第11章　*NP*完全性理论*

首先介绍*P*类问题、*NP*类问题和*NP*完全问题的相关概念，列举一些典型的*NP*问题和*NP*完全问题，然后通过实例说明*NP*完全问题的证明方法。

- 非确定性算法与*NP*类问题
- 一些典型的*NP*问题
- 问题变换与*NP*完全类问题
- 一些典型的*NP*完全问题
- *NP*完全问题的证明方法

11.1　非确定性算法与*NP*问题

11.1.1　可计算性问题

1. 可计算性问题的层次

若某问题存在求解的算法，则称其为可计算的；若不存在算法但是存在求解的过程，则称其为半可计算的；若连求解的过程也不存在，则称其为完全不可计算的。

【问题】可计算问题的计算复杂性是不是都相同呢？

2. 可计算问题的分类

按照最好求解算法的计算时间分类。

① 可以用多项式时间算法求解的问题。例如

- 以比较为基础的搜索问题—— $O(\log n)$
- 以比较为基础的排序问题—— $O(n \log n)$

② 没有找到多项式时间求解算法的问题。例如

- 旅行商问题—— $O(n^2\ 2^n)$
- 0/1背包问题—— $O(2^{n/2})$

这里给出的是当前最好的确定性算法的时间复杂度。

3. 判定问题

判定问题是答案只能是 true 或 false 的问题。大多数问题都可以转化为判定问题（例如，图的最大团问题可以转化为判定图中是否存在不小于 k 的团）。如果知道怎样解决该判定问题，通常就可以解决原问题，比如通过二分查找（下一小节给出了一个例子）。本章主要考虑判定问题。

4. 其他问题转换为判定问题的一个例子*

（1）判定问题。判断图G是否有m团。

```
#include "Clique.h"
#include "algorithm.h"
```

```
bool Clique(Matrix<bool> &G, int m, vector<bool> &MX) // 用MX记录m团
{   int n = G.Rows();
    vector<bool> X(n); // 当前团
    bool success = false; // 是否有m团
```

```
    function<void(int, int)> Clique = [&](int t, int cn)
    {   if(t >= n && cn >= m) MX = X, success = true; // 答案
        else if(t < n)
        {   if(Connected(G, X, t)) // 检查可行，生成左儿子
                X[t] = 1, Clique(t + 1, cn + 1);
            if(cn + n - t > m) // 检查界限，生成右儿子
                X[t] = 0, Clique(t + 1, cn);
        }
    };
```

```
        Clique(0, 0);
        return success;
    }
```

（2）最优化问题。寻找图G的最大团。

```
auto Clique(Matrix<bool> &G)
{   int n = G.Rows();
    vector<bool> X(n), BX; // 当前解, 最优解
    int low = 1, high = n; // 用二分方法寻找最优解
    while(low <= high)
    {   int m = (low + high) / 2;
        if(Clique(G, m, X)) BX = X, low = m + 1;
        else high = m - 1;
    }
    return BX;
}
```

（3）测试程序。使用图11-1所示的图。

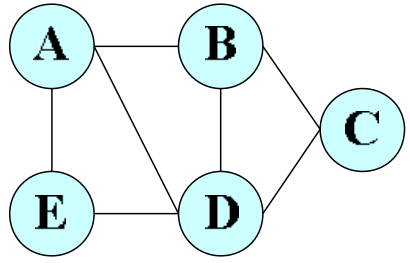


图11-1　一个最大团实例

```
int main()
{   Matrix<bool> G = // 图的邻接矩阵
    {   {0, 1, 0, 1, 1},
        {1, 0, 1, 1, 0},
        {0, 1, 0, 1, 0},
        {1, 1, 1, 0, 1},
        {1, 0, 0, 1, 0}
    };
    cout << "最大团: " << Clique(G) << endl; // 最大团: 1 1 0 1 0
}
```

11.1.2 非确定性算法

前述章节主要介绍确定性算法（算法中每个操作的结果是唯一定义的）。算法一词通常表示确定性算法。

一般将可由多项式时间算法成功求解的问题看作是易处理的问题，而将需要超过多项式时间才能成功求解的问题看作是难处理的问题。

有许多问题，从表面上看似乎并不比排序等问题更困难，然而至今人们还没有找到解决这些问题的多项式时间算法，也没有人能够证明这些问题是否必需要超过多项式时间才能成功求解。也就是说，使用确定性算法，这类问题的计算复杂性至今未知。为了研究这类问题的计算复杂性，引入另一种能力更强的算法——非确定性算法。非确定性算法去除了唯一性限制。使用非确定性算法，许多问题可以在多项式时间内成功求解。

1. 三条指令

为了方便描述非确定性算法，引入下列三条指令，时间耗费均为 $O(1)$ 。

- Choice(S)。从集合 S 中任意选取一个元素，不关心如何选取。特别地，用Choice(m, n)从区间 $[m, n]$ 中任意选取一个整数。
- Failure()。发出不成功信号并Stop（不影响return）。
- Success()。发出成功信号并Stop（不影响return）。

2. 非确定性算法的2个阶段

① 用非确定性方法选出候选解 x

② 用确定性方法检查 x 是否是该问题的一个解

3. 说明

① 执行非确定性算法的机器称为非确定机，可以认为非确定机足够聪明，只要问题的解存在，它总可以用最短的时间成功获得问题的解。当且仅当问题的解不存在时，非确定机求解失败。

② 也可以认为一台非确定机由若干台（足够多）独立并行工作的确定性机器组成，第一台成功求解的确定性机器发出成功信号并结束其它确定性机器的计算过程。当且仅当问题的解不存在时，每台确定性机器都发出不成功信号。

③ 非确定机是一种假想的机器，现实中没有这样的机器。

④ 概率算法和非确定性算法并不等价。概率算法由确定性机器执行，非确定性算法由非确定性机器执行。概率算法可以借鉴非确定算法的思想（例如使用拉斯维加斯方法）。

11.1.3 *P*类与*NP*类问题

P 类问题：用确定性算法能在多项式时间内成功求解的问题。

NP 类问题：用非确定性算法能在多项式时间内成功求解的问题。

由于一个确定性算法可看作是某个非确定性算法的特例，所以可用确定性算法在多项式时间内成功求解的问题也可用非确定性算法在多项式时间内成功求解，故 $P \subseteq NP$ 。

11.2 一些典型的*NP*问题

下面给出了一些问题的多项式时间非确定性算法，所以这些问题都是 NP 的。

11.2.1 布尔表达式的可满足性问题SAT

1. 问题描述

由 n 个布尔变量 $x_1, x_2, \ldots, x_n$ 构成的布尔表达式 E 若存在各变量 $x_i (1 \le i \le n)$ 的0/1赋值，使得 $E = 1$，则称布尔表达式 E 是可满足的。

布尔表达式的可满足性问题：给定一个布尔表达式，判定它是否可满足。

2. 非确定性算法

```
bool SAT(function<bool(bool[])> E, bool X[], int n)
{   for(int i = 0; i < n; ++i)
        X[i] = Choice(0, 1);
    if(E(X) == 1) Success();
    else Failure();
}
```

