值

NP的定义

$$\left(\overline{x_1} \vee x_2 \vee x_3\right) \wedge \left(x_1 \vee \overline{x_2} \vee x_3\right) \wedge \left(x_1 \vee x_2 \vee x_4\right) \wedge \left(\overline{x_1} \vee \overline{x_3} \vee \overline{x_4}\right)$$

实例 s

$$x_1 = 1,\ x_2 = 1,\ x_3 = 0,\ x_4 = 1$$

证书 t

SAT, 3SAT $\in$ NP

NP的定义

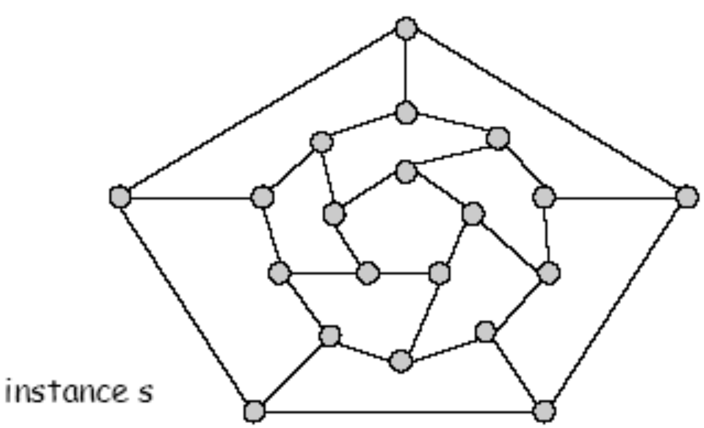
- 例子：Composite,给定一个整数 s, s 是合数吗?
- 证书:
- s的一个非平凡因子。注意到这样一个证书存在当且仅当s是一个合数，此外$|t| \le |s|$.
- 验证程序:

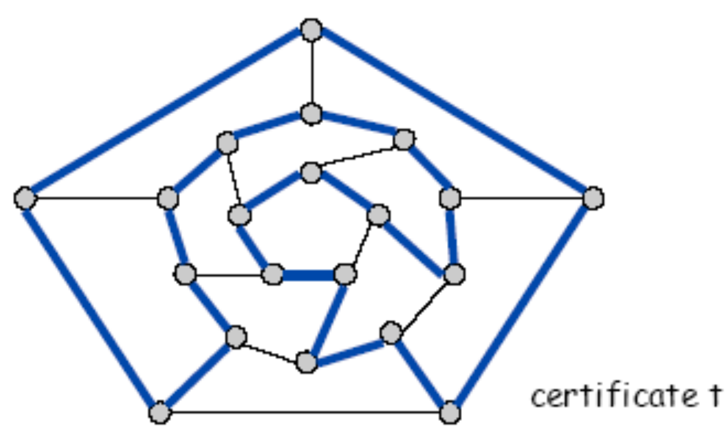
```
boolean C(s, t) {
   if (t ≤ 1 or t ≥ s)
      return false
   else if (s is a multiple of t)
      return true
   else
      return false
}
```

COMPOSITES $\in$ NP

NP的定义

- **哈密尔顿回路.** 给定无向图 G = (V, E), 是否存在一个简单回路C访问到每一个结点?
- 证书： n个结点的一个有序排列
- 验证程序. 检查排列中包含每一个结点；排列中相邻结点有边相连。
- HAM-CYCLE $\in$ NP.





NP的定义

- P = NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]
  - 判定问题和验证问题一样难吗?
  - $1 million prize.
- If yes: Efficient algorithms for 3-COLOR, TSP, FACTOR, SAT, …
- If no: No efficient algorithms possible for 3-COLOR, TSP, SAT, …
- P = NP? 大家的意见一致倾向于不相等.

8.4 NP完全问题

- P=NP这个问题没有太多进展，转向另外一个问题，那些是NP中**最难**的问题？
- 自然的，要求NP中的每一个问题都能够归约到X
- NP完全问题X

i. $X \in NP$

ii. 对于所有的$Y \in NP$, $Y \leq_p X$.

