证明格式

证明 Π∈NP

1. 最优性问题转为判定性问题
2. 构造非确定多项式时间算法
   a) 猜想……（多项式时间复杂度）
   b) 检查……（多项式时间复杂度）
   c) 若是，回答“yes”，否则回答“no”

多项式时间变换$\Pi_1 \leq_p \Pi_2$

1. 对Π_1的每个实例 I（列出所有参数和限制）
2. 构造Π_2对应实例 f(I)（计算参数，检查限制）
3. 证明 f 是多项式时间算法
4. 证明 I 是“yes”实例当且仅当 f(I)是“yes”实例

证明 Π 是 NP 完全的

1. 证明 Π∈NP
2. 找到已知的 NP 完全问题 Π’,证明 Π’$\leq_p$Π

1. SAT ∝ 0-1 整数规划

问题回顾：

SAT: 输入析取范式c_1，$c_2, \dots$，c_p，是否有赋值使得它们都为真

0-1 整数规划: 输入矩阵 C 和列向量 d，是否有 0-1 组成的列向量 X 使得 CX＝d

输入对应：

SAT 中的每个析取范式c_1，c_2, … , c_p是由符号集$\{x_1, x_2, \dots, x_n ; \overline{x_1}, \overline{x_2}, \dots, \overline{x_n}\}$的部分组成的。

构造相应的 0-1 整数规划问题的输入矩阵 C 和列向量 d

$$C_{ij} = \begin{cases} 1, & if\ \ x_j \in c_i \\ -1, & if\ \ \overline{x_j} \in c_i \\ 0, & otherwise \end{cases} \qquad b_i = 1 - (c_i\text{中补变量的个数})$$

$d_j = b_j + \Phi_i$

输入多项式时间转化：

转化的步骤需要构造矩阵 C，构造过程需要的复杂度为矩阵的元素数和列向量的元素数，即为 O(np)，是多项式的。

输出对应：

若 SAT 问题中存在赋值使得c_1，$c_2, \dots$，c_p都为真，将这个赋值作为 0-1 整数规划问题的列向量解 X，则有 CX 所得的列向量的分量有$d_i' = \sum_{j=0}^{n} Cij * x_j \geq b_i$,构造非负的松弛变量$\Phi_i$,使得$\sum_{j=0}^{n} Cij * x_j = b_i + \Phi_i$,则每个 SAT 的解对应上面的变换可以得到 CX = d 的解。

若 0-1 整数规划问题中有解 X 使得 CX＝d，则该 X 对应可以使得c_1，$c_2, \dots$，c_p均为真的赋值。考虑任意c_i，c_i为 0 当且仅当所有 $x_j \in c_i$ 取 $x_j = 0$，所有 $\overline{x_j} \in c_i$ 取 $x_j = 1$，则可知$\sum_{j=0}^{n} Cij * x_j = b_i - 1$，即若$c_i$不成立 CX 对应的分量不可能等于$d_i$，与 CX = d 矛盾。故$c_1$，$c_2$, … , c_p均为真。

2. SAT ∝ CLIQUE

问题回顾：

CLIQUE：输入一个图 G 和一个正整数 k，判断图 G 中是否有大小为 k 的团

输入对应：

图中顶点集为：

N = { <σ, i > | σ 是c_i中出现的符号}

边集为：

E = { {<σ, i >, <δ, j >} | i ≠ j，且 σ≠$\bar{\delta}$ }

k = p

输入多项式时间转化：

转化后所得到的顶点数，至多有 p*n 个，n 为析取范式中可能出现的符号数。

对没有个顶点<σ, i >， 可能与之相连的顶点<δ, j >，符号可能的取值有(2n-1)种，j 可能的取值最多有(p-1)种，故边总共不会超过(p*n) * (2n-1) * (p-1)条。

故转化的整体时间是 $O(p^2n^2)$的。

输出对应：

分析输入对应的性质，图中的每一个顶点其实表达的是一种状态：即符号 σ 可以使析取式取真。

若 SAT 问题中存在赋值使得所有的析取范式都为真，设赋值序列为$\{x_1, x_2, \ldots, x_n\}$，对每一个$c_i$，都可以在赋值序列中找到使得$c_i$对应为真的符号$\sigma_i$，这也就对应了图 G 中的一个点<$\sigma_i$, i>，则我们可以在图 G 中找到 p 个点<$\sigma_1$, 1>，<$\sigma_2$, 2>，…，<$\sigma_p$, p>.这 p 个点的第二个分量从 1 到 p 显然不相同，而由于这 p 个点的第一个符号是对应了同一个赋值序列中使析取式为真的赋值，故不会出现两个符号互为补变量的情况，故这 p 个点之间彼此都有边，即这 p 个点构成了一个 p 大团。

若 CLIQUE 问题解，即图中存在 p 个点彼此之间都有连线。根据构造输入时边的构造规则可知，这 p 个点第二个分量都不相同，所以每一个顶点可以对应一个符号的取值使得对应的析取式为真，同时这 p 个点的第一个分量互相不是补变量，即这些符号能同时取真，所以将这 p 个点对应的符号都取真，便可以对应 SAT 问题的一组使得所有析取式为真的输入。

3. CLIQUE ∝ SET PACKING

问题回顾：

SET PACKING：对集族$\{s_j\}$和一个正整数 l，在集族$\{s_j\}$是否存在 l 个互不相交的集合。

输入对应：

设原图中的节点为 N={1,2,3, … ,n}，每一个节点 i 对应 SET PACKING 问题中一个输入的集合s_i = {{i, j} | {i, j} ∉ A}，（A 是原图的边集）。取 l = k。

输入多项式时间转化：

原问题的节点的个数为 SET PACKING 问题的集合的个数，集合中元素的和为原图的补图中边的条数，所以元素个数至多为 n(n-1)/2,所以构造相应的输入需要的复杂度是 $O(n^2)$的。