3. 复杂性分析

设由 n 个布尔变量构成的布尔表达式的长度为 m（ $m \geq n$ ）。显然，非确定性地选取候选解耗时 $O(n)$，确定性地检查可满足性耗时 $O(m)$。所以整个算法的时间复杂性为 $O(m)$。

11.2.2 合取范式的可满足性问题CNF-SAT

如果一个布尔表达式是一些子句的合取，则称之为合取范式，简称CNF。其中，子句是一些文字的析取，文字是变量 x 或变量的否 $\overline{x}$。例如 $E_1=(x_1+x_2)(x_2+x_3)(\overline{x}_1+\overline{x}_2+x_3)$ 是一个合取范式，而 $E_2=x_1x_2+x_3$ 不是合取范式（是析取范式）。当 $x_1=x_3=0$ 而 $x_2=1$ 时，$E_1=1$，所以 E_1 是可满足的。

合取范式的可满足性问题：给定一个合取范式，判定它是否可满足。

11.2.3 三元合取范式的可满足性问题3-SAT

三元合取范式的每个子句恰好是3个文字的析取。例如，$E=(x_1+x_2+x_5)(x_3+x_4+\overline{x}_5)$ 是一个三元合取范式。当 $x_1=x_2=x_3=x_4=x_5=1$ 时，$E=1$，所以 E 是可满足的。

三元合取范式的可满足性问题：给定一个三元合取范式，判定它是否可满足。

11.2.4 团问题CLIQUE

1. 问题描述

给定无向图 G 和正整数 k，判定图 G 是否存在 k 团（或不小于 k 的团）。

2. 非确定性算法

```
bool Clique(const Matrix<bool> &G, int k, bool X[])
{   int n = G.Rows(), m = 0; // m为选取的顶点数
    for(int i = 0; i < n; ++i) //  非确定性地选取候选解
        X[i] = Choice(0, 1), m += X[i];
    if(m < k) Failure();
    for(int i = 0; i < n; ++i) //  确定性地检查候选解是否是团
        for(int j = 0; j < n; ++j)
            if(i != j && X[i] == 1 && X[j] == 1 && G[i][j] != 1) Failure();
    Success();
}
```

3. 复杂性分析

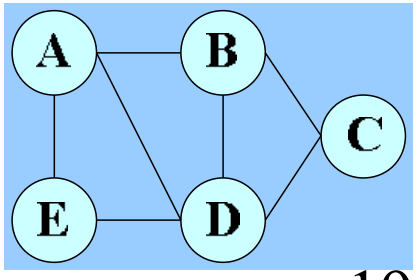
显然，非确定性地选取候选解耗时 $O(n)$，确定性地检查候选解是否是团耗时 $O(n^2)$。所以整个算法的时间复杂性为 $O(n^2)$。

11.2.5　顶点覆盖问题VERTEX-COVER

1. 问题描述

① 顶点覆盖。设V'是无向图G的一个顶点子集，如果G的每一条边都有一个端点在V'中，则称V'是G的一个顶点覆盖。顶点覆盖的顶点个数称为它的大小。大小为k的顶点覆盖称为k顶点覆盖，顶点数最少的顶点覆盖称为最小顶点覆盖。例如，对于图11-1所示的无向图，$\{A,B,D\}$、$\{A,C,D\}$和$\{B,D,E\}$都是它的最小顶点覆盖。

② 顶点覆盖问题。给定无向图G和正整数k，判定G是否存在k顶点覆盖（或不大于k的顶点覆盖）。

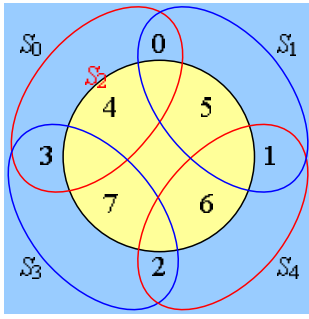


2. 非确定性算法

```
bool VertexCover(const Matrix<bool> &G, int k, bool X[])
{   int n = G.Rows(), m = 0; // m为选取的顶点数
    for(int i = 0; i < n; ++i) //  非确定性地选取候选解X
        X[i] = Choice(0, 1), m += X[i];
    if(m > k) Failure();
    for(int i = 0; i < n; ++i) //  确定性地检查X是否为顶点覆盖
        for(int j = 0; j < n; ++j)
            if(i != j && G[i][j] == 1 && X[i] != 1 && X[j] != 1) Failure();
    Success();
}
```

3. 复杂性分析

显然，非确定性地选取候选解耗时 $O(n)$，确定性地检查候选解是否为顶点覆盖耗时 $O(n^2)$。所以整个算法的时间复杂性为 $O(n^2)$。



11.2.6 集合覆盖问题SUBSET-COVER

1. 集合覆盖

设 X 是一个有限集合，F 是 X 的一个子集族。若 $X=\bigcup_{S\in F}S$，则称子集族 F 覆盖了 X，也称 F 是 X 的一个集合覆盖。F 中覆盖 X 的子集数最少的子集族 C^* 称为 $<X,F>$ 的最小集合覆盖。

例如，假设 $X=\{0,1,2,3,4,5,6,7\}$，$S_0=\{0,3,4\}$，$S_1=\{0,1,5\}$，$S_2=\{4,5,6,7\}$，$S_3=\{2,3,7\}$，$S_4=\{1,2,6\}$。容易看出，$S_0\cup S_1\cup S_2\cup S_3\cup S_4=X$，所以子集族 $F=\{S_0,S_1,S_2,S_3,S_4\}$ 是 X 的一个集合覆盖。而且，因为 $S_0\cup S_2\cup S_4=X$，且 F 中任何 2 个子集都不能覆盖 X，所以子集族 $C^*=\{S_0,S_2,S_4\}$ 是 $<X,F>$ 的一个最小集合覆盖。

2. 问题描述

集合覆盖问题：给定一个有限集 X 及 X 的一个集合覆盖 F 和一个正整数 k，判定是否存在 $C\subseteq F$ 使得 C 覆盖 X 且 $|C|=k$（或 $|C|\leq k$）。

3. 非确定性算法

```
bool SetCover(set<int> X, set<int> F[], int n, int k, bool W[])
{   int m = 0; // 选取的子集数
    // 非确定性地选取候选解W
    for(int i = 0; i < n; ++i)
        W[i] = Choice(0, 1), m += W[i];
    if(m > k) Failure();
    // 确定性地检查W是否覆盖了X
    for(int i = 0; i < n && !X.empty(); ++i)
        if(W[i]) X = X - F[i];
    if(X.empty()) Success();
    else Failure();
}
```

4. 复杂性分析

显然，该算法的实例特征为 n 和 $|X|$。非确定性地选取候选解耗时 $O(n)$，确定性地检查候选解是否为集合覆盖耗时 $O(n|X|)$。所以整个算法的时间复杂性为 $O(n|X|)$，是关于 n 和 $|X|$ 的多项式。

11.2.7 子集和问题SUBSET-SUM

1. 问题描述

给定正整数集合 S 和一个正整数 t，判定 S 是否存在和为 t 的子集。

2. 非确定性算法

```
bool SetSum(int S[], int n, int t, bool X[])
{   int w = 0; // 子集和
    for(int i = 0; i < n; ++i) // 选取候选解，计算和
        X[i] = Choice(0, 1), w += S[i] * X[i];
    if(w != t) Failure();
    else Success();
}
```