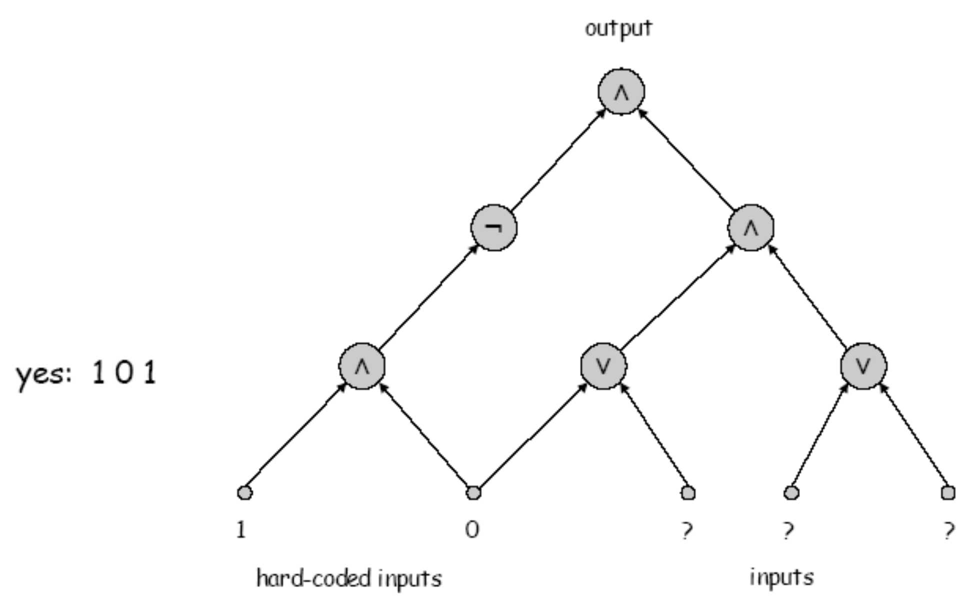
NP完全问题

- 定理8.12. 设X是一个NP完全问题，那么X是在多项式时间内可解的当且仅当P = NP.
- Pf. $\Leftarrow$ 如果 P = NP，那么自然的X在多项式时间内可解因为 X 属于 NP.

  Pf. $\Rightarrow$ 设X在多项式时间内可解.
  - 设Y 是 NP中的任意一个问题，因为 $Y \leq_p X$, 所以 Y 在多项式时间内可解， 这就推出 NP $\subseteq$ P.
  - 显然我们知道 P $\subseteq$ NP， 这样P = NP. ■

NP完全问题

- 是否存在一个天然的NP完全问题？
- 电路可满足性(CIRCUIT-SAT)．给定一个由AND, OR, NOT电路门组成的电路，需要确定是否存在对输入的赋值使得输出值为1?



NP完全问题(电路可满足性)

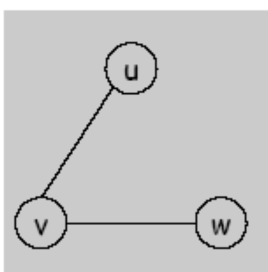
定理8.13([Cook 1971, Levin 1973]) 电路可满足性是**NP**完全的

- Pf. Any algorithm that takes a fixed number of bits n as input and produces a yes/no answer can be represented by such a circuit. Moreover, if algorithm takes poly-time, then circuit is of poly-size.
  - Consider some problem X in NP. It has a poly-time certifier C(s, t). To determine whether s is in X, need to know if there exists a certificate t of length p(|s|) such that C(s, t) = `yes`.
  - View C(s, t) as an algorithm on |s| + p(|s|) bits (input s, certificate t) and convert it into a poly-size circuit K.
    - first |s| bits are hard-coded with s
    - remaining p(|s|) bits represent bits of t
  - Circuit K is satisfiable iff C(s, t) = `yes`.

