

# 算法设计与分析

Computer Algorithm Design & Analysis

计算机学院  
赵峰



# Chapter 34

## NP-Completeness

---

NP 完全性

# 易解问题和难解问题

- 多项式时间算法。
- 通常将存在多项式时间算法的问题看做是易解问题，而将需要超多项式时间算法的问题看做是难解问题。
- 多项式时间算法具有闭包性、独立性

# P类问题和NP类问题

- P类问题和NP类问题的严格定义来源于形式语言的识别问题。
  - 基于某种计算模型给出：图灵机模型、RAM模型
  - 语言识别问题是一种特殊的判定问题
- 考虑判定问题
  - 因为判定问题更容易地表达为语言的识别问题，从而方便在某种计算模型上求解。

# 判定问题

一个判定问题(decision problem)是仅要求回答“是”或“否”的问题（或者答案是“1”或“0”的问题）。

## ■图灵停机问题

■**哈密顿回路问题**：在一个图 $G=(V,E)$ 中，求从某个顶点出发，经过所有顶点一次且仅一次，且回到出发点的回路称为**哈密顿回路问题**。确定在图 $G$ 中是否有一个哈密顿回路的问题是一个判定问题。

■**TSP问题**：对给定的一个带权图 $G$ 和一个正整数 $k$ ，确定是否有一个经过所有顶点一次且仅一次再回到出发点的回路，其总距离小于 $k$ 的问题是一个判定问题。

# 判定问题的一个重要特征

问题求解困难

问题判定简单。

如：

- 求解哈密顿回路是个难解问题，但验证一个给定的顶点序列是不是哈密顿回路却是很容易的。
- 求解一个线性方程组可能是困难的，但验证一组解是不是方程组的解却很容易。

# 判定问题与最优化问题

**最优化问题**：该类问题的每一种可能的解都有一个相关的值，我们的目标是找出其中具有最优值的可行解。

- 如，**SHORTEST-PATH问题**：已知一个无向图 $G$ 及顶点 $u$ 和 $v$ ，求 $u$ 到 $v$ 之间经过最少边的一条路径。

■ **NP完全性适合于判定问题，但不直接适合于最优化问题。**

- 联系：

通过对待优化的问题强加一个界，就可以把一个给定的最优化问题转化为一个相关的判定问题。

- 如，**SHORTEST-PATH问题**：判定给定的有向图，在结点 $u$ 和 $v$ 之间是否存在一条至多包含 $k$ 条边的路径。

# 从判定问题到最优化问题：

判定问题 “更容易一些”：证明最优化问题是一个 “困难的问题” 时，可以利用该问题与相关的判定问题之间的关系来说明：

如果我们能提供足够的证据表明某个判定问题是个困难问题，就等于提供了证据表明与其相关的最优化问题也是困难的——因为判定问题 “更容易一些”。

- 注：引入判定问题是为了说明最优化问题是困难的，不是为了说明最优化问题可解。



# 确定性算法和P类问题

## 定义1 确定性算法

设A是求解问题 $\Pi$ 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。

## 定义2 P类问题

如果对于某个判定问题 $\Pi$ ，存在非负整数 $k$ ，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 $\Pi$ 是一个P类问题。

# 非确定性算法和NP类问题

## 定义3 非确定性算法

设A是求解问题II的一个算法，如果算法A以如下方式工作，就成算法A是一个非确定性算法。

(1)猜测：对问题的输入实例产生一个任意字符串y，在算法的每一次运行时，串y的值可能不同，因此猜测是非确定性的方式工作。

(2)验证：用一个确定性算法验证两件事：

检查在猜测阶段产生的串y的形式是否合适，如果不是，则算法停下来并得到no；

串y是合适的形式，那么算法验证它是否是问题的解，如果是问题的解，则算法停下来并得到yes，否则，算法得到no。

例，TSP问题的判定形式：

假定算法A是求解TSP判定问题的非确定性算法。

算法A以非确定性的形式猜测一个路径是TSP判定问题的解。

用确定性算法检查这个路径是否经过所有结点一次且仅一次并返回出发点。

- 如果答案为yes，则继续验证这个回路的总长度是否 $\leq k$ ；

- 如果答案仍是yes，则算法A输出yes，否则输出no。

- 但如果算法A输出no，并不意味着不存在满足要求的回路，因为算法的猜测可能是不正确的；

算法输出yes，当且仅当对于TSP判定问题的某个输入实例至少存在一条满足要求的回路。

## 定义4 NP类问题

如果对于某个判定问题 $\Pi$ ，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或no的答案，则该判定问题是一个NP类问题。

NP类问题是难解问题的一个子类，并不是任何一个在常规计算机上需要指数时间的问题（即难解问题）都是NP类问题。

如：汉诺塔问题

# 汉诺塔的传说

在印度，有一个古老的传说：在世界中心**贝拿勒斯**（在印度北部）的圣庙里，有一块黄铜板上插着三根宝石针。印度教的主神**梵天**在创造世界的时候，在其中一根针上从上到下地穿好了由大到小的64片金片，这就是所谓的**汉诺塔**。

不论白天黑夜，总有一个僧侣按照下面的法则移动这些金片：

- 一次只移动一片
- 不管在哪根针上，小片必须在大片上面。

僧侣们预言，当所有的金片都从梵天穿好的那根针上移到另外一根针上时，世界就将在一声霹雳中。

需要多少次移动呢？

假设对n片金片，移动次数是 $f(n)$

$$f(1)=1, f(2)=3, f(3)=7,$$

$$f(k+1)=2*f(k)+1。$$

$$f(n)=2^n-1。$$

则，当 $n=64$ 时， $f(64)=2^{64}-1=18446744073709551615。$

设想，即使每秒钟可以移一次，则要需5800多亿年才能将64片金片从梵天穿好的那根针上移到另外一根针上。——宇宙可能真的毁灭

汉诺塔问题不是NP类问题。 Why?

验证： $O(2^n)$

- P类问题和NP类问题的区别在于：
  - 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非确定性算法来进行判定或求解。

# 哈密顿回路问题

## ■图G中是否有一条哈密顿回路？

已知一个问题实例，一种可能的判定算法是罗列出G的顶点的所有排列，然后对每一种排列进行检查，以确定它是否是一条哈密顿回路。

设顶点数为 $m$ ，则总共有 $m!$ 种可能，因此算法的运行时间为指数级。



## 哈密顿回路判定问题：

假设有人说某个给定的图G是哈密顿图，提出可以通过给出沿哈密顿回路排列的顶点来证明它。

**证明是容易的：** 仅需要检查所提供的回路是否是V中顶点的一个排列，以及沿回路的每条连续的边是否在图中存在。

■验证时间： $O(n^2)$ 。

# 非确定性算法与NP类问题

- 如果问题 $\Pi$ 属于P类问题，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。显然也可以构造非确定性算法验证其正确性。因此，问题 $\Pi$ 也属于NP类问题，即  $P \subseteq NP$ 。
- 如果问题 $\Pi$ 属于NP类问题，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。但是不一定能构造多项式时间的确定性算法对它进行求解或判定。因此， $P \neq NP?$  至今无法证明。

