



第8章

NP完全性理论



8.1 基本概念

如何区分一个问题是“难”是“易”？

通常将可在多项式时间内解决的问题看作是“易”解问题，而将需要指数时间解决的问题看作是“难”解问题。

确定性算法

在算法中使用的每个操作的结果唯一确定，算法操作的结果也是唯一确定的。

非确定性算法

算法操作结果不唯一，而是来自可能值的集合



8.1 基本概念

■ 图灵机

- 一条无限长的纸带 **TAPE**。纸带被划分为一个接一个小格子，每个格子上包含一个来自有限字母表的符号，字母表中有一个特殊的符号 表示空白。纸带上的格子从左到右依次被编号为 $0, 1, 2, \dots$ ，纸带的右端可以无限伸展
- 一个读写头 **HEAD**。该读写头可以在纸带上左右移动，它能读出当前所指的格子上的符号，并能改变当前格子上的符号
- 一套控制规则 **TABLE**（程序、状态转移函数）。它根据当前机器所处的状态以及当前读写头所指的格子上的符号来确定读写头下一步的动作，并改变状态寄存器的值，令机器进入一个新的状态





图灵机

- 图灵机 $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ 将以如下方式运作：
 - 开始的时候将输入符号串 从左到右依此填在纸带的格子，其他格子保持空白（即填以空白符）。 M 的读写头指向第 0 号格子， M 处于状态 q_0 。机器开始运行后，按照转移函数 δ 所描述的规则进行计算。例如，若当前机器的状态为 q ，读写头所指的格子中的符号为 x ，设 $\delta(q, x) = (q', x', L)$ ，则机器进入新状态 q' ，将读写头所指的格子中的符号改为 x' ，然后将读写头向左移动一个格子。若在某一时刻，读写头所指的是第 0 号格子，但根据转移函数它下一步将继续向左移，这时它停在原地不动。换句话说，读写头始终不移出纸带的左边界。若在某个时刻 M 根据转移函数进入了状态 q_{accept} ，则它立刻停机并接受输入的字符串；若在某个时刻 M 根据转移函数进入了状态 q_{reject} ，则它立刻停机并拒绝输入的字符串。
 - 注意，转移函数 δ 是一个部分函数，换句话说对于某些 q, x ， $\delta(q, x)$ 可能没有定义，如果在运行中遇到下一个操作没有定义的情况，机器将立刻停机



非确定性图灵机

- 非确定型图灵机和确定型图灵机的不同之处在于，在计算的每一时刻，根据当前状态和读写头所读的符号，机器存在多种状态转移方案，机器将任意地选择其中一种方案继续运作，直到最后停机为止。具体而言，其状态转移函数为

$$\delta : Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L, R\}}$$

- 例如，对于非确定图灵机M的当前状态为q，当前读写头的状态x， $\delta(q, x) = \{(q_1, x_1, d_1), \dots, (q_k, x_k, d_k)\}$



8.1.1 非确定性算法

➤ 非确定性机

- 引入三个函数：
 - **Choice(S)** ...选择集合S中的任一元素.
 - **Failure()** ...失败信号,表示失败的终止.
 - **Success()** ...成功信号,表示成功的终止.
 - 当且仅当不存在产生成功信号的一套选择时, 非确定性算法才不能成功终止
 - 三个函数计算时间均为 $O(1)$
- 按此方式执行非确定性算法的机器称为非确定性机



8.1.1 非确定性算法

- 例8.1:给定集合 $A[1:n]$ ($n \geq 1$)中寻找某一元素 x ,确定使 $A[j]=x$ 的索引 j , 如果 x 不在 A 中, 则 $j=0$ 。

- 算法8. 1:

```
int j=Choice(1,n);  
    if(A[j]==x){  
        System.out.println(j);  
        Success();  
    }  
    System.out.println(0);  
    Failure();
```



8.1.1 非确定性算法

例8.2： 设 $A[i](1 \leq i \leq n)$ 是正整数组成的数组，非确定性算法Nsort(A, n)按非降序排序数组并按次序输出数组元素。

■ **算法8.2：**

```
void Nsort (int A[],int n)
{ int B[SIZE], i, j;
  for (i=1; i<=n; i++) B[i]=0;
  for (i=1; i<=n; i++) {
    j = Choice(1,n);
    if (B[j]) Failure(); else B[j] = A[i];
  }
  for (i=1; i<=n-1; i++) if (B[i] > B[i+1]) Failure();
  for ( i=1;i<=n; i++) System.out.print(b[i]);
  System.out.println();
  Success();
}
```




8.1.1 非确定性算法

- 判定问题、判定算法
 - **答案非0即1的所有问题称为判定问题.**
 - **求解判定问题的算法称为判定算法.**
- 最优化问题、最优化算法
 - **确定代价函数最优值的所有问题称为最优化问题.**
 - **求解最优化问题的算法称为最优化算法.**
- 判定问题与最优化问题
 - **最优化问题可在多项式时间求解，则对应判定问题也可以**
 - **判定问题不能在多项式时间求解，则对应最优化问题也不能**



8.1.1 非确定性算法

- 例8.3: 背包判定问题是确定是否存在 $x_i (1 \leq i \leq n)$ 的0/1赋值序列, 满足 $\sum p_i x_i \geq r$ 和 $\sum w_i x_i \leq m$, 其中 r 是一个给定的数, p_i 和 w_i 非负。如果背包判定问题不能在多项式时间求解, 则最优化问题也不能。

- 算法8.3 [背包判定问题]:

```
void DKP (int p[], int w[], int n, int m, int r, int x[])  
{  
    int W=0, P=0;  
    for (int i=1; i<=n; i++) {  
        x[i] = Choice (0,1);  
        W += x[i]*w[i]; P += x[i]*p[i];  
    }  
    if((W>m) || (P<r)) Failure();  
    else Success();  
}
```



8.1.1 非确定性算法

- 非确定性算法的复杂度
 - 在给定的输入下，如果存在序列使算法成功终止，则所需要的时间是成功终止所需的最小步数。如果不能成功终止，所需时间 $O(1)$
 - 如果对于能让算法成功终止的所有大小为 $n(n \geq n_0)$ 的输入，算法所需时间至多是 $cf(n)$ ，其中 c 和 n_0 都是常数。则非确定性算法复杂度为 $O(f(n))$ 。
- 例如：算法8.3的时间复杂度为 $O(n)$ ，如果输入用二进制表示长度为 q ，则时间复杂度为 $O(q)$



非确定性算法

■ 蕴含等值式和等价等值式

$$A \rightarrow B \Leftrightarrow \neg A \vee B$$

$$A \leftrightarrow B \Leftrightarrow (\neg A \vee B) \wedge (A \vee \neg B)$$

■ 例如

A: 你是王思聪

B: 你是富二代



■ 双重否定律和德摩根律

$$\neg\neg A \Leftrightarrow A$$

$$\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$$

$$\neg(A \vee B) \Leftrightarrow \neg A \wedge \neg B$$

■ 分配律

$$A \wedge (B \vee C) \Leftrightarrow (A \wedge B) \vee (A \wedge C)$$

$$A \vee (B \wedge C) \Leftrightarrow (A \vee B) \wedge (A \vee C)$$



- 由有限个简单合取式构成的析取式成为析取范式
- 由有限个简单析取式构成的合取式成为合取范式
- 范式存在定理：任一命题公式都存在着与之等值的析取范式与合取范式
 - 利用之前的表达式，消去 \rightarrow 和 \leftrightarrow
 - 利用分配律



求公式 $(p \rightarrow q) \leftrightarrow r$ 的析取范式与合取范式.