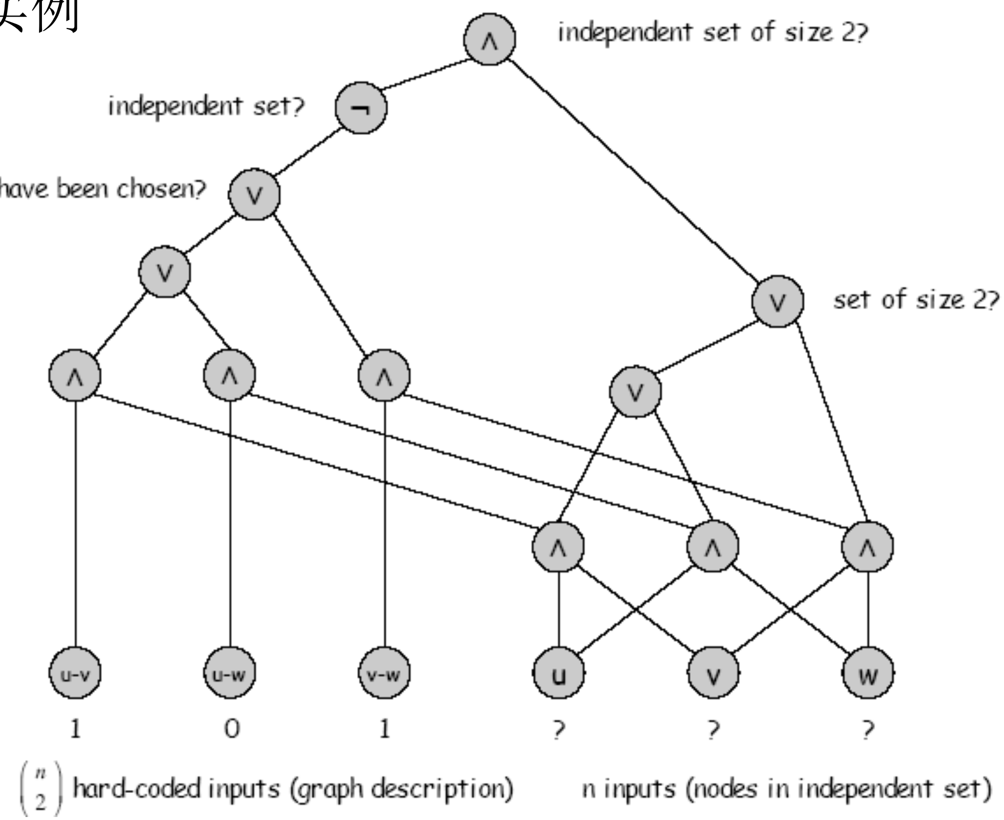
NP完全问题(电路可满足性)

例子：给定一个图G,它包含两个顶点的独立集吗？

构造一个等价的电路可满足实例



G = (V, E), n = 3



NP完全问题

- 如何发现更多的NP完全问题？
- 定理8.14 如果Y是一个NP完全问题，X属于NP且$Y \le_P X$,则X是NP完全的。
- 证明：设Z是NP中的任意一个问题，由Y的NP完全性，有$Z \le_P Y$. 于是，根据传递性，$Z \le_P X$, 根据定义知道X是NP完全的。