输出对应：

如果 CLIQUE 问题中存在一个 k 大团，则将这些顶点对应的集合也构成 SET PACKING 问题一个满足要求的实例，若存在两个集合使得$s_i \cap s_j$不为空，则$s_i \cap s_j$中的元素仅能为{i, j}，但若{i, j}∈s_i，则表示{i, j} ∉ A，这与 i 和 j 是原图 k 大团中的两个顶点{i, j} ∉ A 矛

盾。所以不存在$s_i \cap s_j$不为空，所以选出来的这 l 个集合是满足要求的。

如果 SET PACKING 问题中存在 l 个集合彼此之间没有交集，则这些集合对应的顶点在原图中构成一个 l 大团。设 i 和 j 是根据 SET PACKING 问题的实例选出的两个顶点，若 i 和 j 之间没有边，则{i, j}∈s_i且{i, j}∈s_j，这与$s_i \cap s_j$为空集矛盾，所以选出来的这 l 个顶点，任意两点之间都有边。所以这 l 个点是满足要求的实例。

4. CLIQUE ∝ NODE COVER

问题回顾：

NODE COVER：输入一张图 G'和一个正整数 l，是否存在一个点集 R⊆N'使得|R| ≦ l，且图 G'中的每个边都在 R 中找到与之相关联的点。

输入对应：

在 NODE COVER 问题中取 G'为 CLIQUE 中图 G 的补图，取 l = |N| - k

输入多项式时间转化：

图 G'中顶点的个数为原图的顶点数 n，边的条数为 n(n-1)-m (m 为原图中边的条数)所以转化是多项式时间的

输出对应：

如果 CLIQUE 问题中存在一个满足要求的实例。即图 G 中存在一个 k 阶完全子图 K，设这个子图对应的顶点的集合为 Nk。则 N-Nk 能构成 NODE COVER 问题中的点覆盖。若在图 G'中存在一条边 e 与 N-Nk 中的点不相关，则 e 连接了 Nk 中的两个点，设为 i ,j，即{i, j}属于 G 的补图中，即{i, j}∉K，与 K 为 k 阶完全子图矛盾。所以 G'中任意一条边都与 N-Nk 中的点相关，所以 N-Nk 构成了 G'中的一个点覆盖。且|N-Nk| = |N| - k。

如果 NODE COVER 问题中存在一个满足要求的点覆盖 R。则在图 G'中任意两点 i 和 j∈（N-R），i 和 j 之间没有边，否则与 R 是图 G'的点覆盖矛盾。又因为图 G'是图 G 的补图，所以任意两点 i 和 j 属于(N-R)，在图 G 中 i 和 j 中有边，所以(N-R)构成图 G 中的一个完全子图，这个完全子图的阶数是大于等于 k 的，所以图 G 中存在 k 阶完全子图。

5. NODE COVER ∝ SET COVERING

问题回顾：

SET COVERING：给定有限个集合组成的集族$\{s_j\}$，和一个正整数 k。判断是否存在$\{s_j\}$的一个子集$\{T_h\} \subseteq \{s_j\}$，使得$\cup\{T_h\} = \cup\{s_j\}$，且$|\{T_h\}| \le k$

输入对应：

设{1,2, … , n}为原图 G 中的顶点集，则取集合s_j为 NODE COVER 图 G 中与顶点 j 相关的边组成的集合作为 SET COVERING 问题的输入。且去 k = l。

输入多项式时间转化：

图 G 中有 n 个顶点，所以 SET COVERING 问题中有 n 个集合，这些集合内的元素为图 G 中的边。而图 G 中的边同时与两个点相关，所以会出现在两个集合中，所以所有集合的元素的个数和为 2m，所以输入转化的复杂度为 O(n+2m)

输出对应：

若 NODE COVER 问题中存在满足条件的点覆盖集 R，|R|≤k，则与 R 中的顶点相关的集合作为T_h的元素。则图 G 中任意一条边 e 都能在 R 中找到与之相关的点 i，则 e∈s_i，所以 e∈$\cup\{T_h\}$，所以$\cup\{T_h\} = \cup\{s_j\} =$ 原图的边集，且$|\{T_h\}| = |R| \le k$，所以 SET COVERING 中存在符合条件的$\{T_h\}$。

如果 SET PACKING 问题中存在$\{T_h\}$，则取$\{T_h\}$中的集合对应的顶点作为图 G 的点覆盖 R 即可，因为$\cup\{T_h\} = \cup\{s_j\} =$ 原图的边集，所以对原图中任意一条边 e，都能找到s_j，使得 e∈s_j，所以 e 与 R 中的顶点 j 相关，所以 R 是满足条件的点覆盖。

6. NODE COVER ∝ FEEDBACK NODE SET

问题回顾：

FEEDBACK NODE SET：给定有向图 H 和正整数 k，判断是否有一个点集 R⊆V，使得 H 中一个环在 R 中有一个相关的点，且|R| ≤ k。

输入对应：

取原图的顶点集作为有向图 H 的顶点集 V

将原图中的边转化成 H 中的正反两向的两条边，E = {<u, v> | {u, v} 属于 A'}

取 k = l。

输入多项式时间转化：

转化后的图 H 中的顶点数和原图相同，G 中每条边转化为 H 中两条边，|E| = 2m，所以转化的复杂度为 O(n+2m)

输出对应：

若 NODE COVER 中存在一个点覆盖，则取这个点覆盖为 FNS 问题的 R，则|R| ≤ k。有向图 H 中的任一个环 C，其中任意的边设为<u, v>，这条边对应图 G 中的{u, v}，而 R 中有与{u, v}相关的点，这个点也与<u, v>有关，即与 C 相关。所以 R 是符合要求的 FNS 问题的解。