(1) 合取范式:

$$\begin{aligned}(p \rightarrow q) \leftrightarrow r &\leftrightarrow (p \vee q) \leftrightarrow r \\&\leftrightarrow ((\neg p \vee q) \rightarrow r) \wedge (r \rightarrow (\neg p \vee q)) \\&\leftrightarrow (\neg(\neg p \vee q) \vee r) \wedge (\neg r \vee (\neg p \vee q)) \\&\leftrightarrow ((p \wedge \neg q) \vee r) \wedge (\neg p \vee q \vee \neg r) \\&\leftrightarrow (p \vee r) \wedge (\neg q \vee r) \wedge (\neg p \vee q \vee \neg r)\end{aligned}$$

(2) 析取范式

$$\begin{aligned}(p \rightarrow q) \leftrightarrow r &\leftrightarrow ((p \wedge \neg q) \vee r) \wedge (\neg p \vee q \vee \neg r) \\&\leftrightarrow (p \wedge \neg q \wedge \neg p) \vee (p \wedge \neg q \wedge q) \vee (p \wedge \neg q \wedge \neg r) \vee (r \wedge \neg p) \vee (r \wedge q) \vee (r \wedge \neg r) \\&\leftrightarrow (p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge r) \vee (q \wedge r)\end{aligned}$$



8.1.1 非确定性算法

➤ 可满足性

- 命题演算公式: 由字面量以及与操作和或操作构成的表达式.
- 合取范式(CNF): $\bigwedge_{i=1}^k c_i$, 其中 c_i 是子句, c_i 为 $\bigvee l_{ij}$, l_{ij} 为字面量.
- 析取范式(DNF): $\bigvee_{i=1}^k c_i$, 其中子句 c_i 表示为 $\bigwedge l_{ij}$.
- 可满足性问题: 确定在某些变量赋值之后公式是否为真



可满足性问题

- 可满足性问题有几种特殊情况，其中公式需要具有特定结构。
- 文字是一个变量，称为正文字，或变量的否定，称为负文字。子句是文字（或单个文字）的分离。如果一个子句最多包含一个正文字，则该子句称为**Horn**子句。
- 例如， **x_1** 是正文字， **$\neg x_2$** 是负文字， **$x_1 \vee \neg x_2$** 是子句， **$(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge x_1$** 是联合范式的公式；它的第一和第三个条款是**Horn**条款，但它的第二个条款不是。



8.1.1 非确定性算法

- 假定公式以转化为合取范式（**CNF**），**SAT**可满足性问题要求在有限的时间内判定命题逻辑是否可满足。
- 算法**8.4**：判断可满足性的非确定性算法

```
void Eval (cnf E, int n)
{
    int x[SIZE];
    for ( int i=1; i<=n; i++)
        x[i] = Choice(0,1);
    if (E(x,n)) Success();
    else Failure();
}
```

- 所需时间 $O(n)$ 包括选择E的赋值所需时间和确定性的计算E所需时间。该时间与E的长度成正比。



可满足性问题

- SAT特殊情形
 - 2SAT问题：要求每个子句最多只有两个文字
 - Horn字句集的可满足问题
- 约束满足问题(Constraint Satisfaction Problem, CSP)
 - 给定有限变量以及每个变量的取值范围，找出所有变量的值，使得约束条件被满足



- **n皇后问题**
- **CSP表示**, $x_i=j$ 表示第*i*行*j*列有皇后, 满足约束条件 $x_i \neq x_j$, $|i-j| \neq |x_i-x_j|$
- **SAT表示** $p_{ij}=\text{TURE}$ 表示皇后在第*i*行*j*列, 则命题逻辑公式包括如下字句:
 - 每行至少有一个皇后: 对于任意*i*, $p_{i1} \vee \dots \vee p_{in} = 1$
 - 每行至多有一个皇后: 对于任意*i*和*j*, *k*, $\neg p_{ij} \vee \neg p_{ik}$
 - 每列至多有一个皇后: $\neg p_{ij} \vee \neg p_{kj}$
 - 对角线最多有一个皇后: $\neg p_{ij} \vee \neg p_{k(j+k-i)} \neg p_{ij} \vee \neg p_{k(j+i-k)}$



8.1.2 问题变换与计算复杂性归约

假设有2个问题A和B，将**问题A变换为问题B**是指：

- (1)将问题A的输入变换为问题B的适当输入。
- (2)解出问题B。
- (3)把问题B的输出变换为问题A的正确解。

若用 $O(\tau(n))$ 时间能完成上述变换的第(1)步和第(3)步，则称问题A是 $\tau(n)$ 时间可变换到问题B，且简记为 $\mathbf{A} \propto_{\tau(n)} \mathbf{B}$ 。其中的 n 通常为问题A的规模(大小)。

当 $\tau(n)$ 为 n 的多项式时，称问题A可在多项式时间内变换为问题B。特别地，当 $\tau(n)$ 为 n 的线性函数时，称问题A可线性地变换为问题B。



8.1.2 问题变换与计算复杂性归约

问题的变换与问题的计算复杂性归约的关系：

命题1(计算时间下界归约)：若已知问题A的计算时间下界为 $T(n)$ ，且问题A是 $\tau(n)$ 可变换到问题B，即 $A \propto_{\tau(n)} B$ ，则 $T(n) - O(\tau(n))$ 为问题B的一个计算时间下界。

命题2(计算时间上界归约)：若已知问题B的计算时间上界为 $T(n)$ ，且问题A是 $\tau(n)$ 可变换到问题B，即 $A \propto_{\tau(n)} B$ ，则 $T(n) + O(\tau(n))$ 是问题A的一个计算时间上界。

在命题1和命题2中，当 $\tau(n) = o(T(n))$ 时，问题A的下界归约为问题B的下界，问题B的上界归约为问题A的上界。



8.1.2 问题变换与计算复杂性归约

通过问题变换获得问题的计算时间下界的例子：

(1)判别函数问题：给定 n 个实数 x_1, x_2, \dots, x_n 计算其判别函数

$$\prod_{i \neq j} (x_i - x_j)$$

元素惟一性问题可以在 $O(1)$ 时间内变换为判别函数问题。任何一个计算判别函数的算法，计算出判别函数值后，再作一次测试，判断其值是否为0，即可得到元素惟一性问题的解。由命题1即知，元素惟一性问题的计算时间下界 $\Omega(n \log n)$ 也是判别函数问题的一个计算时间下界。

(2)最接近点对问题：给定平面上 n 个点，找出这 n 个点中距离最近的2个点。

在元素惟一性问题中，将每一个实数 x_i , $1 \leq i \leq n$, 变换为平面上的点 $(x_i, 0)$, $1 \leq i \leq n$, 则元素惟一性问题可以在线性时间内变换为最接近点对问题。



集合不相交问题

问题描述： 设 $S1$ 和 $S2$ 是包含 n 个元素的集合，判定两个集合是否相交，即 $S1 \cap S2 = \Phi$ 。

设： $P1$ 代表判定集合不相交问题；

$P2$ 代表排序问题。

则很容易证明 $P1 \propto_{O(n)} P2$ 。

设 $P1$ 的输入为： $S1 = \{a_1, a_2, \dots, a_n\}$, $S2 = \{b_1, b_2, \dots, b_n\}$

(1) 构造 $P2$ 的输入

$$X = \{(a_1, 1), (a_2, 1), \dots, (a_n, 1), (b_1, 2), (b_2, 2), \dots, (b_n, 2)\}$$

(2) 对 X 按字典序排序，得到有序序列 X' ；

(3) 在 X' 中顺序检查是否存在连续的两个元素 $(x, 1)$ 和 $(y, 2)$ ，满足 $x=y$ 。



集合不相交问题

已经知道排序问题的复杂度上界和下界都是 $\Theta(n\log n)$ 。
现在证明集合不相交问题的复杂度下界是 $\Omega(n\log n)$ 。

构造下面排序算法：

设排序算法的输入为 $X=\{x_1, x_2, \dots, x_n\}$

(1) 构造集合不相交问题的输入： 时间 $O(n)$

$S1=\{(x_1,1), (x_2,1), \dots, (x_n,1)\}$, $S2=\{(x_1,2), (x_2,2), \dots, (x_n,2)\}$

(2) 求解P1 ($S1 \cap S2 = \Phi$?) 设时间为 $T(n)$

分析： 设 $Y=\{y_1, y_2, \dots, y_n\}$ 是 X 的有序排列， 则 $(y_i,1) \in S1$, $(y_i,2) \in S2$ 。 任何求解P1的算法都需要比较 $(y_i,2)$ 与 $(y_{i+1},1)$ 。 可以在求解P1输出所比较的元素偶对 $((x_i,1), (x_j,2))$ 。 这样的偶对 最多 $T(n)$ 个。



集合不相交问题

(3) 构造 X 的有序排列： 时间 $O(n+T(n))$

建立一个有向图 $G=(V,E)$, $V=\{x_1, x_2, \dots, x_n\}$ 。对求解P1算法输出的偶对 $((x_i, 1), (x_j, 2))$ ，判断对 x_i 和 x_j 进行比较。如果 $x_i < x_j$ ，则添加一个有向边 $\langle x_i, x_j \rangle$ ；否则添加 $\langle x_j, x_i \rangle$ 。建立图 G 所需时间为 $n+T(n)$ 。

在图的顶点 $\{x_1, x_2, \dots, x_n\}$ 找出最小元素 y_1 ，然后在 y_1 的邻居顶点中找出最小元素 y_2 ，以此类推。所需时间为 $n+T(n)$ 。

因此构造 X 的有序排列过程所需时间为 $2n+2T(n)$ 。

综合上述三个步骤，这个排序算法的时间为 $3n+3T(n)$ 。因为排序问题的下界为 $\Omega(n \log n)$ ，所以 $3n+3T(n) \geq \Omega(n \log n)$ ，即， $T(n) \geq \Omega(n \log n)$ 。故集合不相交问题的下界为 $\Omega(n \log n)$ 。