NP-Complete Problems

3-SAT$\leq_p$独立集$\leq_p$顶点覆盖$\leq_p$集合覆盖

因此他们都是NP完全问题

NP-Complete Problems

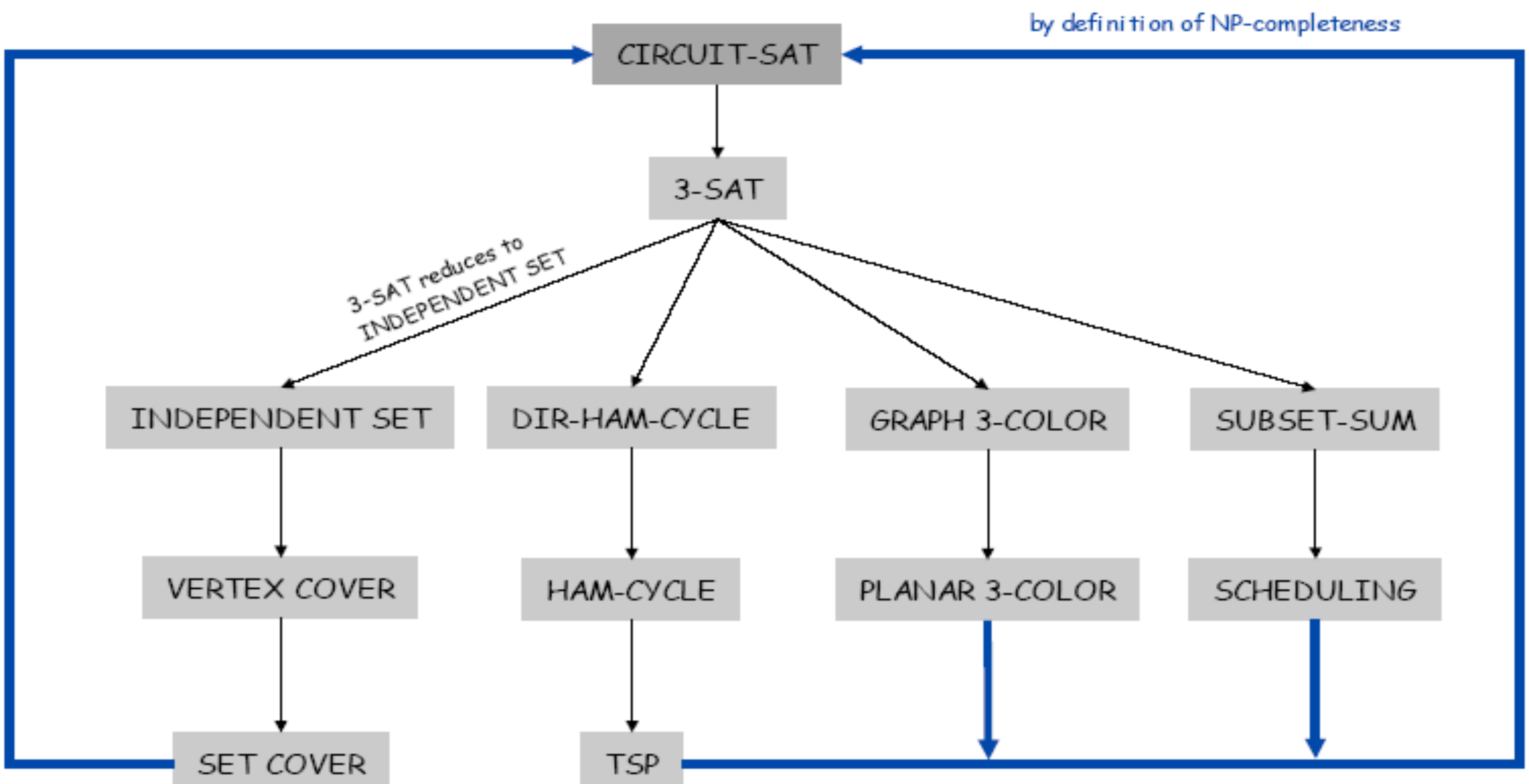
证明NP完全性的通用思路

1. 证明$X \in NP$
2. 选择一个已知的NP完全问题Y
3. 考虑问题Y的**任意**一个实例S_Y, 说明如何在多项式时间内构造X的一个实例S_X：

   S_X和S_Y有相同的答案。

NP完全问题

- 根据前面的多项式归约性质，我们可以得到更多的NP完全问题



NP完全问题

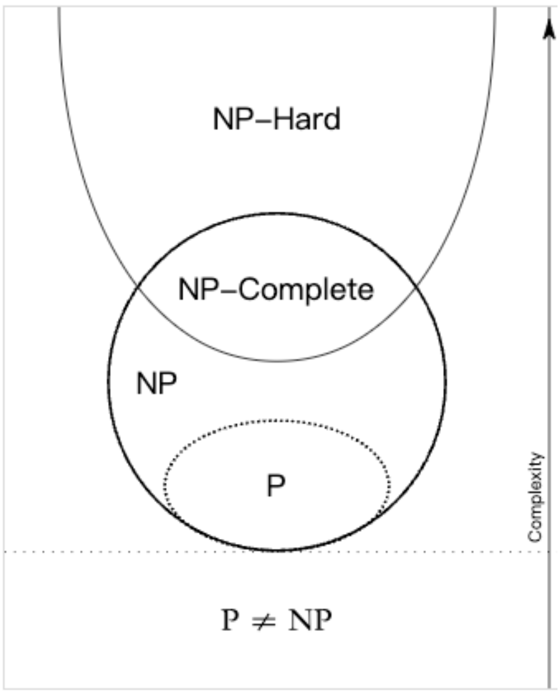
- 难问题的部分分类

1. 包装问题：独立集，集合包装
2. 覆盖问题：顶点覆盖，集合覆盖
3. 划分问题：三维匹配，图着色
4. **排序问题**：哈密尔顿圈，哈密尔顿回路，巡回售货员问题
5. 数值问题：子集和，带开放时间和截止时间的调度问题
6. 约束满足问题：SAT,3SAT