若 FNS 问题中存在一个符合要求的点集，则可取这个点集作为原图 G 中的点覆盖 R，R 满足|R| ≤ k。对于原图中任意一条边{u, v}，图 H 中存在环<u, v><v, u>与之对应，而再 H 中这个环与 R 中的一个点相关，所以 u 或 v 在 R 中，所以在图 G 中{u, v}在 R 中存在与之相关的点，所以 R 是图 G 的一个点覆盖。

7. NODE COVER ∝ FEEDBACK ARC SET

问题回顾：

FEEDBACK ARC SET：给定有向图 H 和正整数 k，判断是否有一个边集 S ⊆ E，使得 H 中一个环在 S 中有一个相关的边，且|S| ≤ k。

输入对应：

将原图中的每个点分成出点和入点。V = N×{0, 1}

原图所有的边转化成从出点连到入点的两条边，所有的点的入点到出点有边

即 E = { <<u, 0>, <u, 1>> | u∈N' }∪{ <<u, 1>, <v, 0>> | {u, v}∈A'}

取 k = l。

输入多项式时间转化：

有向图 H 中的顶点个数为 G 中的两倍，G 中的所有的边转化成了 H 中两条相关的边，E 中还有从每个点的入点到出点的边，所以 E 中元素个数为 n+2m。所以转化的复杂度为 O(2n+n+2m) = O(3n+2m)。

输出对应：

如果 NODE COVER 问题中存在点覆盖 R，在 H 中取 E = {<<u, 0>, <u, 1>> | u∈R}。则|E| = |R| ≤ k。对 H 中任何一个环 C 中的连接不同顶点的边<<u, 1>, <v, 0>>，其对应的 G 中的边{u, v}在 R 中有与之相关的点不妨设为 u，所以<<u, 0>, <u, 1>> ∈ E，而由

于环 C 经过点 u，所以<<u, 0>, <u, 1>>为环 C 中的一边，即 E 中存在与环 C 相关的点。所以 E 满足要求。

如果 FAS 问题中存在满足要求的 E，将 E 做如下的转化：若存在<<u, 1>, <v, 0>>∈E，则可将<<u, 1>, <v, 0>>替换为<<v, 0>, <v, 1>>。因为所有与<<u, 1>, <v, 0>>相关的环 C，经过了点 v，则环 C 与<<u, 1>, <v, 0>>相连的下一条边必为<<v, 0>, <v, 1>> 所以所有与<<u, 1>, <v, 0>>相关的环，都与<<v, 0>, <v, 1>>相关，所以替换后得到的 E'仍满足要求，且|E'|≤|E|≤k。完成所有的转化后则得到一个均由<<v, 0>, <v, 0>>元素组成的边集 E''，取 E''中边对应的点为图 G 中的点覆盖 R。首先，|R| = |E''|≤|E'|≤|E|≤k 对图 G 中的任一条边{u, v}，在 H 中存在一个对应的环<<v, 0>, <v, 1>> -> <<v, 1>, <u, 0>> -> <<u, 0>, <u, 1>> -> <<u, 1>, <v, 0>>，这个环在 E''中存在相关的边为<<v, 0>, <v, 1>>或者<<u, 0>, <u, 1>>，则 u 或 v 属于 R，即边{u, v}能在 R 中找到与之相关的点。所以 R 是满足条件的点覆盖。

8. NODE COVER ∝ DIRECTED HAMILTON CIRCUIT

问题回顾：

哈密顿图：能找到一条经过所有顶点一次的回路。

输入对应：

设点覆盖问题对应的图为 G，顶点集为 N，边集为 A，要找的点覆盖的顶点数不超过 l。

定义有向哈密顿图的顶点集为：(将原图中的边编号从 1 到 m)

V = $\{a_1, a_2, a_3, \dots, a_l\}$∪{<u, i ,0>, <u, i, 1> | u 是原图 G 中的顶点，i 是与 u 相连的边}

定义有向哈密顿图的边集为：

E = { <<u, i, 0>, <u, i, 1>> | <u, i, 0>∈V } 第①类边

∪{ <<u, i, a>, <v, i, a>> | u 和 v 被编号 i 的边相连,a∈{0, 1}} 第②类边

∪{ <<u, i, 1>,<u, j, 0>> | 不存在 i<k<j,使得<u, k, 0>∈V} 第③类边

∪{ <<u, i, 1>, a_f> | 1≤f≤l,而且不存在 h>i，使得<u, h, i>∈V} 第④类边

∪{ <a_f, <u, i, 0>> | 1≤f≤l,而且不存在 h<i，使得<u, h, i>∈V} 第⑤类边

输入多项式时间转化：

设原图有 n 个点 m 条边。则得到的有向哈密顿图 H 有 l+4m 个点。因为每条边连接两个点，对应了 4 个 H 中的<u, i ,a>型顶点。

分析各类边的条数：（认为<u, i, a>中 a = 1 表示出点，a = 0 表示入点）

第①类边：原图中一条边连接两个点，对应两个不同的<u, i>，所以这类边有 2m 条

第②类边：一条边 i 对应的两个不同的<u, i><v, i>之间的 4 条边，这类变有 4m 条。

第③类边：每个点 u 连接的边按编号大小顺序排好，相邻的边 i 和 j 对应的顶点<u, i, 1><u, j, 0>之间有边，所以连接 u 和 v 的边，在两个顶点处都至多作为一条第③类边的始点，所以第③类边最多有 2m 条。