集合不相交问题

根据集合不相交问题的下界，可以推导出很多有关集合问题的下界。如：

- (1) 求两个集合的交集
- (2) 求两个集合的并集
- (3) 求两个集合的差
- (4) 判断两个集合是否相等
- (5) 元素唯一性问题：

判定序列 $\{x_1, x_2, \dots, x_n\}$ 中元素互不相同



8.2 P类与NP类问题

➤ 多项式时间复杂度

- 若算法A在所有规模为n的输入下计算时间均为 $O(p(n))$ ，其中 $p()$ 是某个多项式，则称算法A具有多项式时间复杂度

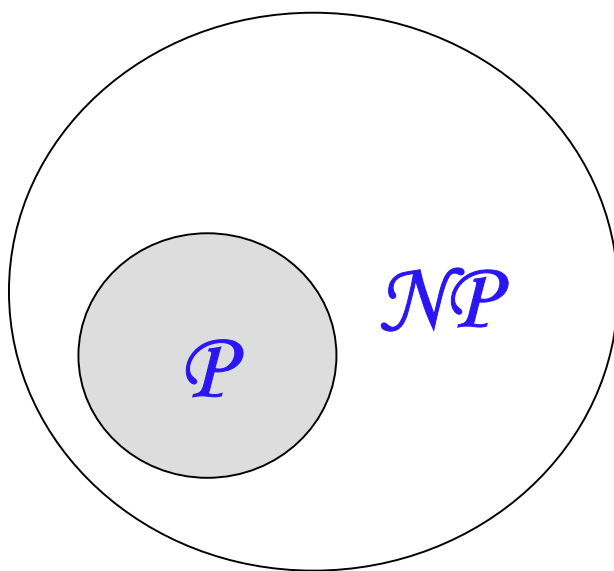
➤ P 与 NP

- P 是在多项式时间内使用确定性算法可解得所有判定问题的集合， NP 是在多项式时间内使用非确定性算法可解的所有判定问题的集合。
- 设一个非确定性算法在时间 $T(n)$ 内可以解决判定问题 P_0 。则存在一个对应的确定性算法，它可在时间 $O(c^{T(n)})$ 内解决 P_0 ，其中 c 是一个常数。



8.2 P类与NP类问题

➤ P 和 NP 的关系: $P \subseteq NP$



P/NP问题 (P versus NP)

霍奇猜想 (The Hodge Conjecture)

庞加莱猜想 (The Poincaré Conjecture)

黎曼猜想 (The Riemann Hypothesis)

杨-米尔斯存在性与质量间隙 (Yang-Mills Existence and Mass Gap)

纳维-斯托克斯存在性与光滑性 (Navier-Stokes existence and smoothness)

贝赫和斯维讷通-戴尔猜想 (The Birch and Swinnerton-Dyer Conjecture)

➤ $P=NP$ 或者 $P \neq NP$?



P类与NP类问题

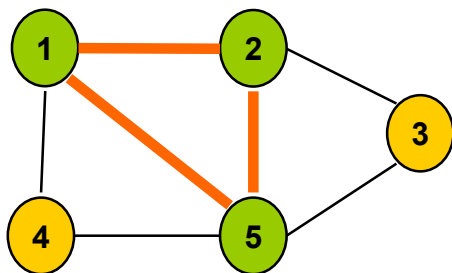
- **P 问题(easy to find), NP 问题(easy to check)**
 - 假设你正在为400名大学生组织住宿，但是空间有限只有100名学生能在宿舍里找到位置。更复杂的是还给了你一份不相容学生的名单，并要求在你的最终选择中不要出现这份名单中的任何一对
 - 斯蒂芬·库克和列昂尼德·莱文在1971年独立地提出了P(即容易找到)和NP(即容易检查)问题。
 - 计算1-1000的连续整数之和 vs 计算地球上所有原子个数之和
 - P类问题是可以在多项式时间内解决并验证的一类问题，NP类问题是可以多项式时间验证但是不确定能否在多项式时间内解决的一类问题。



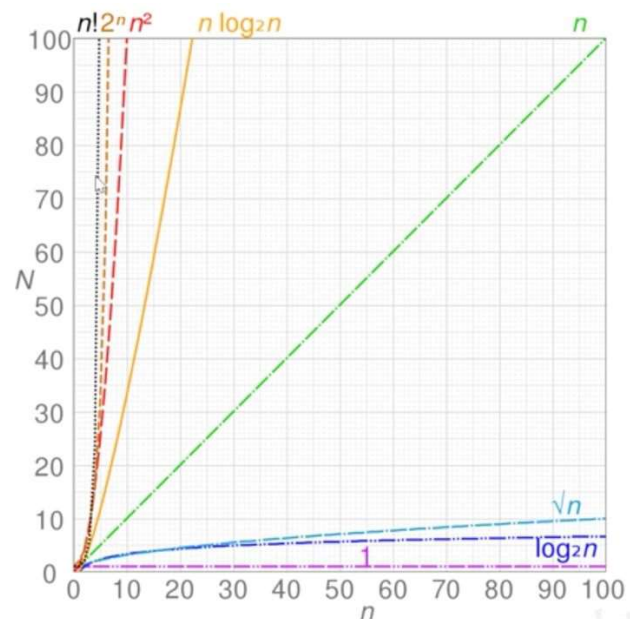
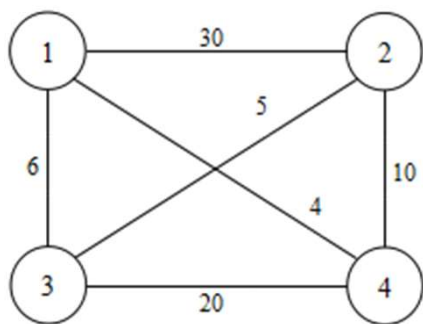
P类与NP类问题

■ NP(non-deterministic Polynomial time)

□ 最大团问题



□ 旅行售货员问题



P=NP: 是否所有NP类问题都可以在多项式时间内解决并验证，也就是转化为P类问题



8.3 NP完全问题

- 8.3.1 多项式时间变换
- 8.3.2 Cook定理



8.3.1 多项式时间变换

设 X 和 Y 是两个分别定义在实例集 I 和 J 上的判定问题。所谓问题 X 能在**多项式时间内变换**为问题 Y (简记为 $X \propto_p Y$)是指存在映射 $f: I \rightarrow J$, 且 f 满足:

- (1) f 是在多项式时间内可计算的函数;
- (2)对于所有 $x \in I$, $x \in X$, 当且仅当 $f(x) \in Y$ 。

定义: 问题 L 是**NP完全**的当且仅当

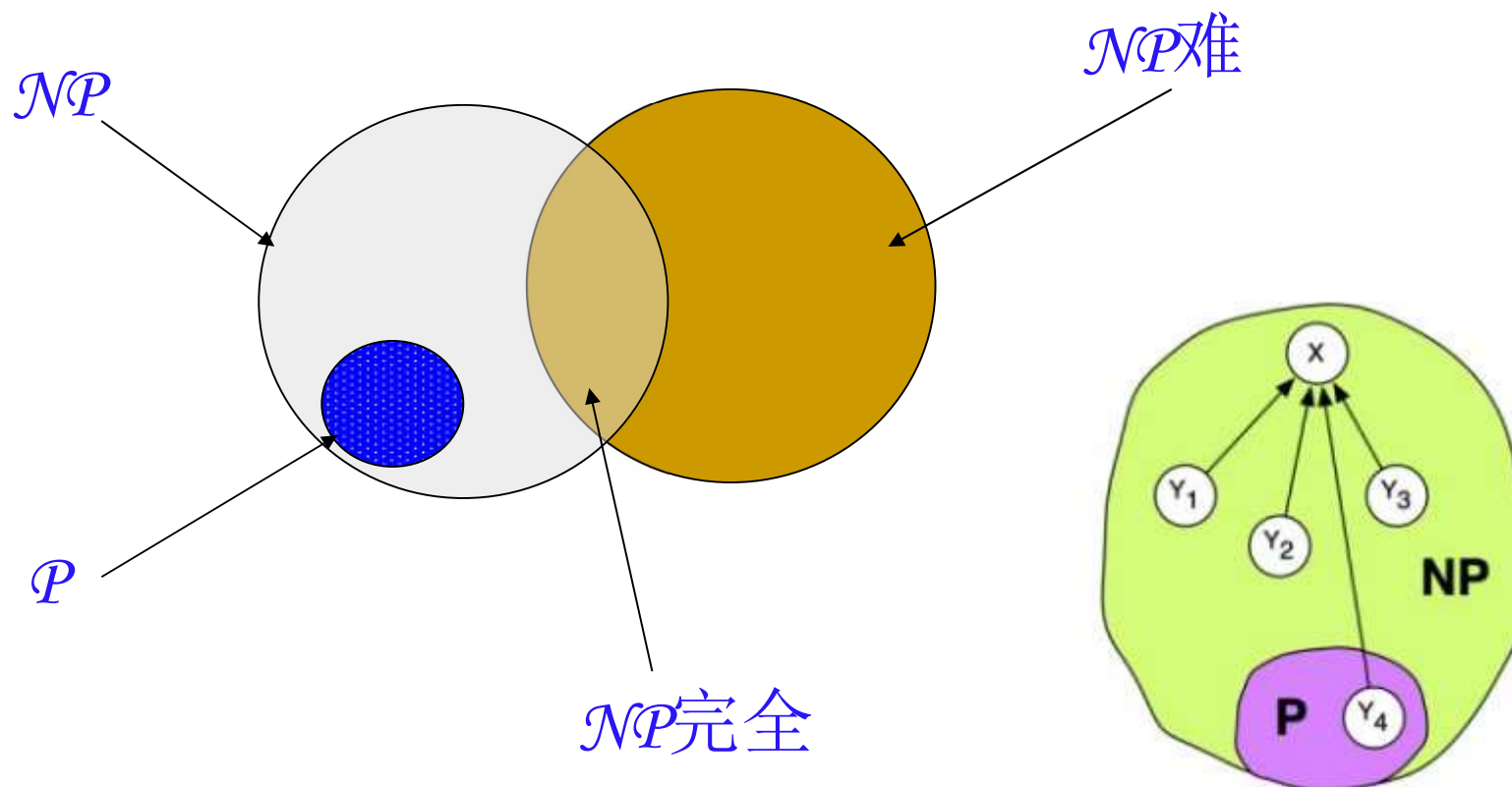
- (1) $L \in NP$;
- (2)对于所有 $L' \in NP$ 有 $L' \propto_p L$ 。