- 一般的经验是，NP问题要么是P,要么是NP完全问题

P & NP & NPC & NPH

- http://www.matrix67.com/blog/archives/105
- https://yzhang-gh.github.io/notes/others/p-np.html#np-%E5%AE%8C%E5%85%A8%E9%97%AE%E9%A2%98



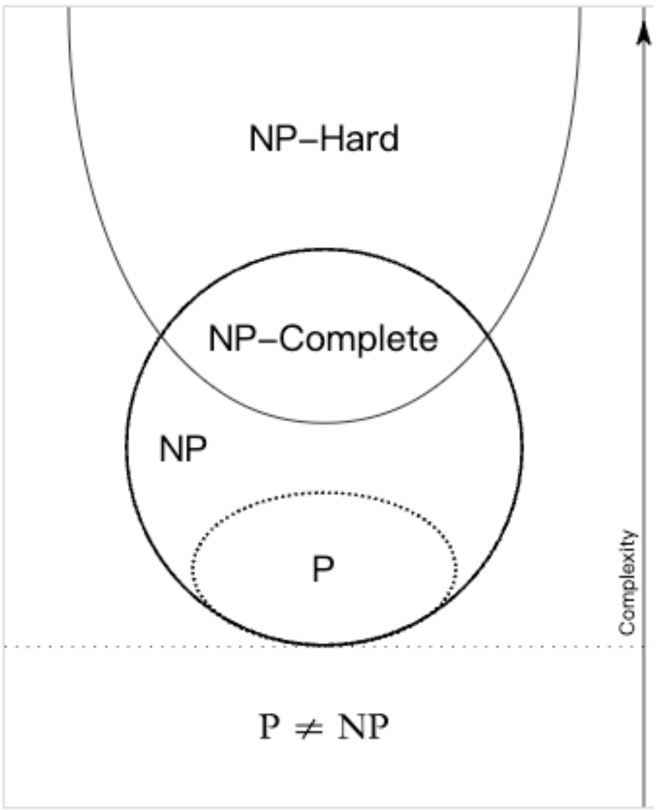
▪**NP-Hard问题: 不要求**是否能在多项式时间里验证一个解的问题，所有的NP问题可以（在多项式时间内）规约到它

▪**NP-Complete问题:** 可以在多项式时间里验证一个解的问题，且所有的NP问题可以（在多项式时间内）规约到它

▪**NP 问题:** 可以在多项式时间里验证一个解的问题

▪**P 问题:** 可以在多项式时间找出解的问题

P & NP & NPC & NPH



- **NP-Hard问题: 不要求**是否能在多项式时间里验证一个解的问题，所有的NP问题可以（在多项式时间内）规约到它
  - NPC问题；
    - **旅行商问题 (TSP)**
      也即**最短路径问题**，已知一系列城市以及两两之间的距离，经过每个城市恰好一次又返回到起点的**最短路径**是哪条（其决策版本「是否存在一条长度小于 ℓ 的路径」则缩小为 **NP 完全**问题）
    - **停机问题**：给定一个程序和其输入，判断这个程序会不会一直运行下去。SAT 问题可以归约为停机问题（**NP-hard** ✓）。而停机问题是不可判定问题，显然不是 **NP 完全**问题。
- **NP-Complete问题:** 可以在多项式时间里验证一个解的问题，且所有的NP问题可以（在多项式时间内）规约到它
  - **布尔可满足性问题** (Boolean satisfiability problem, aka **SAT**)——第一个被证明是 **NP 完全**的问题。
    对于一个由若干布尔变量构成的表达式 (formula)，问是否存在一组变量取值使得表达式为 True（也即 **satisfiable**）
  - 前文提到的**哈密顿环 / 路径问题**
  - **图着色问题**：用 k 种颜色给图的顶点着色，要求相邻顶点不能是相同颜色
- **NP 问题:** 可以在多项式时间里验证一个解的问题
  - P问题+NPC问题 此外，在 **NP** 问题中还存在一些问题，目前人们既不能确定其是不是 **P**，也不确定是不是 **NP 完全**，比如前面提到的整数分解问题，目前已知的最好算法比指数时间要快，比多项式时间要慢。
- **P 问题:** 可以在多项式时间找出解的问题
  - 排序算法：冒泡 $O(n^2)$，归并 $O(n\log n)$