第④类边：每个点标号最大的边的出点与所有的用于标记的$\{a_1, a_2, a_3, \dots, a_l\}$相连，故有 nl 条边

第⑤类边：同上这一类也有 nl 条边

综上复杂度为 O(4m+l+2m+4m+2m+2nl) = O(m+nl)，也是多项式复杂度的。

输出对应：

- 若原图中存在一个最小的点覆盖 R = $\{r_1, r_2, r_3, \dots, r_t\}$ (t ≤ l)。我们可以向 R 中添加若干个其他点使得 R 的规模变成 l，即我们可以得到一个 R' =$\{r_1, r_2, r_3, \dots, r_l\}$
  按如下方法可遍历有向哈密顿图中的顶点：

- 将图 G 中所有的边按 R'划分，首先将所有的与r_1相连的边放在A_{r_1}中，然后将所有与r_2相连的边放在A_{r_2}中，则这样划分后我们得到了分别与$r_1,r_2,r_3,\dots,r_t$相连的边的集合：$A_{r_1},A_{r_2},\dots,A_{r_l}$。可知这些集合并集为 A 且都不会是空集。因为我们选择的时候可以选择有边与之相关连的点。但这些集合彼此之间可能有交集。
- 在图 G 中我们可以从r_1出发，按照A_{r_1}中边的编号顺序，先从r_1通过一条边走到下一个点，然后再从这个点回到r_1，如此可以把A_{r_1}中的每一条边走一遍，而对应到哈密顿图上的路线是：$<r_1,e_1,0><v_1,e_1,0><v_1,e_1,1><r_1,e_1,1><r_1,e_2,0><v_2,e_2,0><v_2,e_2,1><r_1,e_2,1>\cdots\ldots\ldots$达到的效果是将哈密顿图中所有与$A_{r_1}$中边有关的点都遍历了。
- 用上面的方法我们能对每一个A_{r_i}，在哈密顿图中找到一条遍历所有与A_{r_i}有关的点的路线。由于A_{r_i}的并是 A，所以哈密顿图中所有非a_i的点都在这些路线上。剩下所需要做的事情就是将这些路线和$\{a_1,a_2,a_3,\dots,a_l\}$连成一个不重复经过某些顶点的环。
- 从a_1出发，由第⑤类边的构成可知，a_1能走到A_{r_i}对应的路线的第一个点$<r_1,e_1,0>$，则a_1能与A_{r_i}对应的路线相连，现在得到的路线的末端是$<r_1,e_h,1>$，其中e_h是与r_1相连的编号最大的边，由第④类边的构成可知$<r_1,e_h,1>$与a_2相连，而a_2又能与A_{r_2}对应的路线的起点相连，所以我们可以通过a_2连接器A_{r_1}和A_{r_2}对应的路线。
- 如果$A_{r_1},A_{r_2},\dots,A_{r_l}$之间没有交集的话，则我们可以直接通过$\{a_1,a_2,a_3,\dots,a_l\}$将这些路线连接起来，最后一个集合对应的终点再连接到a_1就构成了一个不重复经过任意顶点的环，且这个环经过了图中所有顶点。即是一条哈密顿回路。如果连接路线的过程中发现当前连接的路线与前面的路线有相同的点，则说明这个相同的点对应的边 i 连接的两个点都在 R'中，设为 u 和 v。则这些重复的点出现在前面是以<u, i, 0><v, i, 0><v, i, 1><u, i, 1>的形式，出现在后面是以<v, i, 0><u, i, 0><u, i, 1><v, i, 0>的形式，则只需要将前面的路线该部分调整为<u, i, 0><u, i, 1>，后面的路线该部分调整为<v, i, 0><v, i, 1>，则这两条路线就没有了重叠的部分，并还是能经过原先能经过的点。所以只要在连接的时候进行这样的调整就能连出一条 H 中的哈密顿回路。

● 如果图 H 中存在一条哈密顿回路。

则取与$\{a_1,a_2,a_3,\dots,a_l\}$相连接的点对应的点集，则可以构成原图 G 的一个点覆盖。先对哈密顿图中的边作分析可知，设边 i 连接点 u 和 v，则可能存在的情况：

<u, i, 0>后：<u, i ,1><v, i, 0>

<u, i, 0>前：<v, i, 0><u, j, 1>，a_i

<u, i, 1>后: <u, j, 0> <v, i, 1>，a_i

<u, i, 1>前：<u, i, 0> < v, i ,1>

- 考虑与a_1相关的顶点<u, i, 0>，这个点的含义是从顶点 u 通过边 i 走到另外一个顶点 v。根据之前的分析

  在哈密顿图中能与<u, i, 0>相连的点只有<u, i, 1>，<v, i, 0>（v 是通过 i 与 u 相连的点）。

  如果<u, i, 0>连接了<u, i, 1>，则在<u, i, 1>后面只能连接<u, j, 0>或者a_i（如果存在另外的边与 u 相连，记为 j）。此时对应于图 G 中的行为相当于从 u 出发通过 i 走到了另一个点再回到 u 点，然后准备走下一条与 u 相连的边 j 或者回到a_2。

如果<u, i, 0>连接了<v, i, 0>。则<v, i, 0>后只能连接<v, i, 1>。而由于<u, i, 1>前只能是<u, i, 0>或者<v, i, 1>则这里<v, i, 1>后只能连接<u, i , 1>，则又回到了与上面相同的情况。即又回到了<u, j, 0>或者a_i。

所以可以发现从a_1出发到走到下一个a_i前，这条回路上的点经过的都是与 u 有关的边。即 u 可以覆盖回路上这一段点对应的所有边。

所以按照这种方法可以分析得到，在从a_i出发到<u', i', 0>到走到下一个a_j前，回路上所有的点相关的边都能被 u'覆盖。所以从$\{a_1, a_2, a_3, \ldots, a_l\}$总共将回路分为 l 段。每一段上涉及图 G 中的边都能被一个顶点覆盖。所以至多 l 个顶点就能覆盖 G 中所有的边。所以 G 中存在规模不超过 l 的点覆盖。