如果有一个问题 L 满足上述性质(2), 但不一定满足性质(1), 则称该问题是**NP难**的。所有NP完全问题构成的问题类称为**NP完全问题类**, 记为**NPC**。



NP 难和 NP 完全类

- P 、 NP 、 NP 完全和 NP 难问题的关系





8.3.1 多项式时间变换

定理8-2的(1)可知：如果任一NPC问题可在P时间内求解，则所有NPC问题可在P时间内求解。反之，若 $P \neq NP$ ，则所有NPC问题不可在P时间内求解。

定理8-2： 设L是NP完全的，则

(1) $L \in P$ 当且仅当 $P = NP$;

(2) 若 $L \propto_p L_1$ ，且 $L_1 \in NP$ ，则 L_1 是NP完全的。

定理8-2的(2)可用来证明问题的NP完全性。但前提是：要有第一个NP完全问题L。



8.3.2 Cook定理

定理8-7(Cook定理): 布尔表达式的可满足性问题SAT是NP完全的。

证明: SAT的一个实例是 k 个布尔变量 x_1, \dots, x_k 的 m 个布尔表达式 A_1, \dots, A_m 。若存在各布尔变量 ($1 \leq i \leq k$) 的0, 1赋值, 使每个布尔表达式 ($1 \leq i \leq m$) 都取值1, 则称布尔表达式 $\dots A_1 A_2 \dots A_m$ 是可满足的。

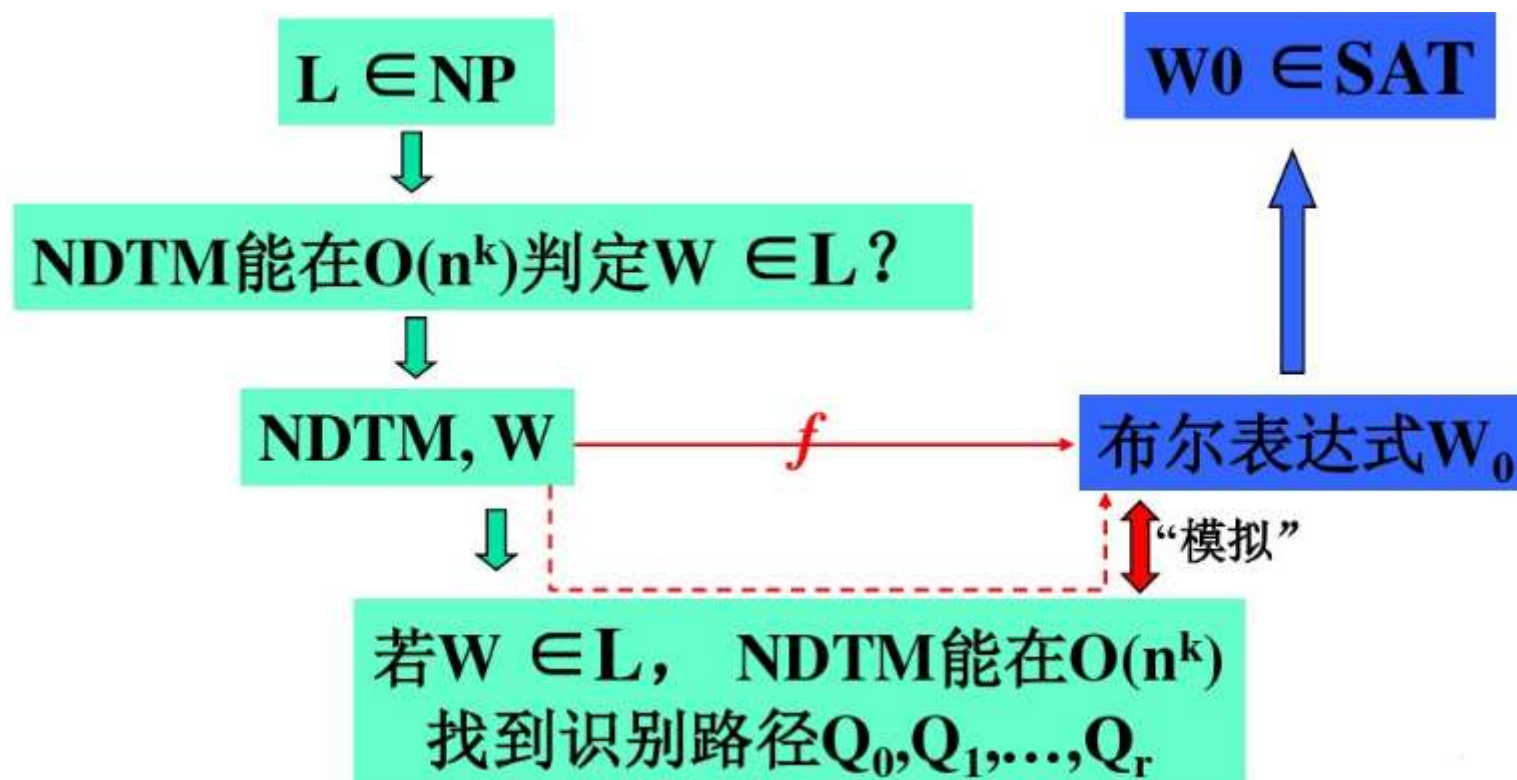
SAT \in NP是很明显的。对于任给的布尔变量 x_1, \dots, x_k 的0, 1赋值, 容易在多项式时间内验证相应的 $A_1 A_2 \dots A_m$ 的取值是否均为1。因此, SAT \in NP。

现在只要证明对任意的 $L \in \text{NP}$ 有 $L \leq_p \text{SAT}$ 即可。
(详细证明见书本P307~310)



Cook定理

■ 证明思路





Cook定理

- $L \in NP$, 设 M 是一台能在多项式时间内识别 L 的非确定性图灵机, W 是对 M 的一个输入
- 由 M 和 W 能在多项式时间内构造一个布尔表达式 W_0
- 模拟由 M 接受 W 的所有瞬像序列
- 使得 W_0 是可满足的当且仅当 M 接受 W



Cook 定理

- 设 M 为一台单带NDTM
- s 个状态 q_0, \dots, q_{s-1}
- m 个带符号 x_1, \dots, x_m
- 多项式时间 $p(n)$ 内 M 识别 L
- 设 W 是 M 的一个长度为 n 的输入，若 M 接受 W ，只需不超过 $p(n)$ 次移动
- 对于 W ，存在 M 的一个瞬像序列 Q_0, \dots, Q_r, Q_0 为初始瞬像， Q_r 为接受瞬像， $r \leq p(n)$



Cook定理

判断 $Q_0, Q_1, \dots, Q_{P(n)}$ 为一条接受 W 的计算路径等价于判断下述 7 条事实：

- (1) 在每一瞬象中读写头恰只扫描一个方格；
- (2) 在每一瞬象中，每个方格中的带符号是唯一确定的；
- (3) 在每一瞬象中恰有一个状态；
- (4) 在该计算路径中，从一个瞬象到下一个瞬象每次最多有一个方格（被读写头扫描着的那个方格）的符号被修改；
- (5) 相继的瞬象之间是根据移动函数 δ 来改变状态，读写头位置和方格中符号的；
- (6) Q_0 是 M 在输入 W 时的初始瞬象；
- (7) 最后一个瞬象 $Q_{P(n)}$ 中的状态是接受状态。