# NP完全问题 (NPC)

NP问题的一个子类。对这个子类中的任何一个问题，如果能够证明可以用多项式时间的确定性算法来进行求解，那么NP完全问题中的所有问题都可以通过多项式时间的确定性算法来进行求解或判定。

我们希望能够在NP类问题的内部找到一种方法，比较两个问题的计算复杂性，并进一步找到NP类问题中最难的问题。

- 比较两个问题的计算复杂性的方法是问题变换（归约）。

# 问题变换与计算复杂性归约

**定义5** 假设问题  $\Pi'$  存在一个算法  $A$ ，对于问题  $\Pi'$  的输入实例  $I'$ ，算法  $A$  求解问题  $\Pi'$  得到一个输出  $O'$ ；另外问题  $\Pi$  的输入实例是  $I$ 。对应于输入  $I$ ，问题  $\Pi$  有一个输出  $O$ ，则问题  $\Pi$  变换到  $\Pi'$  是一个3个步骤的过程：

- **输入转换**：把问题  $\Pi$  的输入  $I$  转换为问题  $\Pi'$  的适当输入  $I'$ ；
- **问题求解**：对问题  $\Pi'$  应用算法  $A$  产生一个输出  $O'$ ；
- **输出转换**：把问题  $\Pi'$  的输出  $O'$  转换为问题  $\Pi$  对应输入  $I$  的正确输出  $O$ 。

- 若在 $O(\tau(n))$ 的时间内完成上述输入和输出的转换，则称问题 $\Pi$ 以 $\tau(n)$ 时间变换到问题 $\Pi'$ ，记为 $\Pi \propto_{\tau(n)} \Pi'$ ，其中 $n$ 为问题规模；
- 若在多项式时间完成上述输入到输出转换，则称问题 $\Pi$ 以**多项式时间**转换到问题 $\Pi'$ ，记为 $\Pi \propto_p \Pi'$ 。

例：设算法A可以求解排序问题。输入 $I'$ 是一组整数 $X=x_1, x_2, \dots, x_n$ ，输出 $O'$ 是这组整数的一个排列 $x_{i1} \leq x_{i2} \leq \dots \leq x_{in}$ 。

考虑**配对**：输入 $I$ 是两组整数 $X=x_1, x_2, \dots, x_n$ 和 $Y=y_1, y_2, \dots, y_n$ 。

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，次小值与次小值配对，依次类推。

办法：使用一个已存在的排序算法A，首先将这两组整数排序，然后根据它们的位置进行配对，**将配对问题转换为排序问题**：

- (1) 把配对问题的输入I转化为排序问题的两个输入；
- (2) 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 $O_1'$  和  $O_2'$  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。

建立配对问题代价的一个上限：**转换都是多项式时间内完成的，所以如果排序问题有多项式时间的算法，则配对问题也一定有多项式时间的算法。**

**注：**问题变化的主要目的不是给出解决问题的算法，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。

定理2 若已知问题 $\Pi$ 的计算时间下限是 $T(n)$ 且问题 $\Pi$ 可 $\tau(n)$ 变换到问题 $\Pi'$ ，即  $\Pi \propto_{\tau(n)} \Pi'$ ，则 $T(n) - O(\tau(n))$ 为问题 $\Pi'$ 的一个计算时间下限。

定理3 若已知问题 $\Pi'$ 的时间上限是 $T(n)$ ，且问题 $\Pi$ 可 $\tau(n)$ 变换到问题 $\Pi'$ ，即  $\Pi \propto_{\tau(n)} \Pi'$ ，则 $T(n) + O(\tau(n))$ 为问题 $\Pi$ 的一个计算时间上限。

定理4 设 $\Pi$ 、 $\Pi'$ 和 $\Pi''$ 是3个判定问题，若  $\Pi \propto_p \Pi'$  且  $\Pi' \propto_p \Pi''$ ，  
则  $\Pi \propto_p \Pi''$ 。

## 证明:

设A是一个对于某个多项式 $p$ , 在 $p(n)$ 步内实现  $\Pi \propto_p \Pi'$  的算法;

设B是一个对于某个多项式 $q$ , 在 $q(n)$ 步内实现  $\Pi' \propto_q \Pi''$  的算法。

设 $x$ 是算法A的一个规模为 $n$ 的输入。

显然, 算法A以输入 $x$ 运行(转换)时, 其输出的大小不能超过 $cp(n)$ 。如果算法B接受规模大小为 $cp(n)$ 的输入时, 存在某个多项式 $r$ , 它的运行(转换)时间 $O(q(cp(n))) = O(r(n))$ , 由此, 从问题 $\Pi$ 到问题 $\Pi''$ 是一个多项式变换 证毕。

一般来说, 问题变换不是一个可逆的过程, 如果问题 $\Pi$ 和问题 $\Pi'$ 可相互转换, 即  $\Pi \propto_{r(n)} \Pi'$  且  $\Pi' \propto_{r(n)} \Pi$ , 则称问题 $\Pi$ 和问题 $\Pi'$ 是 $r(n)$ 时间等价的。



# NPC问题

有大量的问题具有这样的特性：存在多项式时间的非确定性算法，但不知道是否存在多项式时间的确定性算法。同时也不能证明这些问题中的任何一个不存在多项式时间的确定性算法，这类问题称为NP完全问题。

**定义6** 令 $\Pi$ 是一个判定问题，如果问题 $\Pi$ 属于NP类问题，并且对于NP类问题中的每一个问题 $\Pi'$ ，都有  $\Pi' \leq_p \Pi$ ，则称判定问题 $\Pi$ 是一个**NP完全问题**，简称NPC (NP-Complete)。

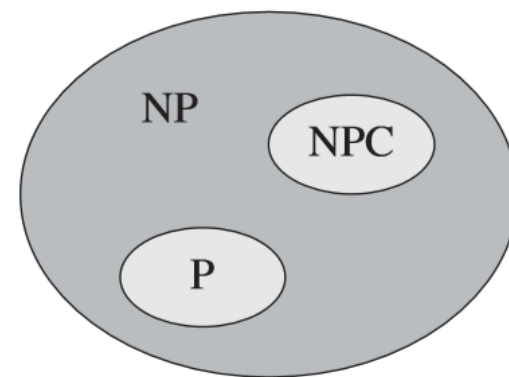
## ■NP难 (NP-hard) 问题

- NP完全问题是NP中最难的一类□任何一个问题都没有找到多项式时间算法。

- 对  $P \neq NP$  问题的研究都是以NP完全问题为中心的。
- 大部分从事理论研究的计算机科学家都认为  $P \neq NP$ ，即NPC类中的所有问题，都不存在多项式时间算法。