3. 复杂性分析

显然，非确定性地选取候选解并计算和耗时 $O(n)$，确定性地检查候选解是否满足要求耗时 $O(n)$。所以整个算法的时间复杂性为 $O(n)$。

11.2.8　0-1背包问题BKNAP

1. 问题描述

给定正数 c 和 v，及正数数组 $V=\{v_i\}_{i=0}^{n-1}$ 和 $W=\{w_i\}_{i=0}^{n-1}$，判定是否存在满足 $\sum_{i=0}^{n-1} w_i x_i \le c, x_i \in \{0,1\}$ 的 $\{x_i\}_{i=0}^{n-1}$，使得 $\sum_{i=0}^{n-1} v_i x_i = v$（或 $\sum_{i=0}^{n-1} v_i x_i \ge v$）。

2. 非确定性算法

```
bool BKNAP(double V[], double W[], int n, double c, double t, bool X[])
{   // 效益数组V, 重量数组W, 背包容量c
    double fv = 0, fw = 0; // 最后效益, 最后重量
    for(int i = 0; i < n; ++i) // 选取候选解, 计算效益和重量
        if(Choice(0, 1) == 1) X[i] = 1, fv += V[i], fw += W[i];
        else X[i] = 0;
    if(fw <= c && fv >= t) Success();
    else Failure();
}
```

3. 复杂性分析

显然，非确定性地选取候选解并计算效益和重量耗时 $O(n)$，确定性地检查候选解是否满足要求耗时 $O(n)$。所以整个算法的时间复杂性为 $O(n)$。

11.2.9 哈密尔顿回路问题HAMILTON

1. 问题描述

一个无向图是否含有哈密尔顿回路。

2. 非确定性算法

```
bool Hamilton(const Matrix<bool> &G, int X[])
{   int n = G.Rows();
    iota(X, X + n, 0); // 顶点号从0开始
    for(int i = n - 1, j; i >= 1; --i) // 非确定性地选取一个顶点序列
        j = Choice(1, i), swap(X[i], X[j]);
    // 确定性地检查该顶点序列是否构成回路
    for(int k = 0; k < n; ++k)
    {   int i = X[k], j = X[(k + 1) % n];
        if(G[i][j] != 1) Failure();
    }
    Success();
}
```

3. 复杂性分析

显然，非确定性地选取一个顶点序列耗时 $O(n)$，确定性地检查该顶点序列是否构成回路耗时 $O(n)$。所以整个算法的时间复杂性为 $O(n)$。

11.2.10　旅行商问题TSP

1. 问题描述

给定一个边权值非负的无向加权图 G 和一个非负数 t，判定 G 中是否存在费用为 t（或不超过 t）的旅行。

2. 非确定性算法

```
bool TSP(const Matrix<double> &G, double t, int X[])
{   int n = G.Rows();
    iota(X, X + n, 0); // 顶点号从0开始
    for(int i = 1, j; i < n; ++i) // 非确定性地选取一个顶点序列
        j = Choice(i, n - 1), swap(X[i], X[j]);
    // 确定性地计算回路费用
    double s = 0;
    for(int k = 0, i, j; k < n && s <= t ; ++k)
        i = X[k], j = X[(k + 1) % n], s += G[i][j];
    if(s > t) Failure(); // 不满足要求
    Success(); // 满足要求
}
```

3. 复杂性分析

显然，非确定性地选取一个顶点序列耗时 $O(n)$，确定性地计算旅行费用耗时 $O(n)$。所以整个算法的时间复杂性为 $O(n)$。

11.3 问题变换与*NP*完全问题

已经知道 $P \subseteq NP$。大多数计算机科学家认为 NP 类中包含了不属于 P 类的问题，即 $P \neq NP$。还存在一类 NP 完全问题，这类问题有一种令人惊奇的性质，如果一个 NP 完全问题能在多项式时间内成功求解，那么每一个 NP 问题都可以在多项式时间内成功求解，即 $P=NP$。但是，目前还没有一个 NP 完全问题有多项式时间算法。

11.3.1 问题变换

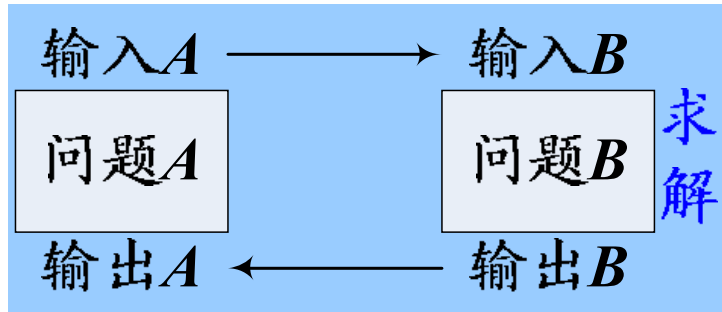
将问题 A 变换为问题 B 是指下列3个步骤。

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。

如果能够使用某个确定性算法在 $O(\tau(n))$ 时间内完成上述第①步和第③步，则称问题 A 可在 $\tau(n)$ 时间变换到问题 B ，简记为 $A \propto_{\tau(n)} B$ ，其中 n 通常是问题 A 的规模。若 $\tau(n)$ 是一个多项式函数，则称问题 A 可在多项式时间内变换到问题 B ，简记为 $A \propto B$ 。显然， $\propto$ 具有传递性，即若 $A \propto B$ ， $B \propto C$ ，则 $A \propto C$ 。



11.3.2 *NP*完全问题类

1. *NP*完全问题的定义

【定义】称问题 X 是 NP 难的当且仅当对任何 $X' \in NP$ 都有 $X' \propto X$ 。

【定义】称问题 X 是 NP 完全的当且仅当

① $X \in NP$；② X 是 NP 难的

【定义】所有 NP 完全问题构成的类称为 NP 完全问题类，记为 NPC 。

并非所有 NP 难问题都是 NP 完全的，例如，确定性算法的停机问题是 NP 难的，却不是 NP 的（请参阅《算法概论》）。

2. 几种复杂性类的关系

如果 $P \neq NP$，则几种复杂性类的关系可能如图11-2所示。

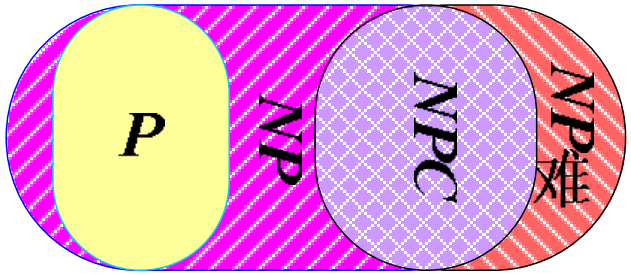


图11-2　几种复杂性类的关系

11.3.3 两个基本定理与*NP*完全树

1. *NP*完全性证明方法定理

【定理】设 X 是 NP 完全的，则

① $X \in P$ 当且仅当 $P=NP$；

② 若 $X \propto X'$，且 $X' \in NP$，则 X' 是 NP 完全的。

【证明】

① 只需证明当 $X \in P$ 时 $NP \subseteq P$ 成立。对任何 $Q \in NP$，由 $X \in NPC$ 可知，$Q \propto X$。由 $X \in P$ 和 $Q \propto X$ 可知，$Q \in P$。