9. DIRECTED HAMILTON CIRCUIT ∝ UNDIRECTED HAMILTON CIRCUIT

输入对应：

无向哈密顿图中的顶点集为：

N=V×{0, 1, 2}　　即将原图中的点分成对应的三个点。

边集定义为：

A = {{<u, 0>, <u, 1>}, {<u, 1>, <u, 2>} | u∈V}

　∪{{<u, 2>, <v, 0>} | <u, v>∈E}

输入多项式时间转化：

生成的无向哈密顿图中的点集大小为原来的三倍，设原图中有 n 个点 m 条边，则生成的新图中有 3n 个点和 2n+m 条边。

输出对应：

若在原来的有向哈密顿图中存在一条哈密顿回路依次经过$u_1 u_2 u_3 \ldots u_n u_1$，则这条回路依次经过的边为$<u_1, u_2>$,$<u_2, u_3>$…$<u_{n-1}, u_n>$,$<u_n, u_1>$, 对于新得到的无向图原图中从 u_1 到 u_2 的路径$<u_1, u_2>$在新图中对应路径$\{<u_1, 0>, <u_1, 1>\}$, $\{<u_1, 1>, <u_1, 2>\}\{<u_1, 2>, <u_2, 0>\}$，从$u_2$到$u_3$的路径$<u_2, u_3>$在新图中对应路径$\{<u_2, 0>, <u_2, 1>\}$, $\{<u_2, 1>, <u_2, 2>\}\{<u_2, 2>, <u_3, 0>\}$，所以经过这种对应我们可以得到一条依次经过$<u_1, 0><u_1, 1><u_1, 2><u_2, 0><u_2, 1><u_2, 2>\ldots<u_n, 0><u_n, 1><u_n, 2>$再回到$<u_1, 0>$的哈密顿回路。

若得到的无向哈密顿图中存在一条哈密顿回路，由于$<u_i, 1>$的前驱只能为$<u_i, 0>$后继只能为$<u_i, 2>$，所以要经过所有的点，则回路只能依次经过：$<u_1, 0><u_1, 1><u_1, 2><u_2, 0><u_2, 1><u_2, 2>\ldots<u_n, 0><u_n, 1><u_n, 2><u_1, 0>$，而$<u_1, 1><u_1, 2><u_2, 0>$在原图中就对应了从$u_1$到$u_2$的有向边，所以将这条哈密顿回路对应了原图中$<u_1, u_2>$,$<u_2, u_3>$…$<u_{n-1}, u_n>$,$<u_n, u_1>$的一条回路，这条回路经过了原图中所有的点，所以是一条哈密顿回路。

10. SATISFIABILITY ∝ SATISFIABILITY WITH AT MOST 3 LITERALS PER CLAUSE

输入对应：

将 SAT 问题中的一个语句做如下的转化：

$\sigma_1 \cup \sigma_2 \cup \ldots \cup \sigma_m = (\sigma_1 \cup \sigma_2 \cup u_1)(\sigma_3 \cup \ldots \cup \sigma_m \cup \overline{u_1})(\overline{\sigma_3} \cup u_1) \ldots (\overline{\sigma_m} \cup u_1)$

一直到每个语句的长度不超过 3 为止。

输入多项式时间转化：

将上面一条长为 m 经过一次转化，得到了 m 个语句，其中有(m-1)个语句长度是不超

过 3 的，还有一条长度为(m-1)的语句，第二次转化的时候就看成将一条长度为(m-1)的语句进行转化。所以将一条长度为 m 的语句全部转化成不超过 3 的语句，转化出来得到的语句数为(m-1)+(m-2)+...+4+3 = (m+2)(m-3)/2。设有 p 条语句，每条语句每场为 n，则最后转化得到的 3SAT 问题中的语句不超过 p(n+2)(n-3)/2 条，所以这个转化是多项式复杂度的。

输出对应：

如果在 SAT 问题中存在一个取值使得所有的语句为真。考虑$\sigma_1 \cup \sigma_2 \cup ... \cup \sigma_m$，则$\sigma_1 \cup \sigma_2$取真或$\sigma_3 \cup ... \cup \sigma_m$取真。若$\sigma_3 \cup ... \cup \sigma_m$值为 0，必有$\sigma_1 \cup \sigma_2$为 1，说明$\sigma_3 ... \sigma_m$都是 0，此时取$u_1$为 0，则$(\sigma_1 \cup \sigma_2 \cup u_1)$因为$\sigma_1 \cup \sigma_2$为 1 取真，$(\sigma_3 \cup ... \cup \sigma_m \cup \overline{u_1})$因为$\overline{u_1}$取 1 为真，后面的小括号因为$\sigma_3 ... \sigma_m$为 0 取真。若$\sigma_3 \cup ... \cup \sigma_m$值为 1，则取$u_1$为 1，也可以保证所有式子取真。则综上，如果 SAT 中一个语句取真，则存在赋值方法使得由它转化得到的语句均为真。

若 3SAT 问题中存在一个赋值使得所有的语句都为真。仍考虑$\sigma_1 \cup \sigma_2 \cup ... \cup \sigma_m$和它转化得到的语句。若$u_1$取 0，则要使得$(\sigma_1 \cup \sigma_2 \cup u_1)$为真，只能是$\sigma_1 \cup \sigma_2$为 1，此时$\sigma_1 \cup \sigma_2 \cup ... \cup \sigma_m$为真。若$u_1$取 1，则要使得$(\sigma_3 \cup ... \cup \sigma_m \cup \overline{u_1})$为真，只能是$\sigma_3 \cup ... \cup \sigma_m$为 1，此时也能得到$\sigma_1 \cup \sigma_2 \cup ... \cup \sigma_m$为真。所以如果原语句转化得到的 3SAT 中的语句都能取真，则这条语句也能取真。所以 3SAT 中一个符合要求的赋值对应了 SAT 问题中一个满足要求的赋值。