- 如果一个NPC问题能在多项式时间内得到解决，那么，NP问题都可以在多项式时间内求解，即  $NP=P$ 。

- 但是谁也没有找出关于任何NP完全问题的多项式时间的算法



大多数理论计算机科学家眼中的P、NP和NPC三者之间的关系，P和NP都完全包含在NP内，且  $P \cap NPC = \emptyset$

**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为问题  $\Pi$ ，但是并不能证明问题  $\Pi$  是NP类问题，所以问题  $\Pi$  不是NP完全的。但是至少与任意NP类问题有同样的难度，称为NP难问题。

**定义7** 如果对于NP类问题中的每一个问题  $\Pi'$  都有  $\Pi' \leq_p \Pi$ ，则称问题  $\Pi$  是一个NP难问题 (NP-hard)。

**一般若判定问题属于NP完全问题，则相应的最优化问题属于NP难问题。**

- 判定图G中是否存在汉密顿回路是NP完全问题，而求解汉密顿回路中的TSP问题是NP难问题。
- 判定图  $G=(V,E)$  中是否存在  $k$  个顶点的团问题是NP完全问题，而求顶点数最多的团问题则是NP难问题。

# 算法导论：

- P类中包含的是多项式时间内可解的问题，即一些可以在 $O(n^k)$ 时间内求解的问题。

- $k$ 为常数， $n$ 是规模。

- NP类中包含的是在多项式内“可验证”的问题：给出某一解决方案的“证书”，能够在问题输入的规模的多项式时间内，验证该证书是否是正确的。

- P类问题属于NP，因为如果某一问题是属于P的，则可以在不给出证书的情况下，在多项式时间内解决它。

- 关于形式语言体系的描述
- 如果一个问题属于NP，且与NP中的任何问题是一样“难”的，则说它属于NPC类□也称为NP完全的。

## ■ 证明NP完全问题概述

- 证明某一特定问题是NP完全问题时，就是在陈述它是一个多么困难的问题。
- 并不是要证明存在某个有效的算法去求解该问题，而是要证明不太可能存在一个有效的算法。

# 基本的NP完全问题

证明一个判定问题 $\Pi$ 是NP完全问题两个步骤 $\square$

(1)证明问题 $\Pi$ 属于NP类问题。即在多项式时间以确定性算法验证一个猜测的解。

(2)证明NP类问题中的每一个问题在多项式时间变换为问题 $\Pi$ 。多项式问题变换有传递性，只需证明一个已知的NP完全问题能够在多项式时间变换即可。

- 证明是NP的**：多项式时间以确定性算法验证一个解。
- 证明是NP难的**：证明已知的NP完全问题多项式时间转化为要证明的问题。

# 第一个NPC问题

1971年，Cook证明了布尔可满足性问题) 是NP完全的一个——Cook定理。

- 1972年，Karp证明了十几个问题都是NP完全的。
- 这些NP完全问题的证明为以后问题的证明奠定了基础。

注：如果证明了至少有一个问题是NP完全问题，就可以用多项式时间可归约性作为工具，来证明其他问题也具有NP完全性。

- 可满足性问题就是这第一个NPC问题。

## 34.3电路可满足性 (CIRCUIT-SAT)

布尔组合电路是由布尔组合元素通过电路互联后构造而成的。

布尔组合元素是任何一种电路元素，有固定输入和输出，执行某种良定义的函数功能。

➤ 布尔值集  $\{0, 1\}$ 。

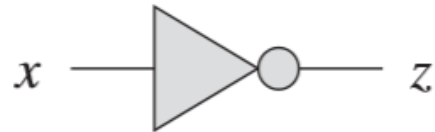
布尔组合元素计算简单的布尔函数。

➤ 这些元素称为逻辑门：

NOT 门（非门）、AND 门（与门）、OR 门（或门）

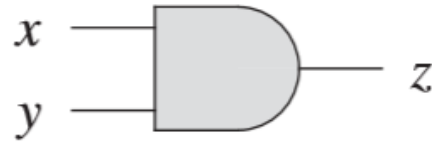
用真值表描述操作的结果。





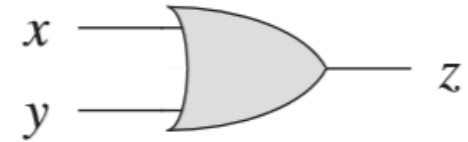
$x$	$\neg x$
0	1
1	0

(a)



$x$	$y$	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

(b)



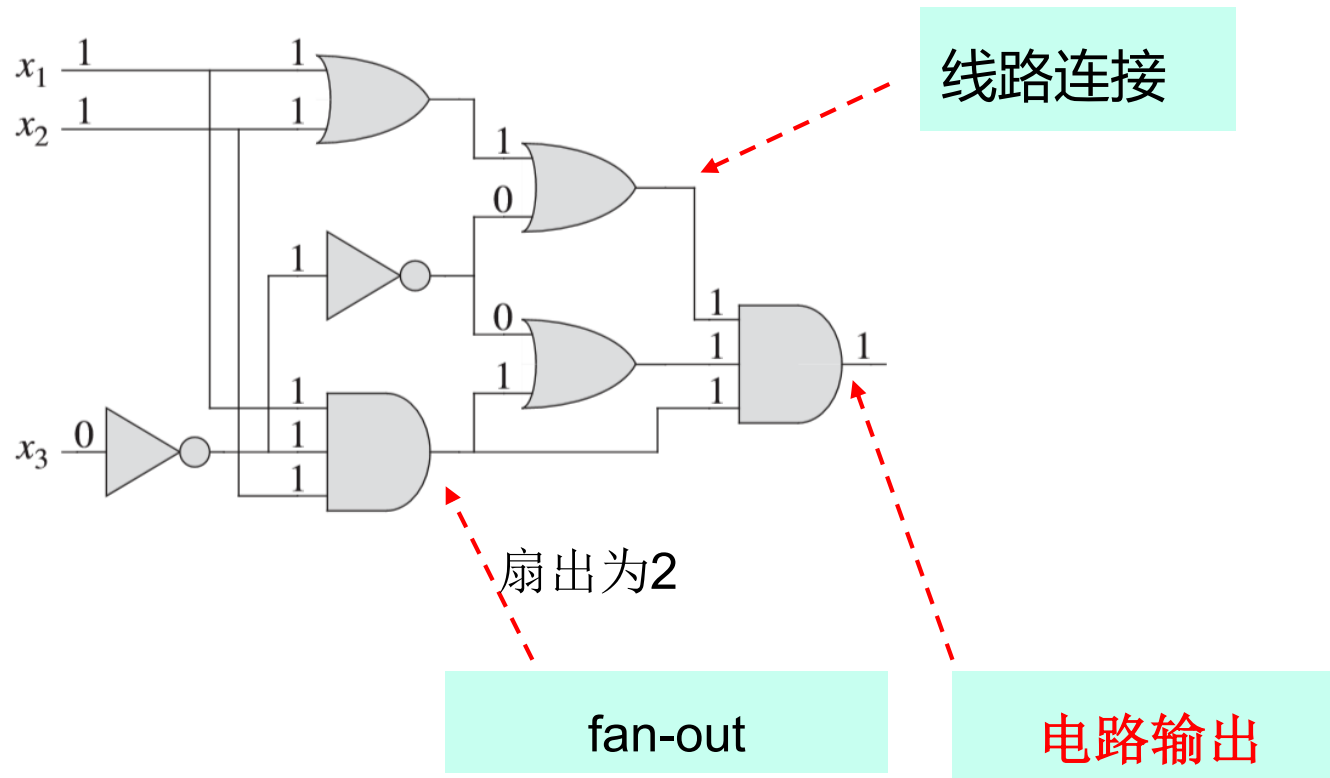
$x$	$y$	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