② 对任何 $Q \in NP$，由 $X \in NPC$ 可知，$Q \propto X$。因为 $X \propto X'$，所以 $Q \propto X'$。由 $Q \propto X'$ 和 $X' \in NP$ 可知，$X' \in NPC$。

定理中的②可以用来证明一个新问题的 NP 完全性，前提是必须有第一个 NP 完全问题。Cook定理回答了该问题。

2. Cook定理

Cook于1971年提出并证明，Cook因此获得Turing奖。

【定理】布尔表达式的可满足性问题是NP完全的，即 $\mathrm{SAT} \in NPC$ 。

【证明】参阅Ellis Horowitz等的《Computer Algorithms, C++》。

3. NP完全树

Cook定理给出了第一个 NP 完全问题，使得对于任何问题 Q ，只要能证明 $Q \in NP$ 且 $\mathrm{SAT} \propto Q$ ，就有 $Q \in NPC$ 。所以，人们很快证明了许多其他问题的 NP 完全性。例如，Cook证明了 $\mathrm{SAT} \in NPC$ 的第二年（1972），Karp一口气提出了24个重要的 NP 完全问题(后来Karp也获得Turing奖,名字 NP 也是Karp给出的)。从此以后， NP 完全问题就像雨后春笋一般到处被发现。到1979年发现的 NP 完全问题就有约300种，现在已经发现了几千个 NP 完全问题。

但是，到目前为止，尚未证明任何一个 NP 完全问题是属于 P 的。这一事实，增强了人们对 $P \neq NP$ 的猜测。遗憾的是，这个猜测迄今仍然还只是个猜测。

这些 *NP* 完全问题都是直接或间接地以 SAT 的 *NP* 完全性为基础而得到证明的。由此逐渐生长出一棵以 SAT 为根的 *NP* 完全问题树。如图11-3所示的 *NP* 完全树是整个 *NP* 完全树的一小部分。每个结点代表一个 *NP* 完全问题，该问题可在多项式时间内变换为它的任一儿子结点表示的问题。目前这棵 *NP* 完全问题树上已有几千个结点，并且还在继续生长。

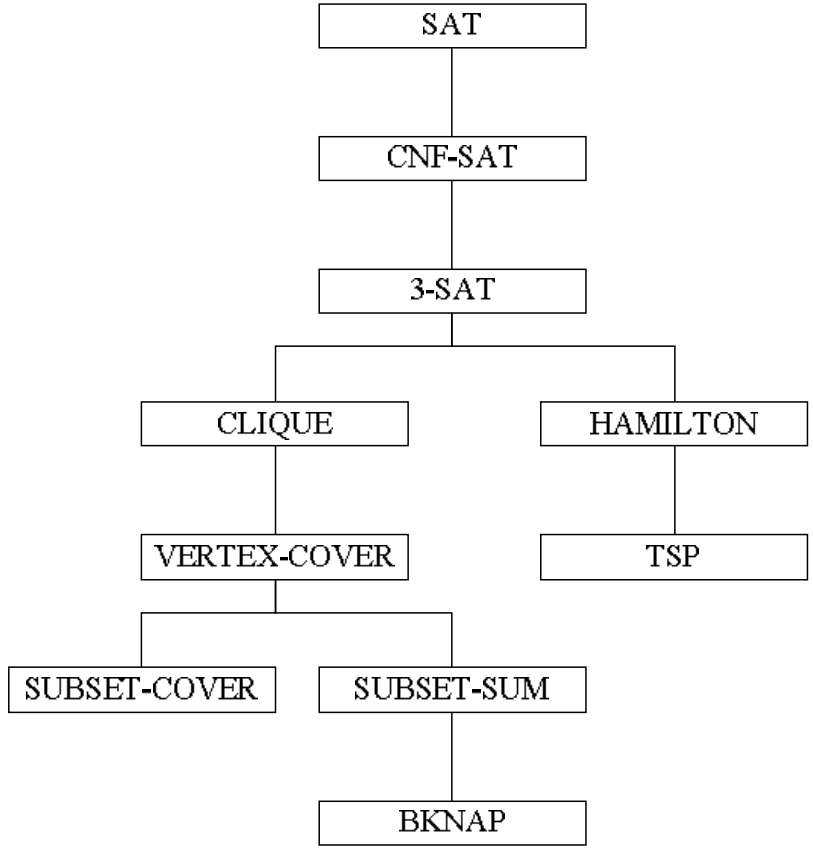


图11-3　*NP* 完全树

11.3.4 一些典型的NP完全问题[①]

1. 布尔表达式的可满足性问题SAT

给定一个布尔表达式，判定它是否可满足。

2. 合取范式的可满足性问题CNF-SAT

给定一个合取范式，判定它是否可满足。

3. 三元合取范式的可满足性问题3-SAT

给定一个三元合取范式，判定它是否可满足。

4. 团问题CLIQUE

给定无向图 G 和正整数 k，判定 G 是否存在 k 团（或不小于 k 的团）。

[①]注：除了这里列出的以外，本书涉及到的许多其他问题，如着色问题，皇后问题，多机调度问题等都是 NP 完全问题。

5. 顶点覆盖问题VRETEX-COVER

给定无向图 G 和正整数 k，判定 G 是否存在 k 顶点覆盖（或不大于 k 的顶点覆盖）。

6. 集合覆盖问题SUBSET-COVER

给定一个有限集 X 及 X 的一个集合覆盖 F 和一个正整数 k，判定是否存在 $C \subseteq F$ 使得 C 覆盖了 X 且 $|C|=k$（或 $|C|\leq k$）。

7. 子集和问题SUBSET-SUM

给定正整数集合 S 和一个正整数 t，判定 S 是否存在和为 t 的子集。

8. 0-1背包问题BKNAP

给定正数 c 和 v，及正数数组 $V=\{v_i\}_{i=0}^{n-1}$ 和 $W=\{w_i\}_{i=0}^{n-1}$，判定是否存在满足 $\sum_{i=0}^{n-1} w_i x_i \leq c, x_i \in \{0,1\}$ 的 $\{x_i\}_{i=0}^{n-1}$，使得 $\sum_{i=0}^{n-1} v_i x_i = v$（或 $\sum_{i=0}^{n-1} v_i x_i \geq v$）。