为了确切地表达上述 7 条事实，需要引进和使用以下几种命题变量：

- (1) $C\langle i, j, t \rangle = 1$, 当且仅当在时刻 t , M 的输入带的第 i 个方格中的带符号为 X_j , 其中, $1 \leq i \leq P(n), 1 \leq j \leq m, 0 \leq t \leq P(n)$ 。
- (2) $S\langle k, t \rangle = 1$, 当且仅当在时刻 t , M 的状态为 q_k , 其中, $1 \leq k \leq s, 0 \leq t \leq P(n)$ 。
- (3) $H\langle i, t \rangle = 1$, 当且仅当在时刻 t , 读写头扫描第 i 个方格, 其中, $1 \leq i \leq P(n), 0 \leq t \leq P(n)$ 。



Cook定理

(1) A 断言在 M 的每一个时间单位中,读写头恰好扫描着一个方格。设 A_t 表示在时刻 t 时 M 的读写头恰好扫描着一个方格,则

$$A = A_0 A_1 \cdots A_{P(n)}$$

其中,

$$A_t = U(H\langle 1, t \rangle, H\langle 2, t \rangle, \dots, H\langle P(n), t \rangle), 0 \leq t \leq P(n)$$

(2) B 断言在每一个单位时间内,每一个带方格中只有一个带符号。设 B_{it} 表示在时 t , 第 i 个方格中只含有一个带符号,则

$$B = \prod_{0 \leq i, t \leq P(n)} B_{it}$$

其中,

$$B_{it} = U(C\langle i, 1, t \rangle, C\langle i, 2, t \rangle, \dots, C\langle i, m, t \rangle), 0 \leq i, t \leq P(n)$$



Cook定理

(3) C 断言在每个时刻 t , M 只有一个确定的状态, 则

$$C = \prod_{0 \leq t \leq P(n)} U(S\langle 0, t \rangle, S\langle 1, t \rangle, \dots, S\langle s-1, t \rangle)$$

(4) D 断言在时刻 t 最多只有一个方格的内容被修改, 则

$$D = \prod_{i, j, t} (c\langle i, j, t \rangle \equiv c\langle i, j, t+1 \rangle + H\langle i, t \rangle)$$

(5) E 断言根据 M 的移动函数 δ , 可以从一个瞬象转向下一个瞬象。设 E_{ijkt} 表示下列 4 种情形之一:

- ① 在时刻 t 第 i 个方格中的符号不是 X_j ;
- ② 在时刻 t 读写头没有扫描着方格 i ;
- ③ 在时刻 t , M 的状态不是 q_k ;
- ④ M 的下一瞬象是根据移动函数从上一瞬象得到的。

由此可得 $E = \prod_{i, j, k, t} E_{ijkt}$ 。其中,

$$\begin{aligned} E_{ijkt} = & \neg C\langle i, j, t \rangle + \neg H\langle i, t \rangle + \neg S\langle k, t \rangle \\ & + \sum_l (C\langle i, j_l, t+1 \rangle S\langle k_l, t+1 \rangle H\langle i_l, t+1 \rangle) \end{aligned}$$



Cook定理

(6) F 断言满足初始条件,即

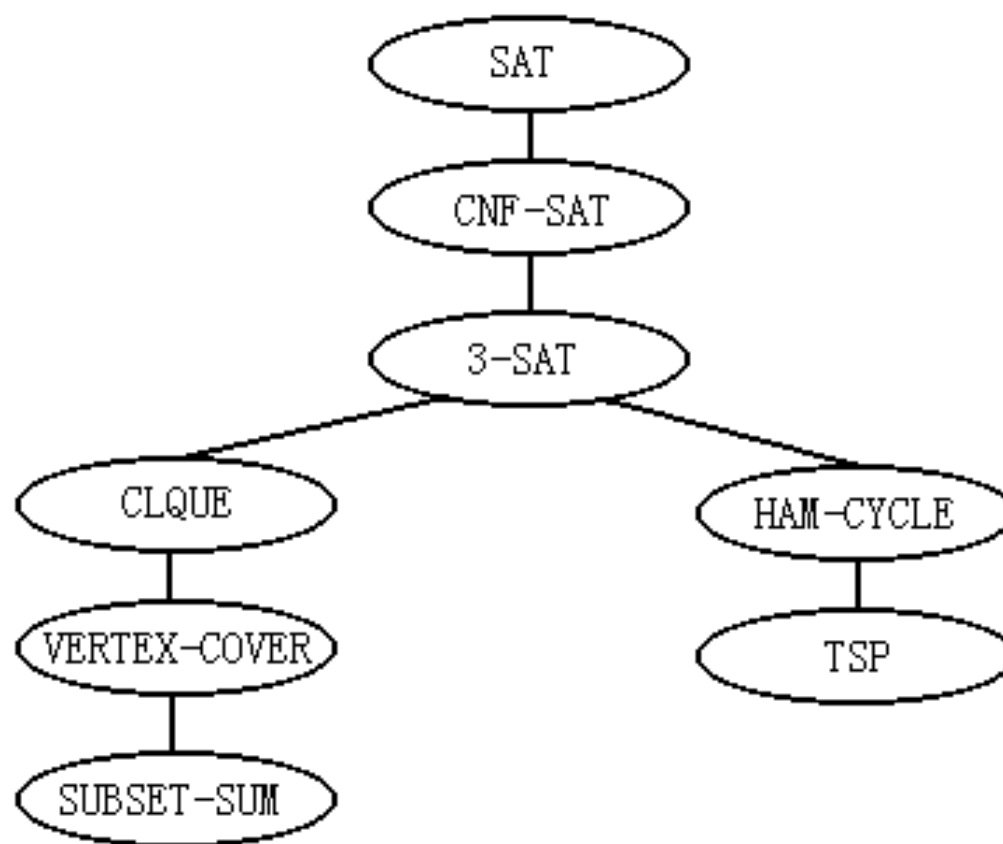
$$F = S\langle 1, 0 \rangle H\langle 1, 0 \rangle \prod_{1 \leq i \leq n} C\langle i, j_i, 0 \rangle \prod_{n < i \leq P(n)} C\langle i, 1, 0 \rangle$$

(7) G 断言 M 最终将进入接受状态。因为已对 M 做了修改,一旦 M 在某个时刻 t 进入接受状态 ($1 \leq t \leq P(n)$),它将始终停在这个状态,所以有 $G = S\langle s-1, P(n) \rangle$ 。不妨取 q_{s-1} 为 M 的接受状态。

$$W_0 = \text{ABCDEFGG}$$



8.4 一些典型的NP完全问题



部分NP完全问题树



6.4.1 合取范式的可满足性问题 (CNF-SAT)

问题描述： 给定一个合取范式 α ，判定它是否可满足。

如果一个布尔表达式是一些因子和之积，则称之为合取范式，简称CNF(Conjunctive Normal Form)。这里的因子是变量 x 或 \bar{x} 。例如： $(x_1 + x_2)(x_2 + x_3)(\bar{x}_1 + \bar{x}_2 + x_3)$ 就是一个合取范式，而 $x_1x_2 + x_3$ 就不是合取范式。

我们已经知道， $\text{SAT} \in \text{NP}$ 。因此，要证明 $\text{CNF-SAT} \in \text{NPC}$ ，只需证明：

- (1) $\text{CNF-SAT} \in \text{NP}$
- (2) $\text{SAT} \propto_p \text{CNF-SAT}$



CNF-SAT问题

CNF-SAT \in NP 是显然的。因为合取范式是布尔表达式的一种特殊形式，由于 SAT \in NP，所以 CNF-SAT \in NP。

为了证明 SAT \propto_p CNF-SAT，需要将一个布尔表达式满足性问题在多项式时间内转化为一个合取范式满足性问题。通过下列两个步骤来完成。

第一步，利用等价公式把所有的 \neg 移到紧靠变元的位置，使布尔表达式变换为文字的逻辑与和逻辑或的形式，这种形式我们称之为与或形。

第二步，将第一步所得到的与或形公式转化为合取范式形式。转化方法如下。



CNF-SAT问题

设 E 是一个与或形公式，如果 E 中只包含1个或2个文字，则 E 已经是一个合取范式。否则， E 可以表示为 $E_1 \wedge E_2$ 或 $E_1 \vee E_2$ 的形式，可采用递归方式先将 E_1 和 E_2 分别转化为可满足性与之等价的合取范式 F_1 和 F_2 ，再用 F_1 和 F_2 构造 E 的合取范式 F 。分两种情况处理。

情况1：若 $E = E_1 \wedge E_2$ ，则 $F = F_1 \wedge F_2$ 。显然 F 是一个合取范式，并且 F 的可满足性与 E 等价。