(c)

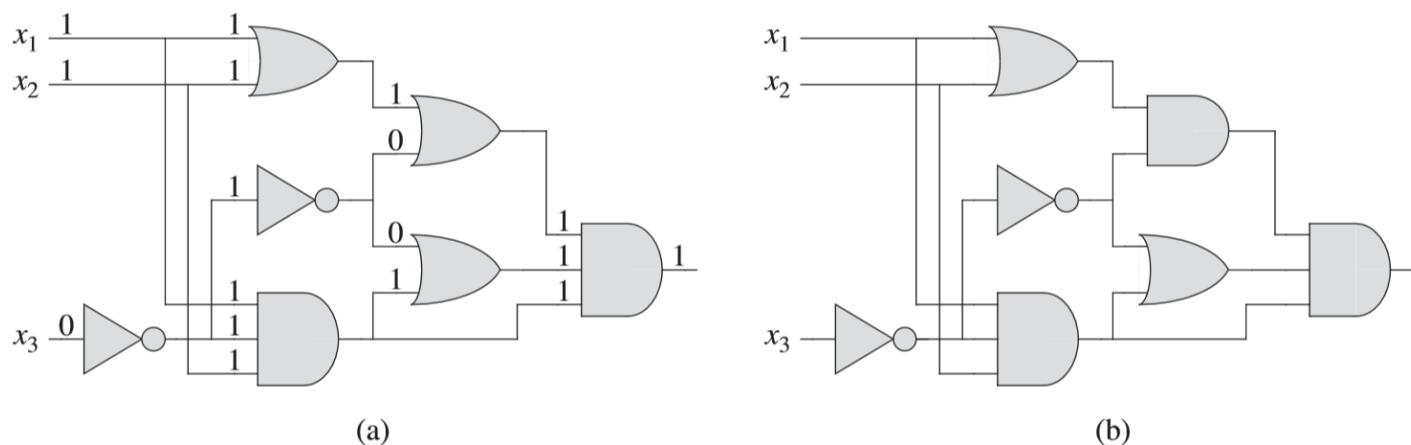
**Figure 34.7** Three basic logic gates, with binary inputs and outputs. Under each gate is the truth table that describes the gate's operation. **(a)** The NOT gate. **(b)** The AND gate. **(c)** The OR gate.

# CIRCUIT-SAT

布尔组合电路由布尔组合元素通过线路连接而成。



- 电路真值赋值
- 如果单输出布尔组合电路具有可满足性赋值，即输出为1的真值赋值，称电路是可满足的。



**Figure 34.8** Two instances of the circuit-satisfiability problem. **(a)** The assignment  $\{x_1 = 1, x_2 = 1, x_3 = 0\}$  to the inputs of this circuit causes the output of the circuit to be 1. The circuit is therefore satisfiable. **(b)** No assignment to the inputs of this circuit can cause the output of the circuit to be 1. The circuit is therefore unsatisfiable.

- 电路 (a) 可满足
- 电路 (b) 不可满足

**电路可满足性问题**：是否是可满足电路。

- 这里，电路可满足性问题是算法导论证明的第一个NP完全问题。
- 后面假设该问题已经被证明过。
  - (证明过程略) □

## 34.4 布尔公式可满足性 (SAT问题)

公式可满足性问题描述如下：

- SAT的一个实例是一个布尔公式  $\phi$ ：

- $n$ 个布尔变量：  $x_1, x_2, \dots, x_n$ ；

- $m$ 个布尔连接词：布尔函数，有一个或两个输入和一个输出，如  $\vee$  (或)，  $\wedge$  (与)，  $\neg$  (非)，  $\rightarrow$  (蕴含)， (当且仅当)