9. 哈密顿回路问题HAMILTON

一个无向图是否含有哈密顿回路。

10. 旅行商问题TSP

给定一个边权值非负的无向加权图 G 和一个非负数 t，判定 G 中是否存在费用为 t（或不超过 t）的旅行。

11.4 *NP*完全问题的证明举例

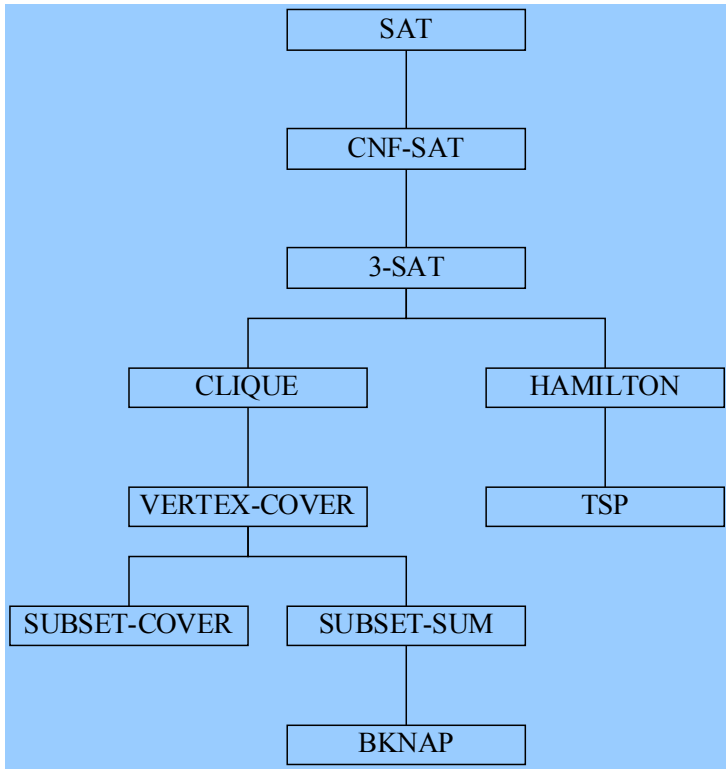
虽然已经发现了几千个 NP 完全问题，但是仍然有些问题依然无法判断它是属于 P 还是属于 NP 完全?下面给出几个证明 NP 完全性的例子。

【证明方法】要证明问题 $X' \in NPC$，只需证明

① $X' \in NP$。为 X' 寻找一个多项式时间内成功求解的非确定性算法。

② 利用一个已知的 NP 完全问题 X，证明 $X \propto X'$。

在下面的叙述中，假定CNF-SAT（以SAT为基础）、哈密尔顿回路问题（以3-SAT为基础）和子集和问题（以顶点覆盖问题为基础）的 NP 完全性已经得到证明。其中，CNF-SAT的 NP 完全性证明可参阅 Ellis Horowitz 等的《Computer Algorithms, C++》）、哈密尔顿回路问题和子集和问题的 NP 完全性证明可参阅傅清祥等的《算法与数据结构》。



11.4.1 3-SAT*

【问题】给定一个三元合取范式，判断它是否可满足。

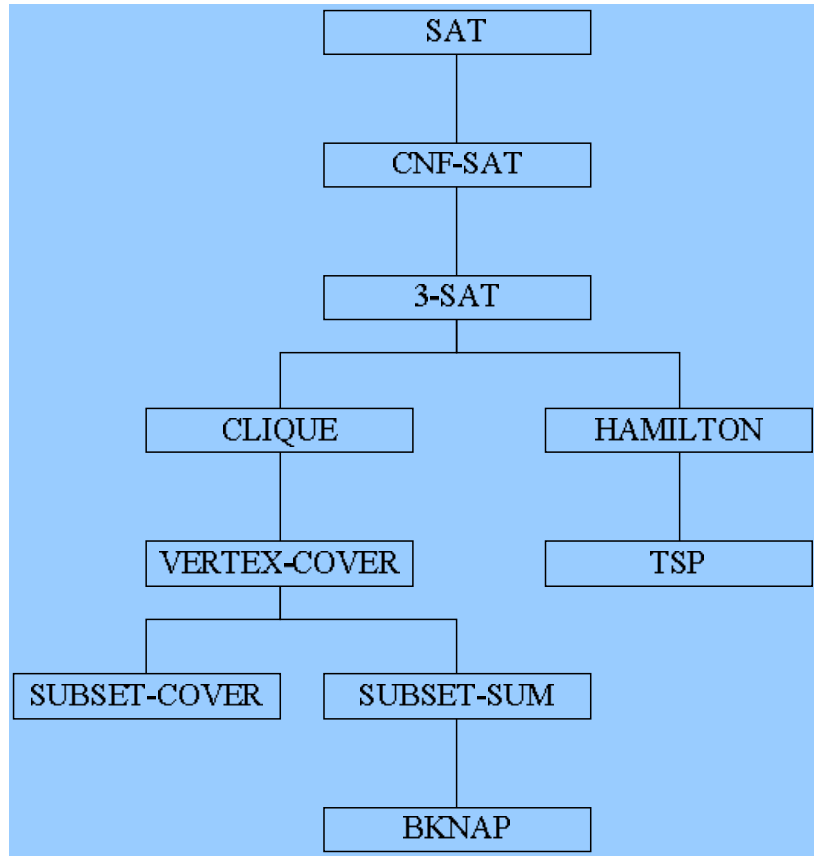
【定理】3-SAT是 NP 完全的。

【证明】

（1）由前已知3-SAT属于 NP 。

（2）证明CNF-SAT可在多项式时间内转化为3-SAT。

设 S 是一个合取范式，$C=x_1+x_2+\ldots+x_k$ 为 S 的一个子句。将 C 转化为三元合取范式 C'，并证明 $C=1$ 当且仅当 $C'=1$ 。分下列4种情况讨论。



① 当 $k \geq 4$ 时，取

$$C' = (x_1 + x_2 + y_1)(x_3 + \overline{y}_1 + y_2)\ldots(x_i + \overline{y}_{i-2} + y_{i-1})\cdots$$
$$(x_{k-2} + \overline{y}_{k-4} + y_{k-3})(x_{k-1} + x_k + \overline{y}_{k-3})$$

其中 $y_1, y_2, \ldots, y_{k-3}$ 是新加入的变量。

($\Rightarrow$) 若 $C = 1$，必有某一 x_i 的值为1，指定 $y_1, y_2, \ldots, y_{i-2}$ 的值为1，$y_{i-1}, y_i, \ldots, y_{k-3}$ 的值为0，显然，此时 $C' = 1$。

($\Leftarrow$) 设 $C' = 1$。因为当所有 $x_i = 0$ 时，

$$y_1(\overline{y}_1 + y_2)\ldots(\overline{y}_{i-2} + y_{i-1})\ldots(\overline{y}_{k-4} + y_{k-3})\overline{y}_{k-3} = 1$$

将导致 $y_1, y_2, \ldots, y_{k-3}$ 和 $\overline{y}_{k-3}$ 均为1，矛盾。所以，必有某一 $x_i = 1$，此时，$C = x_1 + x_2 + \ldots + x_k = 1$。

② 当 $k = 3$ 时，取 $C' = C$。显然，$C = 1$ 当且仅当 $C' = 1$。

③ 当 $k=2$ 时，取 $C'=(x_1+x_2+z)(x_1+x_2+\overline{z})$。

($\Rightarrow$) 若 $C=x_1+x_2=1$，则无论 z 取何值，都有 $C'=1$。

($\Leftarrow$) 若 $C'=1$，则必有 $C=x_1+x_2=1$，否则，将有 $C'=0$。

④ 当 $k=1$ 时，取 $C'=(x_1+y+z)(x_1+\overline{y}+z)(x_1+y+\overline{z})(x_1+\overline{y}+\overline{z})$。