情况2：若 $E = E_1 \vee E_2$ ，不妨设由 E_1 和 E_2 转化所得到的合取范式分别为 $F_1 = c_1 \wedge c_2 \wedge \dots \wedge c_k$ 和 $F_2 = d_1 \wedge d_2 \wedge \dots \wedge d_m$ 。其中， c_i 和 d_i 都是子句。引入新变元 u ，构造合取范式如下：

$$F = (u + c_1)(u + c_2) \dots (u + c_k)(\bar{u} + d_1)(\bar{u} + d_2) \dots (\bar{u} + d_m)$$



CNF-SAT问题

可以证明，E和F的可满足性是有等价的。

设与或形表达式E中包含的 \wedge 和 \vee 运算的次数为n，那么由上述变换所得到的合取范式F中最多有n+1个子句。因此，把E分裂成E1和E2，以及由F1和F2构造F，每部分都是n的线性函数。设bn是把E分裂成E1和E2加上由F1和F2构造F所需时间总和的上界，于是可以得出由E构造F所需时间T(n)的递推方程：

$$T(n) \leq \begin{cases} a & n \leq 1 \\ bn + \max_{0 \leq i \leq n-1} \{T(i) + T(n-i-1)\} & n > 1 \end{cases}$$

运用归纳法可以证明 $T(n)=O(n^2)$



8.4.2 3元合取范式的可满足性问题 (3-SAT)

问题描述： 给定一个3元合取范式 α ，判定它是否可满足。

在一个合取范式中，如果每个子句最多含有 k 个文字，则称这个合取范式为 k 元合取范式，简记为 k -CNF。一个 k -SAT问题是判定一个 k -CNF是否可满足。当 $k=3$ 时，即为3-SAT问题。

证明思路：

3-SAT \in NP是显而易见的。为了证明3-SAT \in NPC，只要证明CNF-SAT \propto_p 3-SAT，即合取范式的可满足性问题可在多项式时间内变换为3-SAT。



3-SAT问题

现证明 $\text{CNF-SAT} \propto_p \text{3-SAT}$ 。解决该问题的关键是如何将合取范式中每个子句转化成可满足性与其等价的3元合取范式。

设一个长度为 k 的子句, $L = x_1 + x_2 + \dots + x_k$

现在来构造一个可满足性与 L 等价的3元合取范式 F , 即当且仅当 F 可满足时, L 才是可满足的。

如果 $k < 4$, L 就是一个3元合取范式。

如果 $k \geq 4$, 先利用子句 L 构造一个合取范式:

$$F = (x_1 + x_2 + y_1)(\bar{y}_1 + x_3 + \dots + x_k)$$

其中 y_1 为新增变量。 F 与 L 的可满足性是等价的。



8.4.3 团问题CLIQUE

问题描述： 给定一个无向图 $G=(V, E)$ 和一个正整数 k ，判定图 G 是否包含一个 k 团，即是否存在， $V' \subseteq V$ ， $|V'|=k$ ，且对任意 $u, w \in V'$ 有 $(u, w) \in E$ 。

证明思路：

(1) 已经知道 $\text{CLIQUE} \in \text{NP}$ 。因为验证图 G 的一个子图是否构成团只需要多项式时间，所以， $\text{CLIQUE} \in \text{NP}$ 。

(2) 通过 $3\text{-SAT} \propto_p \text{CLIQUE}$ 来证明 CLIQUE 是NP难的，从而证明团问题是NP完全的。



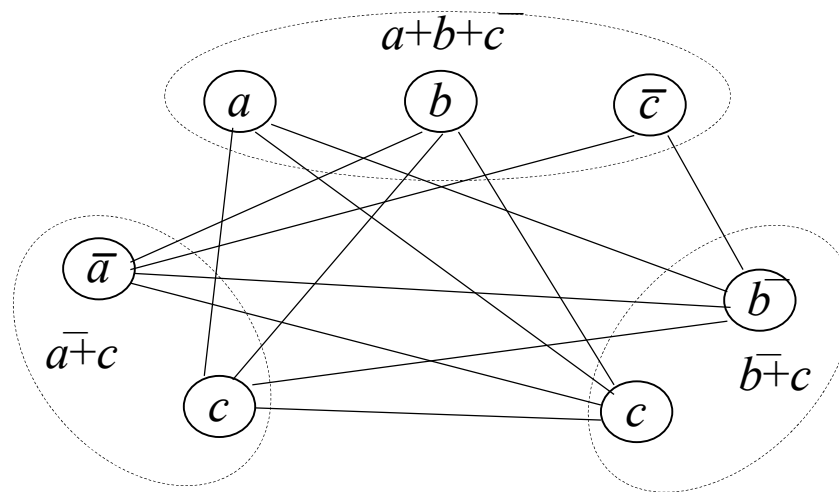
团问题CLIQUE

对于任意一个3元合取范式F，按照如下方法构造相应的图G。

1) 图G的每个顶点对应F中的每个文字，对于多次出现在不同子句中的文字用不同的顶点重复表示；

2) 若图G中两个顶点对应的文字不互补且不出现在同一子句中，则将其连线。例如，合取范式 $(a+b+c)(\bar{a}+c)(\bar{b}+c)$ 对应的图如图所示。

设合取范式f有n个子句，则f可满足当且仅当f对应的图G中有n个顶点的团。





8.4.4 顶点覆盖问题 (VERTEX-COVER)

问题描述: 给定一个无向图 $G=(V, E)$ 和一个正整数 k , 判定是否存在 $V' \subseteq V$, $|V'|=k$, 使得对于任意 $(u, v) \in E$ 有 $u \in V'$ 或 $v \in V'$ 。如果存在这样的 V' , 就称 V' 为图 G 的一个大小为 k 顶点覆盖。

证明思路:

首先, $\text{VERTEX-COVER} \in \text{NP}$ 。因为对于给定的图 G 和正整数 k 以及一个顶点集 V' , 验证 $|V'|=k$, 然后对每条边 (u,v) 检查是否有 $u \in V'$ 或 $v \in V'$, 显然可在多项式时间内完成。

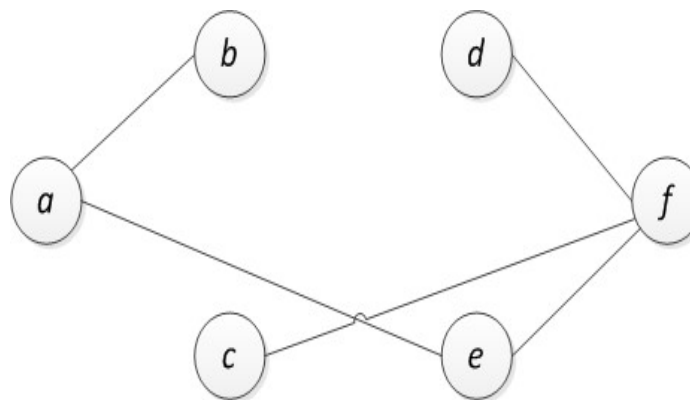
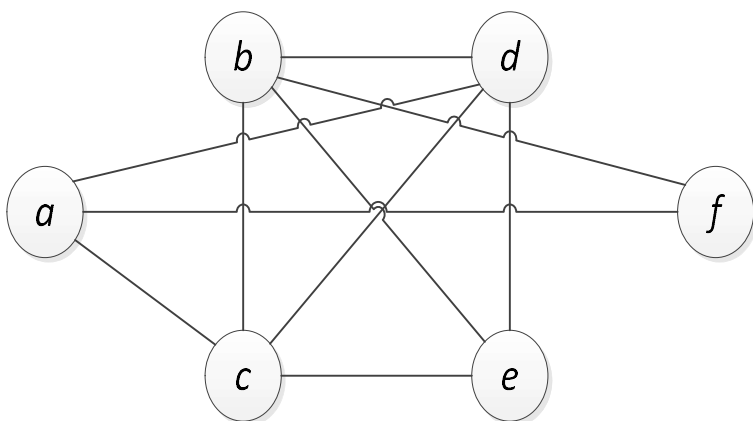
其次, 通过 $\text{CLIQUE} \propto_p \text{VERTEX-COVER}$ 来证明顶点覆盖问题是NP难的。



VERTEX-COVER问题

给定无向图 $G=(V, E)$ ，构造 G 的补图 $G'(V, E')$ 。其中：

$$E' = \{(u, v) | (u, v) \notin E\}$$



根据图 G 与其补图 G' 之间的关系，可以证明图 G 中有一个大小为 k 的团当且仅当 G' 中有一个大小为 $|V|-k$ 的顶点覆盖。



8.4.5 子集和问题 (SUBSET-SUM)

问题描述： 给定整数集合 S 和一个整数 t ，判定是否存在 S 的一个子集 $S' \subseteq S$ ，使得 S' 中整数的和为 t 。

例如，若 $S = \{1, 4, 16, 64, 256, 1040, 1041, 1093, 1284, 1344\}$ 且 $t = 3754$ ，则子集 $S' = \{1, 16, 64, 256, 1040, 1093, 1284\}$ 是一个解。