- 括号。

- 布尔公式  $\phi$  的真值赋值：为  $\phi$  中各变量取的一组值；可满足性赋值是指使公式  $\phi$  的值为1。

- 具有可满足性赋值的公式就是可满足公式

# 合取范式的可满足性问题(CNF-SAT问题)

- 合取范式:  $A_1 \wedge A_2 \wedge \dots \wedge A_n$

- 其中, 子句 $A_i$ :  $a_1 \vee a_2 \vee \dots \vee a_n$ ,

- $a_i$ 称为文字。

- SAT问题: 判定是否存在赋值使得范式取值为真。

- 3合取范式的可满足问题

- 每个 $A_i$  ( $i=1, \dots$ ) 恰好都有三个不同的文字

- 析取范式: 交和并对换

## 定理34.9 布尔公式的可满足性问题是NP完全的。

证明：首先证明 $\text{SAT} \in \text{NP}$ ，接着证明 $\text{CIRCUIT-SAT} \leq_p \text{SAT}$

### (1) 证明 $\text{SAT} \in \text{NP}$

由它的每一个可满足性赋值所组成的证书可以在多项式时间内得到验证。

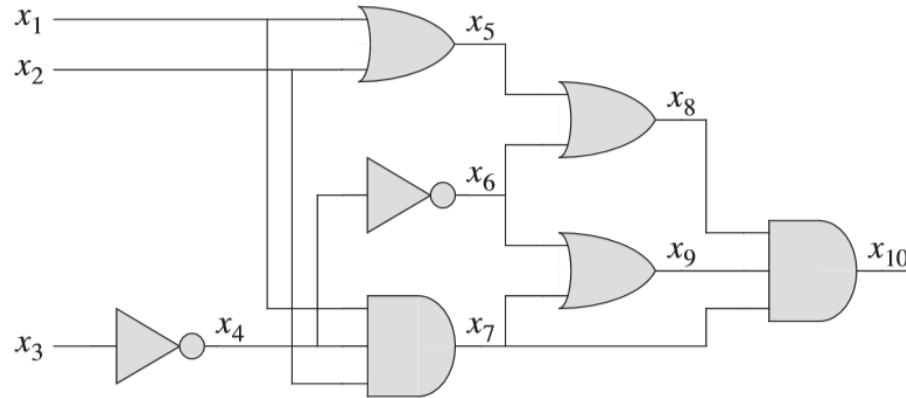
替换变量的值，对表达式求值。——多项式时间

如果表达式的值为1，则说明公式是可满足的。

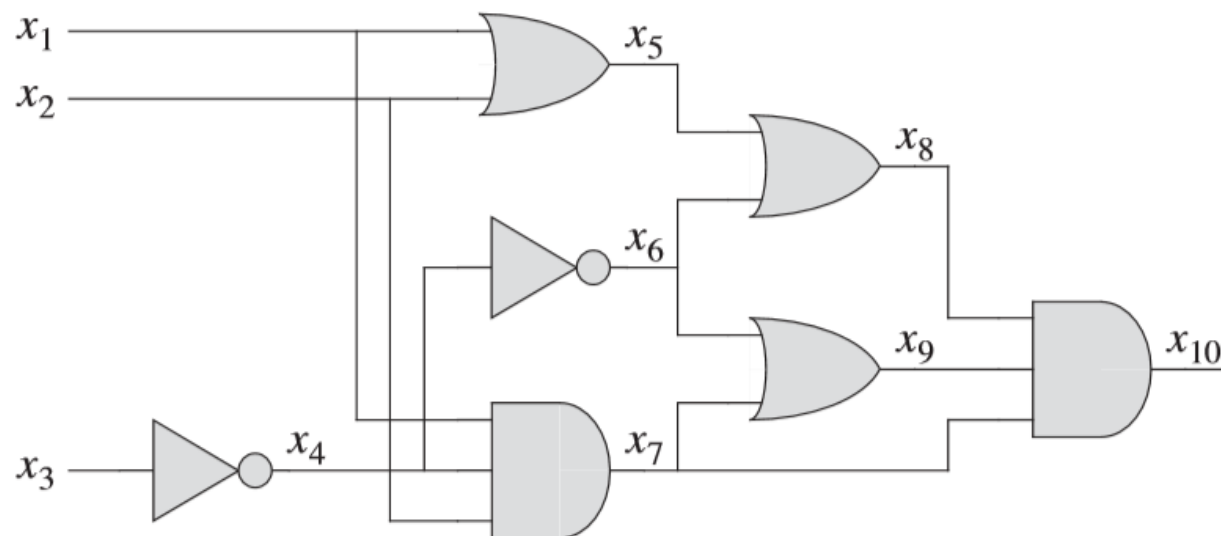
## (2) 证明SAT是NP难的

设CIRCUIT-SAT是我们已知的“第一个” NPC问题，证明  
 $\text{CIRCUIT-SAT} \leq_p \text{SAT}$

CIRCUIT-SAT规约为SAT的基本思想：







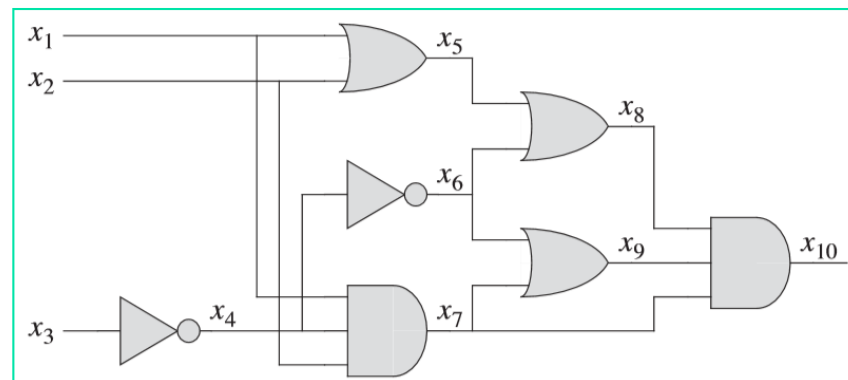
- 对电路C中的每一根接线，公式 $\varphi$ 都有一个对应的变量 $x_i$ 。
- 逻辑门操作表示为关于其附属线路变量的公式。

例如：与门的操作： $x_{10} \quad (x_7 \wedge x_8 \wedge x_9)$ 。

这里，把这些小式子（small formula）称为“子句”  
(clause)

由归约算法产生公式为 $\phi$ :

$$\begin{aligned}\phi = & x_{10} \wedge (x_4 \leftrightarrow \neg x_3) \\ & \wedge (x_5 \leftrightarrow (x_1 \vee x_2)) \\ & \wedge (x_6 \leftrightarrow \neg x_4) \\ & \wedge (x_7 \leftrightarrow (x_1 \wedge x_2 \wedge x_4)) \\ & \wedge (x_8 \leftrightarrow (x_5 \vee x_6)) \\ & \wedge (x_9 \leftrightarrow (x_6 \vee x_7)) \\ & \wedge (x_{10} \leftrightarrow (x_7 \wedge x_8 \wedge x_9)) .\end{aligned}$$



## 转换的正确性

—— 解的**可归约性**，即SAT的解是不是CIRCUIT-SAT的解？

### ■可归约性证明：

如果电路C具有一个可满足性赋值，则电路的每条线路都有一个良定义的值。因此，用线路的值对变量赋值，使得  $\phi$  中每个子句的值为1，因而，所有子句的合取值也为1。

反之，如果存在一个赋值使得  $\phi$  为1，则可证电路C是可满足的。

因此  $\text{CIRCUIT-SAT} \leq_p \text{SAT}$ 。

证毕

## 34.4 3-CNF可满足性

- **文字**：布尔公式中的一个**文字**(literal)是指**变量**或**变量的“非”**。
- **合取范式**：如果布尔公式可以表示为子句的“与”，而每个子句是一个或多个文字的“或”，则称该布尔公式为**合取范式**（**CNF**）。
- **3合取范式**：如果每个子句都是**三个不同的文字**，则称该布尔公式为**3合取范式**（**3-CNF**）。

例：

$$(x_1 \vee \neg x_1 \vee \neg x_2) \wedge (x_3 \vee x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$$

定理34.10 3合取范式的布尔公式的可满足性问题是NPC问题

证明:

(1) 由于3SAT是SAT的特例, 因此3SAT属于NP。

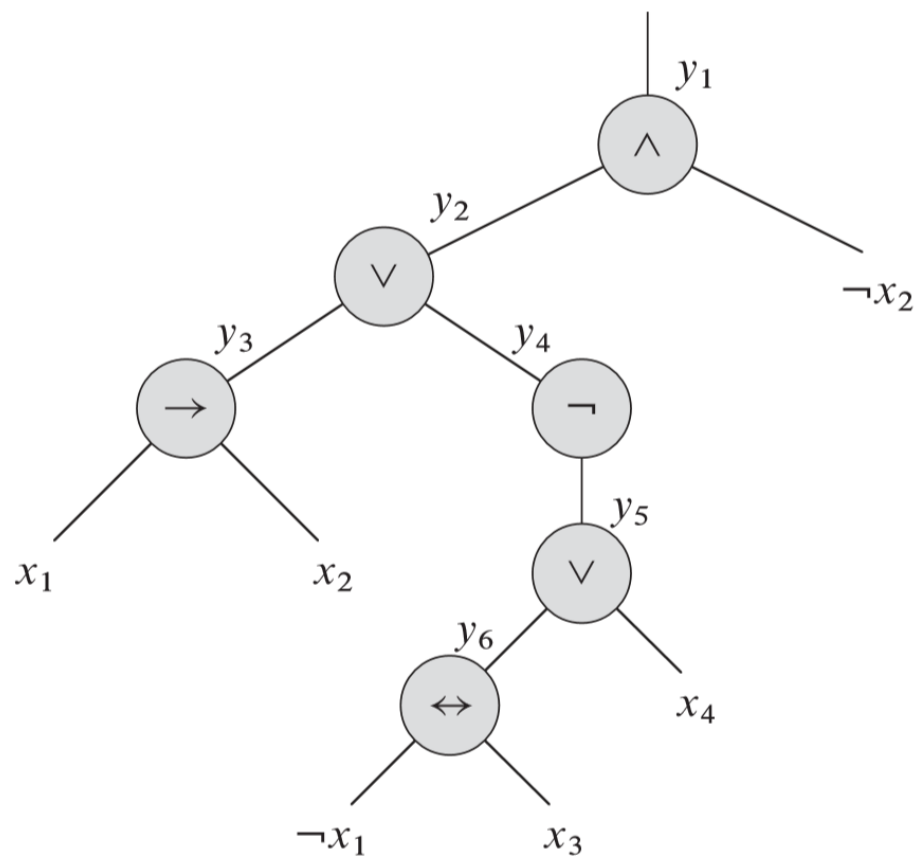
(2)  $SAT \leq_p 3\text{-CNF-SAT}$ :