($\Rightarrow$) 若 $C=x_1=1$，则显然 $C'=1$。

($\Leftarrow$) 若 $C'=1$，则 $C=x_1=1$。否则，

$$\begin{aligned}C' &= (y+z)(\overline{y}+z)(y+\overline{z})(\overline{y}+\overline{z}) \\ &= (y\overline{y}+yz+z\overline{y}+zz)(y\overline{y}+y\overline{z}+\overline{zy}+\overline{zz}) \\ &= (yz+\overline{y}z+z)(y\overline{z}+\overline{yz}+\overline{z}) \\ &= z\overline{z} \\ &= 0\end{aligned}$$

矛盾。

由①~④可知，CNF-SAT可在 $O(|S|)$ 时间内转化为3-SAT，故3-SAT是 NP 完全的。这里，$|S|$ 表示合取范式 S 的长度。

11.4.2 团问题

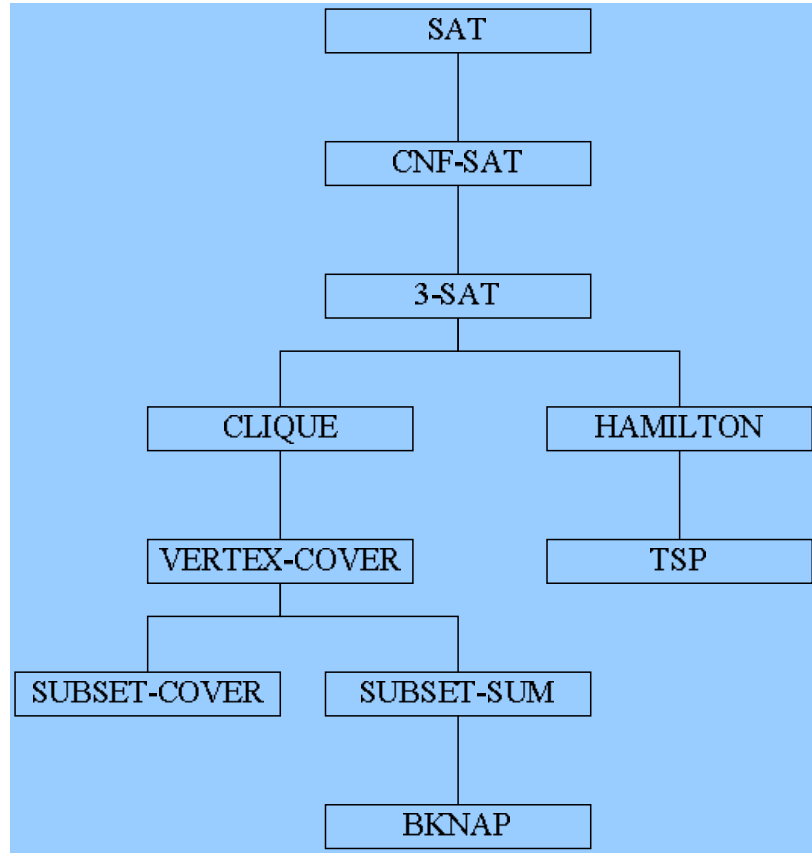
【问题】给定无向图 G 和正整数 m，判定 G 是否存在 m 团（或大小不小于 m 的团）。

【定理】团问题是 NP 完全的。

【证明】

（1）由前已知团问题属于 NP。

（2）证明三元合取范式可满足性问题可在多项式时间内转化为团问题。



设 $f=C_1C_2\ldots C_m$ 是一个三元合取范式，其中 $C_i=v_{i,1}+v_{i,2}+v_{i,3}$ ，$1\le i\le m$ 。构造图 $G=(V,E)$，其中

$$V=\{v_{1,1},v_{1,2},\cdots,v_{m,2},v_{m,3}\},E=\{(v_{r,i},v_{s,j})\,|\,r\neq s,v_{r,i}\neq\overline{v}_{s,j}\}$$

转换实例如图11-4所示。

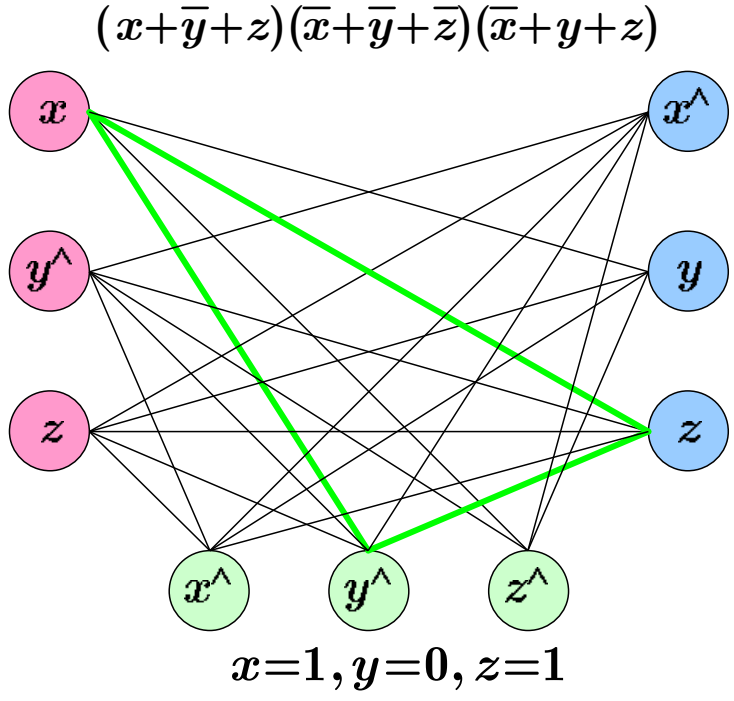


图11-4　三元合取范式转换成图

易知，该构造过程可在 $O(m^2)$ 时间内完成。下面证明 f 是可满足的当且仅当 G 有 m 团。

($\Rightarrow$) 若 $f=1$，则 $C_i=v_{i,1}+v_{i,2}+v_{i,3}=1$ $(1\le i\le m)$，从而 $v_{i,1},v_{i,2},v_{i,3}$ 至少有一个值为1，从中任取一个，记作 $v_{i,f(i)}$。令 $V'=\{v_{1,f(1)},\dots,v_{m,f(m)}\}$。显然，$|V'|=m$。因为对任何 $v_{i,f(i)},v_{j,f(j)}\in V'$，都有 $i\neq j$，由 $v_{i,f(i)}=v_{j,f(j)}=1$ 可知 $v_{i,f(i)}\neq\overline{v}_{j,f(j)}$，所以 $(v_{i,f(i)},v_{j,f(j)})\in E$。这说明 V' 是 G 的一个 m 团。

($\Leftarrow$) 若 V' 是 G 的一个 m 团。因为对任何 $1\le i\le m$，$v_{i,1},v_{i,2},v_{i,3}$ 之间没有边相连，不妨设 $V'=\{v_{1,f(1)},\dots,v_{m,f(m)}\}$。根据 G 的构造，对任何 $r\neq s$，都有 $v_{r,f(r)}\neq\overline{v}_{s,f(s)}$，所以，可以取 $v_{1,f(1)}=\dots=v_{m,f(m)}=1$。显然，此时 $f=1$。