证明思路：

首先，对于子集和问题的一个实例 $\langle S, t \rangle$ ，给定一个“证书” S' ，要验证 $t = \sum_{i \in S'} i$ 是否成立，显然可在多项式时间内完成。因此， $\text{SUBSET-SUM} \in \text{NP}$ ；

其次，证明 $\text{VERTEX-COVER} \propto_p \text{SUBSET-SUM}$ 。

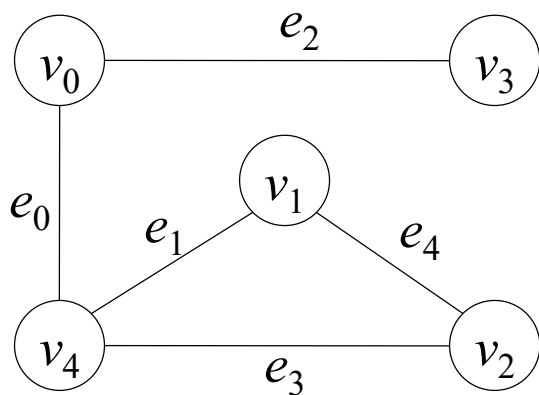


SUBSET-SUM问题

给定一个图的顶点覆盖问题的实例 $\langle G, k \rangle$ ，在多项式时间内将其变换为子集和问题的一个实例 $\langle S, t \rangle$ ，使得 G 有一个 k 团当且仅当 S 有一个子集 S' ，其元素和为 t 。

变换要用到图 G 的关联矩阵 B 。定义 $B = \{b_{ij}\}$ 如下：

$$b_{ij} = \begin{cases} 1 & \text{顶点 } v_i \text{ 与边 } e_j \text{ 相关联} \\ 0 & \text{其他} \end{cases}$$



	e_4	e_3	e_2	e_1	e_0
v_0	0	0	1	0	1
v_1	1	0	0	1	0
v_2	1	1	0	0	0
v_3	0	0	1	0	0
v_4	0	1	0	1	1

关联矩阵 B



SUBSET-SUM问题

构造实例 $\langle S, t \rangle$ 的过程如下:

对于每个顶点 v_i , 构造整数 x_i :

$$x_i = 4^{|E|} + \sum_{j=0}^{|E|-1} b_{ij} 4^j$$

对于每条边 e_j , 构造整数 y_j :

$$y_i = 4^j$$

则 $S = \{x_0, x_1 \cdots x_{|V|-1}, y_0, y_1 \cdots y_{|E|-1}\}$

构造整数 t : $t = k4^{|E|} + \sum_{j=0}^{|E|-1} 2 \cdot 4^j$

		e_4	e_3	e_2	e_1	e_0	
x_0	=	1	0	0	1	0	= 1041
x_1	=	1	1	0	0	1	= 1284
x_2	=	1	1	1	0	0	= 1344
x_3	=	1	0	0	1	0	= 1040
x_4	=	1	0	1	0	1	= 1093
y_0	=	0	0	0	0	1	= 1
y_1	=	0	0	0	1	0	= 4
y_2	=	0	0	1	0	0	= 16
y_3	=	0	1	0	0	0	= 64
y_4	=	0	1	0	0	0	= 256
t	=	3	2	2	2	2	= 3754

可以发现, 当 G 有一个 k 团当且仅当 S 有一个子集 S' , 其元素和为 t 。例如:

顶点覆盖 $\{v_0, v_1, v_2\}$ 对应: $t = x_0 + x_1 + x_2 + y_0 + y_1 + y_2 + y_3$

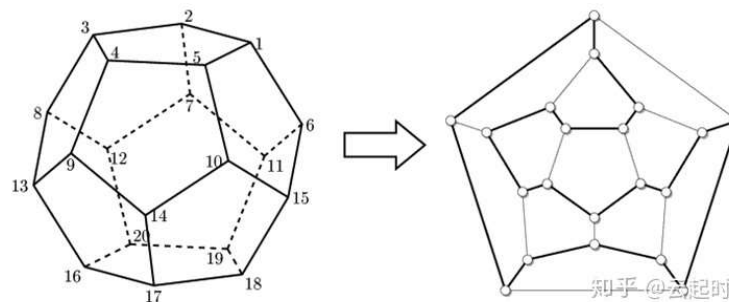
$t = x_0 + x_2 + x_4 + y_1 + y_2 + y_4$, 对应顶点覆盖 $\{v_0, v_2, v_4\}$



6.4.6 哈密顿回路问题 (HAM-CYCLE)

问题描述： 给定无向图 $G=(V,E)$ ，判定其是否含有一哈密顿回路。

证明思路：



首先，对于HAM-CYCLE问题的一个实例，给定所有顶点的一个排列，要验证这个序列是否是一个回路，显然可在多项式时间内完成。因此， $\text{HAM-CYCLE} \in \text{NP}$ ；

其次，证明 $\text{VERTEX-COVER} \propto_p \text{HAM-CYCLE}$ 。



HAM-CYCLE问题

对于无向图 $G = \langle V, E \rangle$ ，给定 G 的顶点覆盖实例 $\langle G, k \rangle$ 。
设顶点 $v \in V$ 的度为 d_v ，与 v 相关联的边记为： $e_1^v, e_2^v, \dots, e_{d_v}^v$

现构造一个有向图边 $G' = \langle V', E' \rangle$ ，过程如下：

顶点集合 V' 由两部分组成：

- (1) 新增 k 个顶点： s_1, s_2, \dots, s_k
- (2) 对于每个顶点 $v \in V$ ，对应 $2d_v$ 个顶点：

$$v_1^{(I)}, v_1^{(O)}, v_2^{(I)}, v_2^{(O)}, \dots, v_{d_v}^{(I)}, v_{d_v}^{(O)}$$



HAM-CYCLE问题

边集合 E' 的组成部分有:

(1) 对于每个顶点 $v \in V$, 对应有向边:

$$(v_1^{(I)}, v_1^{(O)}), (v_1^{(O)}, v_2^{(I)}), (v_2^{(I)}, v_2^{(O)}), \dots (v_{dv}^{(I)}, v_{dv}^{(O)})$$

(2) 对于每条边 $(u, v) \in E$ (假定 $e_i^u = e_j^v$), 对应有向边:

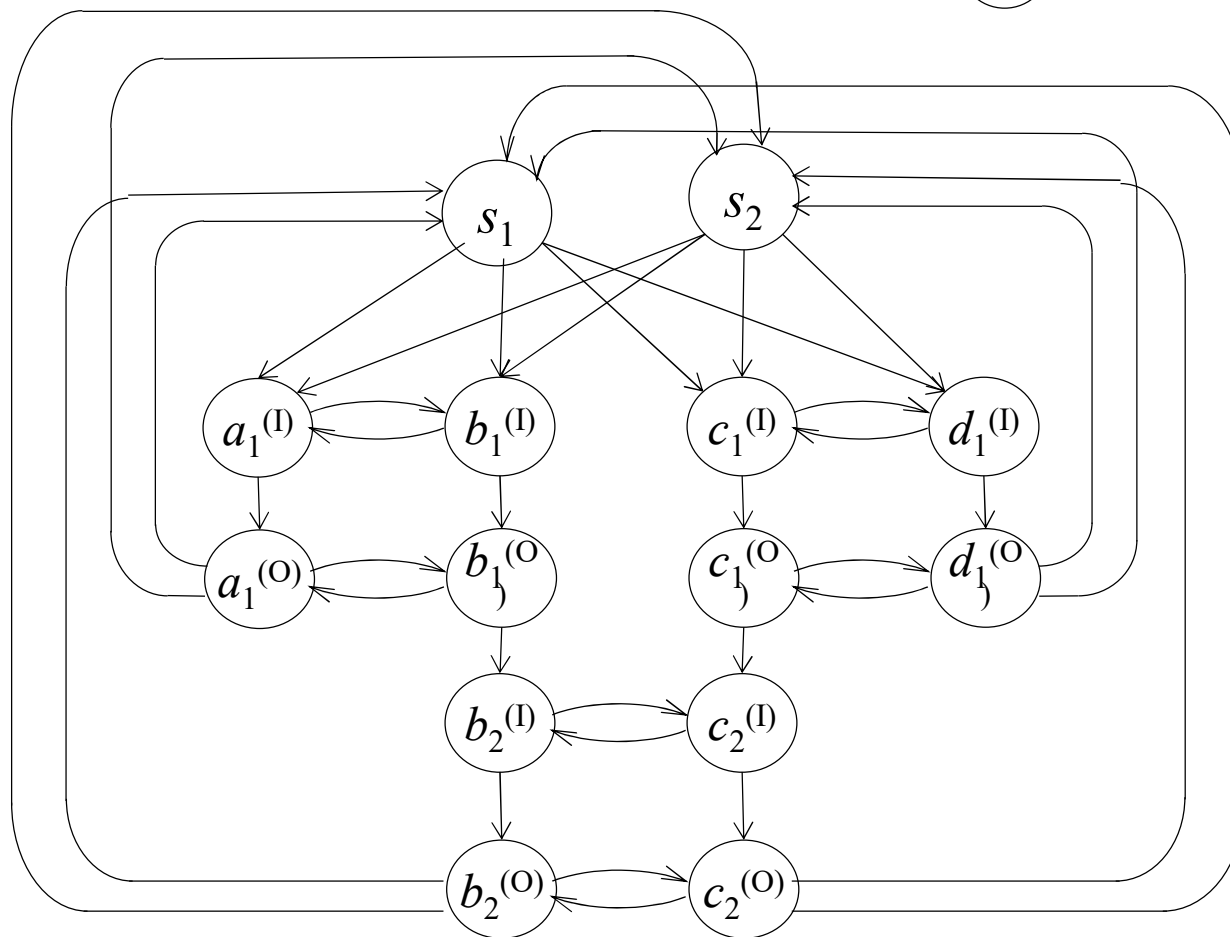
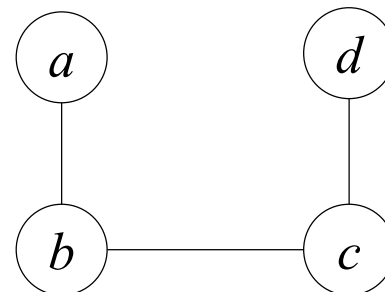
$$(u_i^{(I)}, v_j^{(I)}), (v_j^{(I)}, u_i^{(I)}), (u_i^{(O)}, v_j^{(O)}), (v_j^{(O)}, u_i^{(O)})$$