为输入公式  $\phi$  构造一棵二叉“语法分析树”。

例：

$$\phi = ((x_1 \rightarrow x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge \neg x_2$$

语法分析树：

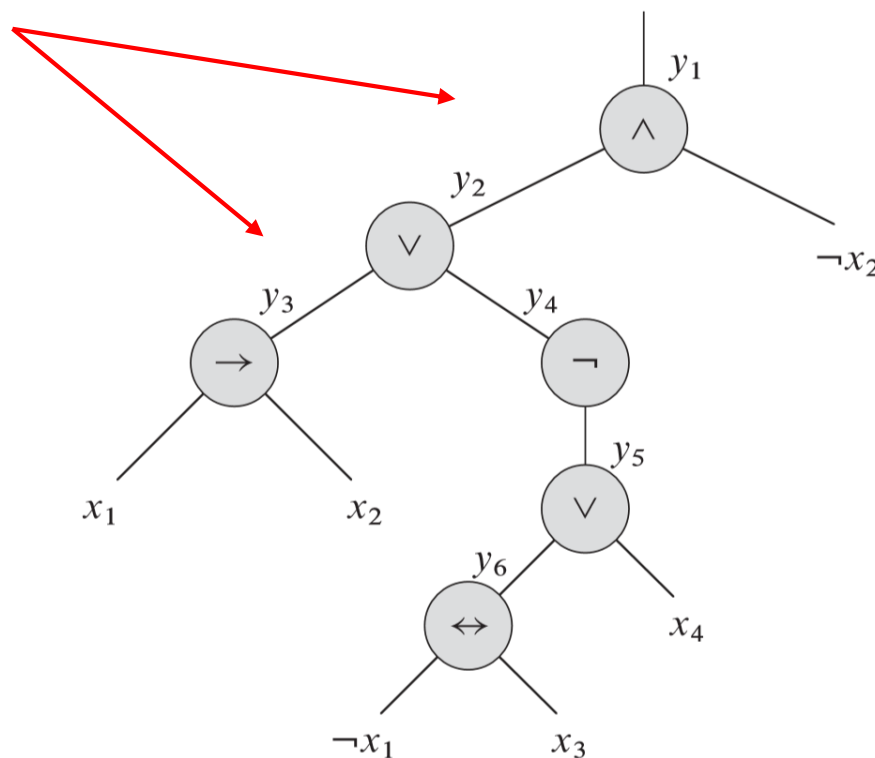


注：利用结合律使树中每个内部节点均有一个或两个孩子。

把二叉语法分析树看作一个电路：

—— 从叶子结点输入，根代表电路的输出。

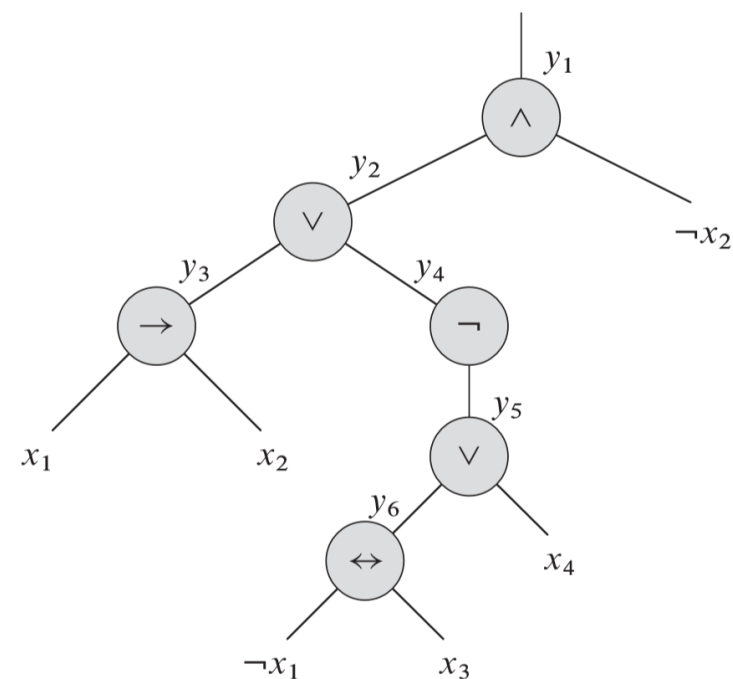
并为每个内部结点的输出引入一个变量 $y_i$



把 $\varphi$ 改写为根变量与描述每个节点操作子句的“与”连接。

改写后表达式为:

$$\begin{aligned}\phi' = & y_1 \wedge (y_1 \leftrightarrow (y_2 \wedge \neg x_2)) \\ & \wedge (y_2 \leftrightarrow (y_3 \vee y_4)) \\ & \wedge (y_3 \leftrightarrow (x_1 \rightarrow x_2)) \\ & \wedge (y_4 \leftrightarrow \neg y_5) \\ & \wedge (y_5 \leftrightarrow (y_6 \vee x_4)) \\ & \wedge (y_6 \leftrightarrow (\neg x_1 \leftrightarrow x_3))\end{aligned}$$



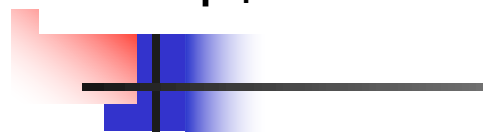


把每个子句 $\varphi_i'$  变换为合取范式。

- 对 $\varphi_i'$  中变量的所有赋值进行计算，构造 $\varphi_i'$  的**真值表**。
- 每一行由子句变量的一种赋值和值组成。
- 用表中**0值项**，构造公式的析取范式。
- 用DeMorgan定律对文字取补，互换“与”、“或”，从而把该公式变成CNF式 $\varphi_i''$ ：