11.4.3 顶点覆盖问题

【问题】给定无向图 G 和正整数 k，判定 G 是否存在 k 顶点覆盖（或不大于 k 的顶点覆盖）。

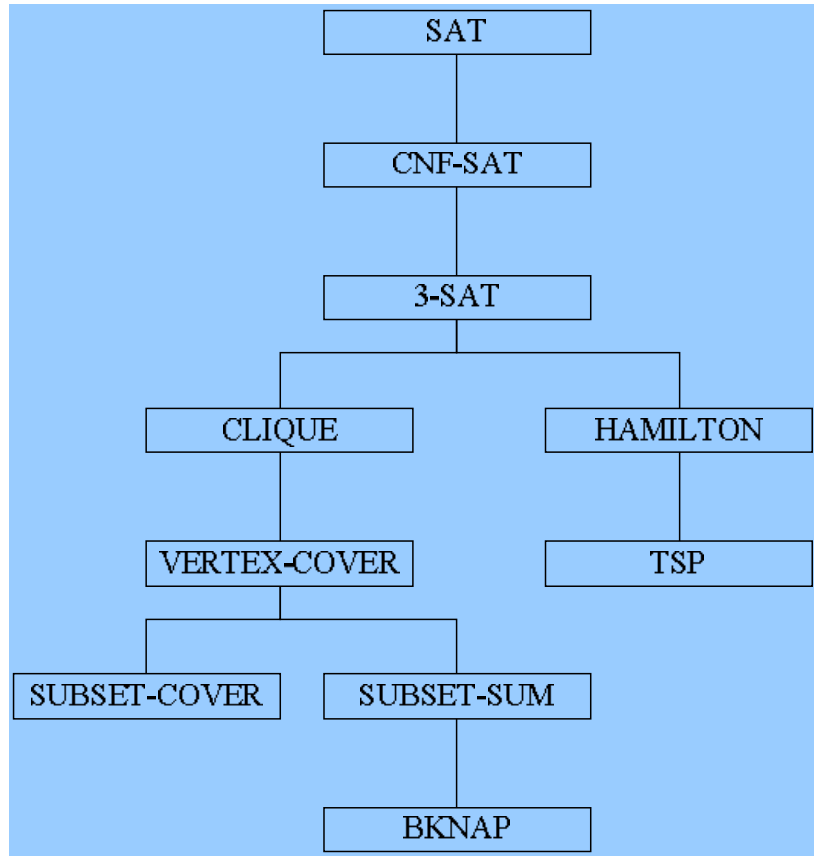
【定理】顶点覆盖问题是 NP 完全的。

【证明】

（1）由前已知顶点覆盖问题属于 NP。

（2）证明团问题可多项式时间转化为顶点覆盖问题。

易知，可以在 $O(n^2)$ 时间内构造出图 G 的补图 $\overline{G}$，其中 n 为图 G 的顶点数。



下面证明 G 有 $n-k$ 团当且仅当 $\overline{G}$ 有 k 顶点覆盖。

($\Rightarrow$) 设 V' 是 G 的 $n-k$ 团。显然，$|V-V'|=k$。对 $\overline{G}$ 的任何边 (u,v)，都有 $(u,v)\notin E$。因为 V' 是 G 的团，所以 $u\notin V'$ 或 $v\notin V'$，即 $u\in V-V'$ 或 $v\in V-V'$，从而边 (u,v) 被 $V-V'$ 覆盖。因此，$V-V'$ 是 $\overline{G}$ 的大小为 k 的顶点覆盖。

($\Leftarrow$) 设 V' 是 $\overline{G}$ 的 k 顶点覆盖。显然，$|V-V'|=n-k$。对任何 $u,v\in V-V'$，因为 $u\notin V'$ 且 $v\notin V'$，所以 (u,v) 不是 $\overline{G}$ 的边，从而 (u,v) 是 G 的边。这说明，$V-V'$ 是 G 的 $n-k$ 团。

11.4.4 集合覆盖问题

【问题】给定一个有限集 X 及 X 的一个集合覆盖 F 和一个正整数 k，判定是否存在 $C \subseteq F$ 使得 C 覆盖了 X 且 $|C|=k$（或 $|C|\leq k$）。

【定理】集合覆盖问题是 NP 完全的。

【证明】

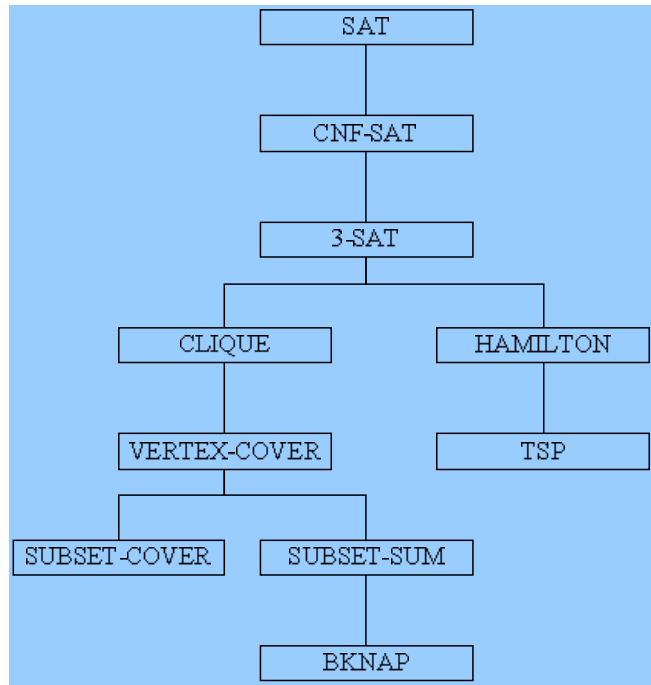
（1）由前已知集合覆盖问题属于 NP。

（2）证明顶点覆盖问题可在多项式时间内转换为集合覆盖问题。

设 $G=(V,E)$ 是一个无向图。对 $1\leq i\leq|V|$，用 S_i 表示与顶点 v_i 关联的所有边的集合。令 $X=E$，$F=\{S_1,S_2,\ldots,S_n\}$。

显然，X 和 F 的构造能在 $O(|E|)$ 时间内完成。

根据 X 和 F 的构造易知，$\{S_{i_1},S_{i_2},\ldots,S_{i_k}\}$ 覆盖 E 当且仅当 $\{v_{i_1},v_{i_2},\ldots,v_{i_k}\}$ 是 G 的一个顶点覆盖。