11. SATISFIABILITY WITH AT MOST 3 LITERALS PER CLAUSE ∝ CHROMATIC NUMBER

输入对应：

取 3SAT 中的所有的符号和它们的补符号，以及所有的析取范式$D_1, D_2, ..., D_r$，以及引入$v_1, v_2, ..., v_m$表示 m 种颜色。(m 为 3SAT 问题中符号的个数)

N = $\{u_1, u_2, ..., u_m\} \cup \{\overline{u_1}, \overline{u_2}, ..., \overline{u_m}\} \cup \{v_1, v_2, ..., v_m\} \cup \{D_1, D_2, ..., D_r\}$

构造图中的边：

A = $\{\{u_i, \overline{u_i}\} \mid i = 1, 2, ..., m\}$ 第①类边：所有的符号和它的补之间有边

$\cup \{\{v_i, v_j\} \mid i \neq j\}$ 第②类边：所有表示颜色的点之间有边

$\cup \{\{v_i, u_j\}, \{v_i, \overline{u_j}\} \mid i \neq j\}$ 第③类边：每个符号和它的补与编号不同的颜色点有边

$\cup \{\{u_i, D_f\} \mid u_i \notin D_f\} \cup \{\{\overline{u_i}, D_f\} \mid \overline{u_i} \notin D_f\}$

第④类边：如果$u_i \in D_f$，则D_f与u_i之间没有边，与和$\overline{u_i}$之间有边；如果$\overline{u_i} \in D_f$，则D_f与u_i有边，与$\overline{u_i}$之间没有边。如果$u_i \notin D_f$且$\overline{u_i} \notin D_f$，则$D_f$与$u_i$和$\overline{u_i}$之间都有边，即$D_f$与所有未出现在其中的符号有边。

输入多项式时间转化：

N 中顶点的个数为 3m+r，A 中边的个数:第①类有 m 个，第②类有 m(m-1)/2，第③类有 2m(m-1)个，第④类有至多 2mr 个。综上转化的复杂度是 O(m²+mr)的。

输出对应：

若 3SAT 中存在一个满足要求的取值使得所有的析取范式为真，我们按如下的顺序染色。首先所有取 1 的u_i对应的点染成颜色 i，将对应的$\overline{u_i}$染成颜色 m+1。将取 0 的u_i对应的点染成颜色 m+1，将对应的$\overline{u_i}$染成颜色 i，然后将所有的v_i染成与之编号相同的颜色 i。则对第①类边，它连接的是符号u_i和$\overline{u_i}$，这两个点一个被染成颜色 i，一个被染成颜色 m+1，故第①类边连接的顶点颜色不同。对第②类边，它连接的是所有表示颜色的点，这些点染的颜色是与自己编号对应的，所以第②类边连接的点因为编号不同所以颜色不同。对第③类边，它连接的是点u_i和$\overline{u_i}$和与它们编号不同的颜色对应的节点，则u_i和$\overline{u_i}$

被染为 i 和 m+1 色，与之相连的表示颜色的点v_j，j≠i，所以v_j染的颜色不是 i 或 m+1，所以第③类边连接的点颜色也不同。接下来对D_p染色和第④类边。对每一个析取式D_p，若存在$u_i∈D_p$，且u_i取 1，则将D_p染成颜色 i，因为能染成颜色 i 的 u 型点点此时只有u_i，而D_p与它不相连，所以将D_p染成 i 色不会产生矛盾。若存在$\overline{u_i}∈D_p$，且$\overline{u_i}$取 1，则将D_p染成 i，和前面一样分析可知这样染色不会产生矛盾。而对于每一个D_p，必定存在$u_i∈D_p$，且u_i取 1 或$\overline{u_i}∈D_p$，且$\overline{u_i}$取 1，否则D_p不会取真。所以对每一个D_p都能找到一个合理的染色方式。所以我们能按这种方法将所有的点染色，且互相不冲突。

如果原图中存在一个 m+1 色的染色方案。首先，$\{v_1, v_2, \ldots, v_m\}$肯定染上的是 m 种不同的颜色，我们不妨把这些颜色按v_1，v_2，…，v_m顺次编号为 1 到 m，剩下一种颜色编号为 m+1。对每一对顶点u_i和$\overline{u_i}$，它们只能被染成 i 和 m+1，若u_i染成了 i，则取原先的输入为$u_i = 1$，若是$\overline{u_i}$染成了 i，则取$u_i = 0$。对每一个析取式D_p，若存在$u_i∈D_p$，且u_i染成了 i，则说明u_i取 1 能是D_p取真。同样如果存在$\overline{u_i}∈D_p$，且$\overline{u_i}$染成了颜色 i，则说明$\overline{u_i}$取 1 能是D_p取真。若对任意的$u_t∈D_p$，$\overline{u_s}∈D_p$，这些点都没有染成对应的颜色 t 和 s，则说明与D_p相连的$\overline{u_t}$和u_s，它们被染成了颜色 t 和颜色 s，所以D_p不能被染成 s 和 t。而对于u_i和$\overline{u_i}$都没有出现在D_p的编号 i，D_p不能染成 i 和 m+1（因为是 3SAT 问题，所以这样的点肯定存在）。如此，D_p不能染 1 到 m+1 的任意一种颜色。这与原图能找到 m+1 的染色方案矛盾。所以对任意D_p总能找到$u_t∈D_p$或$\overline{u_s}∈D_p$被染上了对应的 t 或者 s，使得D_p取值为真。