例：子句 $\varphi_1' = (y_1 \vee (y_2 \wedge \neg x_2))$ 变换为CNF。

$\varphi_1'$  真值表。



$y_1$	$y_2$	$x_2$	$(y_1 \leftrightarrow (y_2 \wedge \neg x_2))$
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	1
0	0	0	1

取0值的项，与 $\neg\varphi_1'$  等价的DNF公式为：

$$\begin{aligned} & (y_1 \wedge y_2 \wedge x_2) \vee (y_1 \wedge \neg y_2 \wedge x_2) \\ & \vee (y_1 \wedge \neg y_2 \wedge \neg x_2) \vee (\neg y_1 \wedge y_2 \wedge \neg x_2) \end{aligned}$$

取反的到CNF公式:

$$\begin{aligned}\phi_1'' = & (\neg y_1 \vee \neg y_2 \vee \neg x_2) \wedge (\neg y_1 \vee y_2 \vee \neg x_2) \\ & \wedge (\neg y_1 \vee y_2 \vee x_2) \wedge (y_1 \vee \neg y_2 \vee x_2),\end{aligned}$$

$\phi''$  的子句最多包含3个文字。

第三步，进一步变换使得每个子句恰好有三个不同的文字。

根据 $\phi''$  的子句构造的3-CNF

公式 $\phi'''$ 。

使用两个辅助变量 $p$ 和 $q$ ，对 $\phi'''$ 中的每个子句 $C_i$ ，使 $\phi'''$ 包含下列子句：

- 如果 $C_i$ 有三个不同的文字 $\square$ 则直接作为 $\phi'''$ 的一个子句；
- 如果 $C_i$ 有两个不同的文字，即如果 $C_i = (l_1 \vee l_2) \square$ 则把 $(l_1 \vee l_2 \vee p) \wedge (l_1 \vee l_2 \vee \neg p)$  作为 $\phi'''$ 的子句。
- 加入 $p$ 和 $\neg p$ 满足每个子句必须有三个不同的文字的语法要求，等价于 $(l_1 \vee l_2)$

- 如果 $C_i$ 中仅有一个文字 $l$ ，则把

$$(1 \vee p \vee q) \wedge (1 \vee p \vee \neg q) \wedge (1 \vee \neg p \vee q) \wedge (1 \vee \neg p \vee \neg q)$$

作为 $\phi'''$ 的一个子句。

- 3-CNF公式 $\phi'''$ 是可满足的，当且仅当上述三个步骤的每一步中 $\phi$ 都是可满足的：

第一步根据 $\phi$ 构造 $\phi'$ 的过程保持可满足性；

第二步产生的CNF公式 $\phi''$ 在代数上等价于 $\phi'$ ；

第三步产生的3-CNF公式 $\phi'''$ 也等价于 $\phi''$ 。

（对变量 $p$ 和 $q$ 的任意赋值，上述公式代数上与 $\phi''$ 等价）

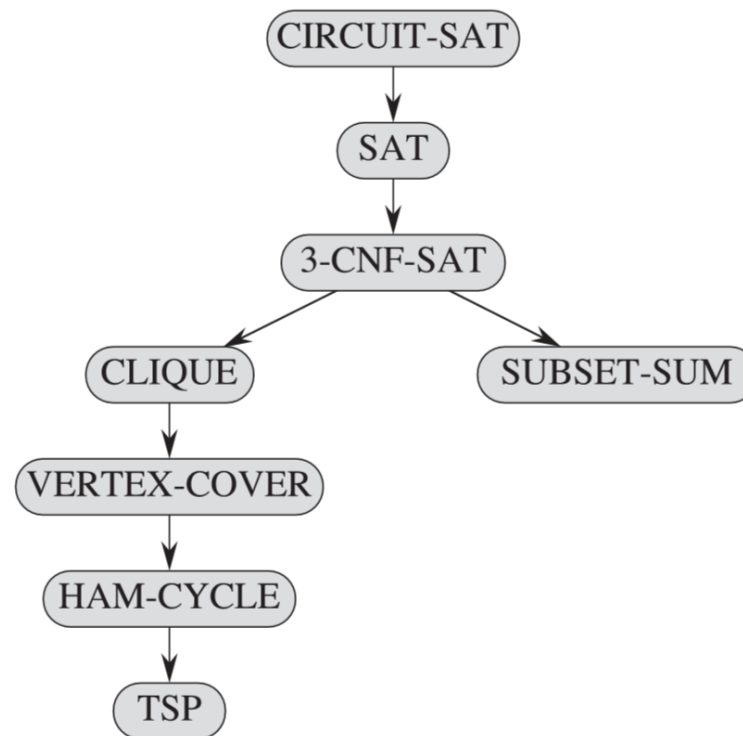
证明归约可以在多项式时间内完成：

- 从  $\Phi$  构造  $\Phi'$ ：每个连接词至多引入了一个变量和一个子句；
- 从  $\Phi'$  构造  $\Phi''$ ： $\Phi'$  中的每一个子句至多引入8个子句。（ $\Phi''$  中的每个子句至多有3个变量，真值表至多有  $2^3=8$  行）
- 从  $\Phi''$  构造  $\Phi'''$ ：每个子句至多在  $\Phi''$  中引入4个子句。

因此，所产生的公式  $\Phi$  是关于原始公式长度的多项式。

## 34.5 NP完全问题

NP完全性证明的流程结构：



CIRCUIT-SAT NPC问题。

## 34.5.1 团问题 (The clique problem )

- **团**：无向图  $G = \langle V, E \rangle$  中的团是一个顶点子集  $V' \subseteq V$ ，其中每一对顶点之间都由  $E$  中的一条边相连。