11.4.5 0-1背包问题

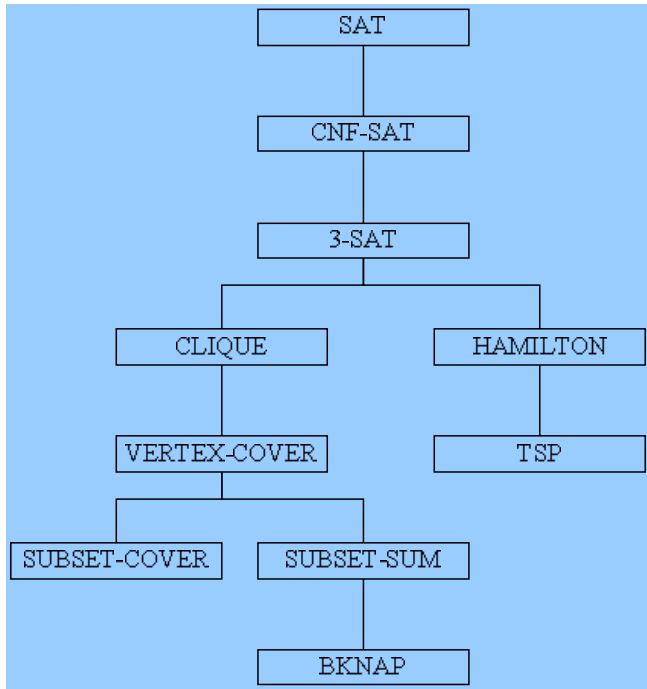
【问题】给定正数 c 和 v，及正数数组 $V=\{v_i\}_{i=0}^{n-1}$ 和 $W=\{w_i\}_{i=0}^{n-1}$，判定是否存在满足 $\sum_{i=0}^{n-1} w_i x_i \le c, x_i \in \{0,1\}$ 的 $\{x_i\}_{i=0}^{n-1}$，使得 $\sum_{i=0}^{n-1} v_i x_i = v$（或 $\sum_{i=0}^{n-1} v_i x_i \ge v$）。

【定理】0-1背包商问题是 NP 完全的。

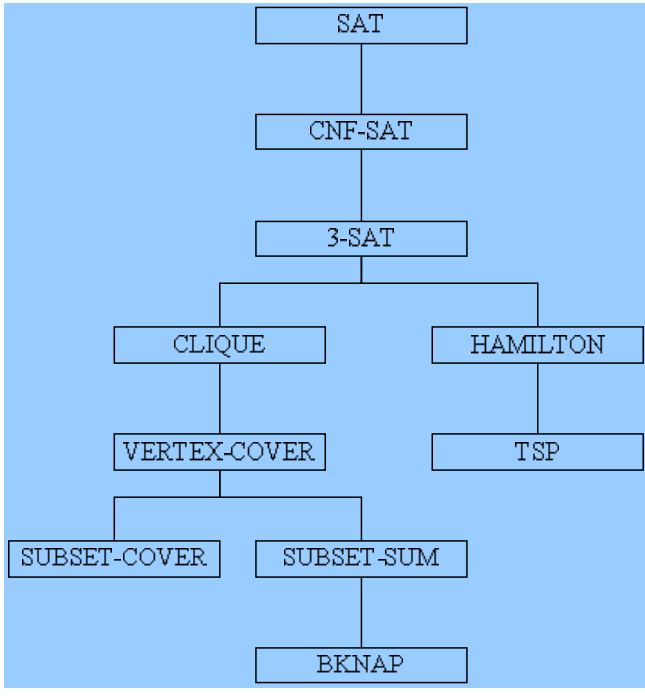
【证明】

（1）由前已知0-1背包问题属于 NP。

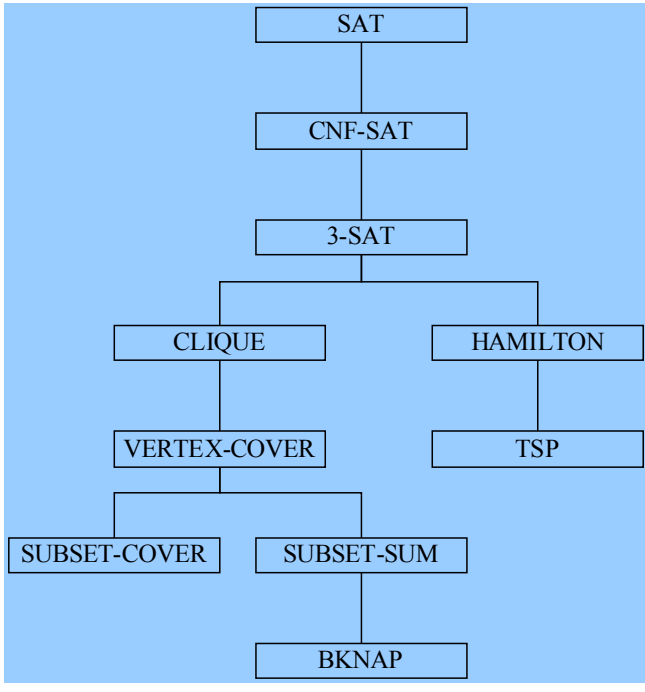
（2）证明子集和问题可在多项式时间内转换为0-1背包覆盖问题。



对于子集和问题的实例$<S,t>$，其中S是正整数集合，t是一个正整数，构造0-1背包问题的实例$<V,W,c,v>=<S,S,t,t>$。显然，$<V,W,c,v>$的构造能在$O(|S|)$时间内完成。根据$<V,W,c,v>$的构造易知，$\sum_{i=0}^{n-1} s_i x_i = t$当且仅当$\sum_{i=0}^{n-1} v_i x_i = v$（$\sum_{i=0}^{n-1} w_i x_i \le c$）。



11.4.6 旅行商问题



【问题】给定一个边权值非负的无向加权图 G 和一个非负数 t，判定 G 中是否存在费用为 t（或不超过 t）的旅行。

【定理】旅行商问题是 NP 完全的。

【证明】

（1）由前已知旅行商问题属于 NP。

（2）证明哈密尔顿回路问题可在多项式时间内转化为旅行商问题。

设图 $G=(V,E)$，构造完全图 $G'=(V,E')$，其中 $E'=\{(i,j)\,|\,i,j\in V\}$，费用函数定义为 $c(i,j)=\mathrm{iif}((i,j)\in E,0,1)$。显然，该构造过程可在 $O(|V|^2)$ 时间内完成。

若 G 有哈密尔顿回路，则该回路在 G' 中对应的回路就是费用为0的旅行。若 G' 有费用为0的旅行，则该旅行在 G 中对应的回路就是一条哈密尔顿回路。所以，G 有哈密尔顿回路当且仅当 G' 有费用为0的旅行。

11.5 练习题

1、解释下列概念。

P 类问题，NP 类问题，NPC 类问题，NP 难类问题

2、证明下列问题是 NP 的。

3-SAT，团问题，顶点覆盖问题，0-1背包问题，旅行商问题，着色问题，皇后问题，多机调度问题。

3、证明下列结论。

（1）CNF-SAT可在多项式时间内转化为3-SAT;

（2）3-SAT可可在多项式时间内转化为团问题;

（3）3-SAT可在多项式时间内转化为顶点覆盖问题;

（4）团问题可在多项式时间内转化为顶点覆盖问题;

（5）顶点覆盖问题可在多项式时间内转化为团问题;

（6）子集和问题可在多项式时间内转化为0-1背包问题;

（7）Hamilton回路问题可在多项式时间内转化为旅行商问题。

4、证明下列问题是 NPC 的。

3-SAT，团问题，顶点覆盖问题，0-1背包问题，旅行商问题。