12. CHROMATIC NUMBER ∝ CLIQUE COVER

问题回顾：

CLIQUE COVER：属于一个图 G'和一个正整数 l，判断图中的顶点 N'中是否能分成至多 k 个团的顶点的并。

输入对应：

在 CLIQUE COVER 中取图 G'为 k 着色问题图 G 的补图，取 l 等于 k。

输入多项式时间转化：

所需要做的事情就是把一个图转化成它的补图，是多项式时间的。

输出对应：

如果 k 着色问题中存在一种着色方案，则被染成 i 颜色的顶点之间在图 G 中彼此没有边，则它们在 G'中构成一个团。所以每一种颜色对应 G'中一个团，因为不超过 k 成颜色，所以在 G'中对应的团不超过 k 个，又因为每个点都被染色了，所以每个点都在这些对应的团中。所以我们在 G'中找到了对应的不超过 l 个的团。

如果在 CLIQUE COVER 问题中存在不超过 l 个团，使得它们并起来是图 G'的顶点。则对于其中一个团，团中任意两个顶点之间在图 G'中有连线，则在 G 中这些点彼此不相连，所以他们能被染成一种颜色。所以我们按顺序将这不超过 l 个团，从颜色 1 开始染一个团内的点，所以只需要不超过 l 种颜色就能将这些团中的点全都染色。这种染色方案在图 G 中不会产生冲突。所以我们在图 G 中找到了对应的染色方案。

13. CHROMATIC NUMBER ∝ EXACT COVER

输入对应：

转化出的恰好覆盖的需要覆盖的元素集合为

N∪A∪{<u, e, f> | 顶点 u 与边 e 相关联，1≦f≦k}

用来覆盖这些元素的s_j有两种形式：

I = { u, <u, e, f> | e 与 u 相关联}，对每一个 1≦f≦k 和每一个 u∈N 对应这样一个集合

II = {e, <u, e, f>, <v, e, g> | f≠f_1, g≠f_2, u 和 v 是与 e 相关联的两个点}

输入多项式时间转化：

设原图有 n 个顶点和 m 条边

转化后需要覆盖的元素的个数为|N| + |A| + |{<u, e, f>}| = O(n + m + n(n-1)k)

第一类集合的个数为 O(n(n-1)k)，第二类集合的个数为 O(2m(k-2))

所以转化的过程是多项式时间的

输出对应：

若原先的图中存在一种着色方案，节点u_i着色 f(i)，则在 I 类集合中取出所有{u_i, <u_i, e, f(i)>}，在 II 中选择{e, <u_i, e, f>, <u_j, e, g> }，f_1= f(i)，f_2= f(j)。首先证这些集合之间没有交集，I 类集合与 II 类集合互相之间没有交集，因为 II 类集合中选取的颜色是与第一类中的 f(i)不等的，I 类集合彼此之间没有交集，因为它们对应的u_i不同，所以它们对应的<u_i, e, f(i)>不同，II 类集合彼此之间没有交集，因为它们对应的 e 不同，所以它们对应的<u_i, e, f>也不同。所以选出来的这些集合彼此之间没有交集。再证这些集合能覆盖所有元素。首先所有的点u_i和所有的边 e 显然是能被覆盖。剩下的是所有的<u, e, f>，而所有这些元素可以根据 u 和相应的 e 分类，每一个 u 和相关的 e 对应 k 个<u, e, f>。固定 u 和 e，设 e 连接了 u 和 v，u 染了 f(u)，v 染了 f(v)，则<u, e, f(u)>在选出的 I 类集合中，剩下的(k-1)个相关元素在{e, <u_i, e, f>, <u_j, e, g> }，f_1= f(u)，f_2= f(v)中，所以所有的元素都能被覆盖。故原图的着色方案能对应一个恰好覆盖的解。

如果恰好覆盖存在一个解。由于要覆盖所有的元素，且集合之间没有交集，所以在 I 类集合中，每个不同的 u 选择了一个集合，则将选出的 I 类集合中 u 对应的 k 作为原图的染色方案。下证这种染色方案不会出现一条边连接的两个顶点染上了同一种颜色。对于任意一条边 e，用来覆盖这条边的集合为 II 类集合，设这条连接了 u 和 v，u 对应的颜色为 f(u)，v 对应的颜色为 f(v)。易知 e 对应的{e, <u, e, f>, <v, e, g> | f≠f_1, g≠f_2}，中必有f_1= f(u)且f_2= f(v)，若f_1≠f(u)，<u, e, f(u)>既属于选出的 I 类集合又属于选出的 II 类集合，则这两个集合之间有交集，不满足恰好覆盖的要求。所以必有f_1= f(u)且f_2= f(v)，而根据输入转化的构造方法$f_1 \neq f_2$，所以 f(u) ≠ f(v)，所以任意一条边连接的两个点会被染上的颜色不一样，所以这是一个符合要求的染色方法。

14. EXACT COVER ∝ HITTING SET

问题回顾：

EXACT COVER：给定一些集合s_j，在其中找到一部分集合能覆盖{u_i}，且彼此无交集。

HITTING SET：给定一些集合u_i，一个集族{s_j}，u_i是集族{s_j}的子集，有一个{s_j}的子集 W 使得对任意的集合u_i，|W∩u_i| = 1.