- **团的规模**：顶点数。

- **团问题**：寻找规模最大的团。

- **判定问题**：在图中是否存在一个给定规模为  $k$  的团？

即：

$$C = \{ \langle G, k \rangle : G \text{ 是一个包含规模为 } k \text{ 的团的图} \}$$



## 团问题的朴素解 (The clique problem) :

列出V的所有k子集, 逐一检查, 看是否成一个团。

■时间复杂度:  $\Omega(k^2 \binom{|V|}{k})$

➤如果k为常数, 则该算法是多项式时间的。

➤但k接近于 $|V|/2$ , 算法的运行时间是超多项式的

——事实上, 团问题的有效算法是不大可能存在的。

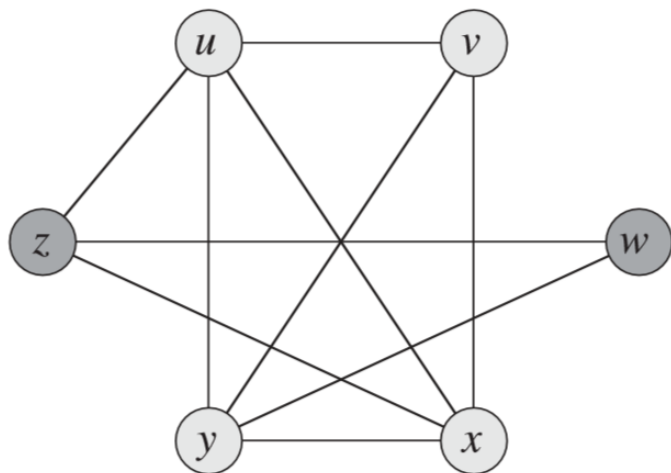
定理34.11 团问题是NP完全的。

证明略。

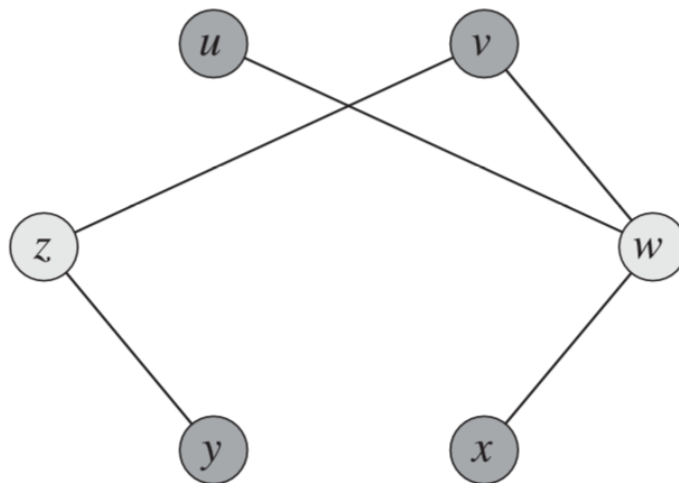
## 34.5.2 顶点覆盖问题

无向图 $G=(V,E)$ 的顶点覆盖是指子集 $V' \subseteq V$ 满足如果 $(u, v) \in E$ , 则 $u \in V'$  或 $v \in V'$  (或两者都成立)。

- 每个顶点“覆盖”与其相关联的边，顶点覆盖是覆盖 $E$ 所有边的顶点所组成的集合。
- **规模**：顶点数。



(a)



(b)

## 顶点覆盖问题：

在给定的图中，找出规模最小的顶点覆盖。

## 问题表述为判定问题：

$\text{VERTEX-COVER} = \{ \langle G, k \rangle : \text{图}G\text{具有规模为}k\text{的顶点覆盖} \}$

## 定理34.12 顶点覆盖问题是NP完全的

证明：（略）

### 34.5.3 汉密顿回路问题

定理34.12 汉密顿回路问题是NP完全问题

证明：（略）

## 34.5.4 旅行商问题□

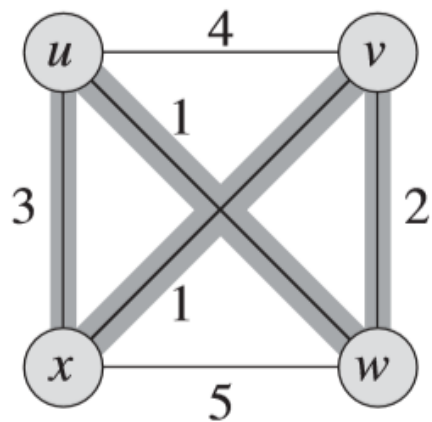
一个售货员必须访问 $n$ 个城市。经过汉密顿回路，恰好访问每个城市一次，并最终回到出发的城市。

从城市 $i$ 到城市 $j$ 的旅行费用为一个整数 $c(i,j)$ ，售货员旅行所需的全部费用是他旅行经过的各边费用之和。售货员希望整个旅行费用最低。

**判定问题：**

$\text{TSP} = \{ \langle G, c, k \rangle : G = (V, E) \text{ 是一个完全图, } c \text{ 是 } V \times V \rightarrow \mathbb{Z} \text{ 上的一个整数, } k \in \mathbb{Z}, G \text{ 是否包含一个费用至多为 } k \text{ 的旅行回路。} \}$

例：



旅行商问题的一个实例，阴影覆盖的边表示费用最低的旅行路线，其费用为7。

定理34.14 旅行商问题是NP完全的

证明：（略）

## 34.5.5 子集和问题

已知一个正整数的有限集 $S$ 和一个整数目标 $t > 0$ ，问是否存在一个子集 $S' \subseteq S$ ，其元素和为 $t$ 。

子集和问题的定义：

$\text{SUBSET-SUM} = \{ \langle S, t \rangle : \text{存在一个子集 } S' \subseteq S, \text{ 满足 } t = \sum_{s \in S'} s \}$ .

定理34.15 子集和问题是NP完全的。

# NP完全问题的计算机处理

(1) 先进的算法设计技术

(2) 充分利用限制条件

许多问题虽然被归结为NP完全问题，有些问题加上限制条件后可能会改变性质。

如：0/1背包问题中，限定物品的重量和价值均为正整数，则利用动态规划，背包问题存在一个伪多项式时间 $O(nW)$ 的算法。



令  $S^i = \{ (P_j, W_j) \mid \text{为从第1件物品到第} i \text{件物品, 在总放置重量不超过} W \text{的前提下, 所有可能的放置方案所获得的效益值和所占重量} \}$

若每件物品的重量  $w_i$  和效益值  $p_i$  均为 **整数**, 则  $S^i$  中每个序偶

$(P, W)$  的  $P$  值和  $W$  值也是整数,  $P \leq \sum_{0 \leq j \leq i} p_j$ ,  $W \leq M$

问题: 在  $1 \sim n$  范围内有多少个互异的整数?

答案:  $n$  个

在任一  $S^i$  中, 所有序偶具有 **互异**  $P$  值和  $W$  值。

故有:  $|S^i| \leq 1 + \sum_{0 \leq j \leq i} p_j$        $|S^i| \leq 1 + \min \{ \sum_{0 \leq j \leq i} w_j, M \}$

至少有一个  $(0, 0)$

故，在所有 $w_j$ 和 $p_j$ 均为整数的情况下，0/1背包问题的时间和空间复杂度将为：

$$O(\min \{2^n, n \sum_{1 \leq i \leq n} \lfloor p_i \rfloor, nW\})$$

- 尽管背包问题的时间复杂度为 $O(nW)$ ，但它仍然是一个NP完全问题。这是因为 $W$ 同问题的输入大小并不成线性关系。
  - 问题的输入大小仅取决于表示输入所需的比特数。事实上， $\log W$ ，即表示 $W$ 所需的比特数，同问题的输入长度成线性关系。
  - 背包问题存在完全逼近多项式时间的方案，但作为NP完全问题，背包问题没有一种既准确又快速（多项式时间）的算法。

### (3)近似算法

很多问题的输入是近似的，同时很多问题的解允许有一定误差。采用近似算法可以在一定时间内得到问题的近似解。

### (4)概率算法

### (5)并行算法

利用多台处理机共同完成一项计算，是解决计算密集型问题的必经之路。

### (6)智能算法

遗传算法、人工神经算法、DNA算法、蚁群算法、免疫算法、模拟退火算法。