(3) 对于 s_i 与每个顶点 $v \in V$, 添加有向边:

$$(s_i, v_1^{(I)}), (v_{dv}^{(O)}, s_i) \quad (1 \leq i \leq k)$$



例如，对于右图所示的
无向图 G ，构造 G' 如下：





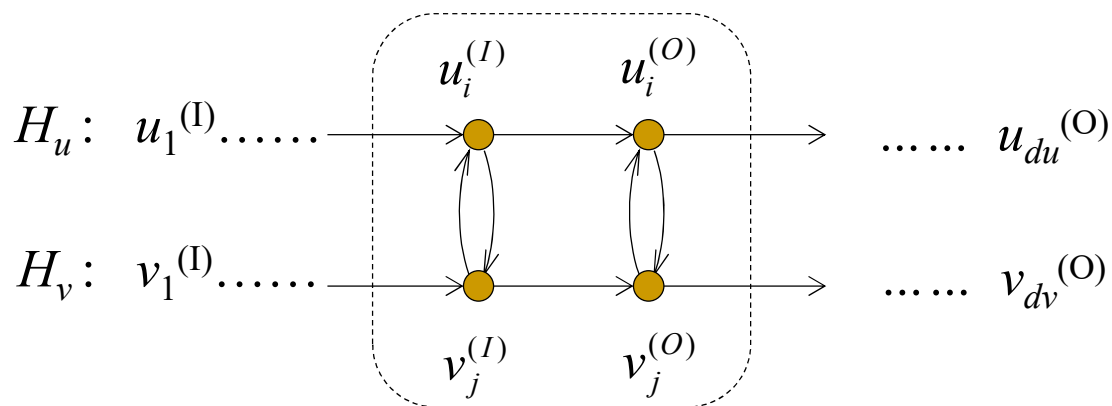
HAM-CYCLE问题

对于G中的每个顶点 $v \in V$, 在G'中对应有一个通路

$$H_v: v_1^{(I)} \rightarrow v_1^{(O)} \rightarrow v_2^{(I)} \rightarrow v_2^{(O)} \rightarrow \dots \rightarrow v_{dv}^{(I)} \rightarrow v_{dv}^{(O)}$$

对于G中的每条边 $(u, v) \in E$, 通路 H_u 和 H_v 之间存在边:

$$(u_i^{(I)}, v_j^{(I)}), (v_j^{(I)}, u_i^{(I)}), (u_i^{(O)}, v_j^{(O)}), (v_j^{(O)}, u_i^{(O)})$$

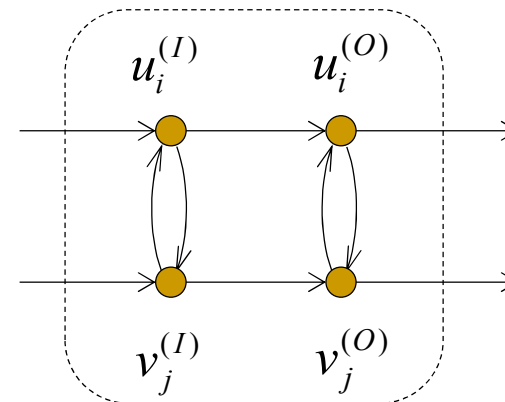




HAM-CYCLE问题

这个子图有下列特点：

- (1) 每个顶点的入度和出度都为2；
- (2) 若有哈密尔顿回路由 $u_i^{(I)}$ 进入，则必须从 $u_i^{(O)}$ 退出该子图。若从 $v_j^{(O)}$ 退出，则 $u_i^{(O)}$ 或 $v_j^{(I)}$ 将始终无法被经过。



因此，从 $u_1^{(I)}$ 进入的哈密尔顿回必须从 $u_{du}^{(O)}$ 退出，即必定经过 H_u 通路中的所有顶点。

接下来，我们证明 G' 存在哈密尔顿回路的充要条件是图 G 有 k 个顶点的覆盖。



HAM-CYCLE问题

设图G有一个k个顶点的覆盖 $C=\{v_1, v_2, \dots, v_k\}$ ，可根据C构造G'的一条哈密尔顿回路。步骤如下：

(1) 先构造一条回路P

$$s_1 \rightarrow H_{v_1} \rightarrow s_2 \rightarrow H_{v_2} \rightarrow \dots \rightarrow s_k \rightarrow H_{v_k} \rightarrow s_1$$

(2) 对于任意一条边 $(u, v)=e_i^u=e_j^v$ ， u 和 v 中至少有一个是在C中。如果 u 和 v 都在C中，则边 (u, v) 在G'中对应的4个顶点已经包含在回路P中。不妨设 u 在C中而 v 不在，那么在回路P中的 H_u 可以由 $u_i^{(1)} \rightarrow v_i^{(1)} \rightarrow v_i^{(0)} \rightarrow u_i^{(0)}$ 绕道经过 $v_i^{(1)}$ 和 $v_i^{(0)}$ ，从而将 $v_i^{(1)}$ 和 $v_i^{(0)}$ 加入到回路P中。由于C是顶点覆盖，G'中的所有顶点都可添加到P中。

反之，也可从G'的哈密尔顿回路构造图G的定点覆盖。



6.4.7 旅行售货员问题TSP

问题描述： 给定一个无向完全图 $G=(V,E)$ 及定义在 $V \times V$ 上的一个费用函数 c 和一个整数 k ，判定 G 是否存在经过 V 中各顶点恰好一次的回路，使得该回路的费用不超过 k 。

TSP: n 个城市, 解空间规模 $n!$

$N = 30, N! = 2.6525 \times 10^{32}$



峰值计算速度: 5.496亿亿/秒

$2.6525 \times 10^{32} / (5.496 \times 10^{16} / 3600 \times 24 \times 365) = 1.5304 \times 10^8 \text{年}$

1.5亿年

知乎 @



TSP问题

首先，给定TSP的一个实例 (G, c, k) ，和一个由 n 个顶点组成的顶点序列。验证算法要验证这 n 个顶点组成的序列是图 G 的一条回路，且经过每个顶点一次。另外，将每条边的费用加起来，并验证所得的和不超过 k 。这个过程显然可在多项式时间内完成，即 $TSP \in NP$ 。

其次，旅行售货员问题与哈密顿回路问题有着密切的联系。哈密顿回路问题可在多项式时间内变换为旅行售货员问题。



TSP问题

设图 $G=(V,E)$ 是HAM-CYCLE问题的一个实例，构造一个完全图 $G'=(V,E')$ ，且定义费用函数 c 为：

$$c(i, j) = \begin{cases} 0 & (i, j) \in E \\ 1 & (i, j) \notin E \end{cases}$$

则相应的TSP的实例为 $G'=(V,c,0)$ 。这个构造过程可以在 $\Theta(n^2)$ 时间内完成。

即 $\text{HAM-CYCLE} \propto_p \text{TSP}$ 。从而，旅行售货员问题是NP难的。因此， $\text{TSP} \in \text{NPC}$ 。