输入对应：

对于恰好覆盖的s_j和{u_i}，转化成 HITTING SET 问题的输入u_i' = {s_j | u_i∈s_j}，集族{s_j}的每个元素就是恰好覆盖输入的集合。

输入多项式时间转化：

设 EXACT COVER 有 p 个集合，n 个元素。则对应的 HITTING SET 问题的集合有 n 个集合，每个集合至多 p 个元素。集族的元素个数为 p。所以转化的复杂度是 O(np)的。

输出对应：

如果恰好覆盖存在一个解，则将这些选出的集合组成集合 W，则 W 为 HITTING SET 的解。因为这些集合覆盖了$\{u_i\}$，所以对任意u_i'，W 中都存在一个元素s_j使得$u_i \in s_j$，所以$|W \cap u_i'| > 0$，若$|W \cap u_i'| > 1$，即存在s_j和s_k属于 W，使得$u_i \in s_j$且$u_i \in s_k$，这与s_j和s_k交集为空矛盾，所以$|W \cap u_i'| = 1$，所以选出的 W 满足要求。

如果 HITTING SET 存在一个解 W，则 W 中的集合构成了恰好覆盖，对于原问题的任意一个元素u_i，$|W \cap u_i'| = 1$，即 W 中仅存在一个元素s_j使得$u_i \in s_j$，所以 W 中的集合能覆盖所有的元素u_i，另一方面设s_j和s_k属于 W，则它们之间交集为空，否则存在一个元素u_i，使得$u_i \in s_j$且$u_i \in s_k$，则$|W \cap u_i'|$ 至少为 2，这与$|W \cap u_i'| = 1$ 矛盾，所以 W 中所有的集合之间交集为空。所以 W 中所有集合能构成原问题的一个恰好覆盖。

15. EXACT COVER ∝ STEINER TREE

问题回顾：

STENIER TREE：在图 G 中能找到一颗权重为 k 的树 T，使得图的顶点的子集 R 包含于树 T

输入对应：

图 G 中的顶点集：$N = \{n_0\} \cup \{s_j\} \cup \{u_i\}$，需要包含的顶点集 $R = \{n_0\} \cup \{u_i\}$

图 G 中的边集：$A = \{\{n_0, s_j\}\} \cup \{\{s_j, u_i\} |\ u_i \in s_j\}$

边的权重函数：$w(\{n_0, s_j\}) = |s_j|, w(\{s_j, u_i\}) = 0$，要达到的权重 $k = |\{u_i\}|$.

输入多项式时间转化：

构造顶点的复杂度 $= O(1+p+n)$，p 表示集合个数，n 表示元素个数

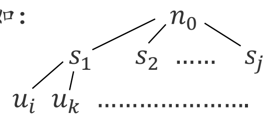
构造边集的复杂度不超过 $O(pn + p + 1)$

构造权重函数 $O(pn+2p+1)$

故整体复杂度为 $O(pn)$,是多项式时间的

输出对应：

转化后的图 G 形如：



基本是树形，但可能存在最下面一层不同的 s 连向相同的 u

如果原来 EXACT COVER 存在解，则在图 G 中选择对应的 s 节点，则这些 s 节点和节点n_0，和所有 u 节点构成了一颗树，因为这些集合 s 恰好覆盖了所有 u 节点，故边的权重之和为 u 节点的个数，即为 k，另一方面，因为 s 之间互相没有交集，所以选出来的节点能构成一棵树。

如果 STEINER TREE 问题有对应的解 T，则在 T 中的 s 即为恰好覆盖问题中的解。首先这些 s 集合中各自包含的元素之和为 u 节点的个数（因为权重和为 k）。其次，若选出来的 s 节点中存在两个s_i和s_j使得它们的交集不是$\emptyset$，设存在u_k同时属于这两个集合，则选出来的 T 中存在边$\{n_0, s_i\}$,$\{n_0,s_j\}\{s_i,u_k\}\{s_j,u_k\}$这四条边构成了一个环，与 T 是树矛盾，故选出来 s 集合中任意两个元素均没有交集，故这些集合恰好能覆盖所有的 u 元素。

16. EAXCT COVER ∝ 3-DIMENSIONAL MATCHING

问题回顾：

3-DIMENSIONAL MATCHING：

T 是一个有限集合，给出一个集合 U ⊆ T×T×T，找出 U 的一个子集 W ⊆ U，使得|W| = |T|，且 W 中的每个元素三个方向的维度都不相等。

输入对应：

令 T = {<i, j> | $u_i \in s_j$}，令 α 是一个从u_i到 T 的单射，令 π：T→T 是使得对每个固定的 j，{<i, j> | $u_i \in s_j$}是 π 的一个圈（即轮换）的函数。β 是一个与<i, j>有关的函数。

U = { <<α(u_i), <i, j>, <i, j>> | <i, j> ∈ T }

  ∪ { <β, σ, π(σ)> | 对所有的 i, β≠α(u_i)}

输入多项式时间转化：

T 中的元素本质是描述原来输入集合的性质，即某个集合包括了哪些元素，所以|T| = ∑|s_j|，对 T 中一个元素<i, j>，它在 U 中第二个维度出现只有两种形式：<<α(u_i), <i, j>, <i, j>> 和 < β(<i, j>), <i, j>, π(<i, j>) >，故 U 中的元素个数有 2|T|个，故构造的三维匹配的输入的元素个数和为 3|T|，故输入转化为 O(np).

输出对应：

T 中元素的个数 = ∑|s_j|

如果恰好覆盖问题有解，我们让这里的 α 和 β 是恒等映射。对于选出的集合s_j，和该集合当中的任何一个元素u_i，选出 U 中对应的元素：<<i, j>, <i, j>, <i, j>>，则这些元素三个维度彼此不相同。对于未选出的集合s_k，对于该集合中任何元素u_l，选出 U 中对应的元素<<l, k>, <l, k>,π<l, k>>，这些元素与前一类元素三个维度一定不相同，因为它们对应不同的 s 集合，且 π 函数是与 s 相关的轮换，所以每个维度的第二个变量都不相同。而这一类元素彼此之间每个维度也不相等，首先如果它们来自不同的 s 集合，则必然第二个分量都不一样，如果它们与同一个s_j有关，则第一个维度和第二个维度一定不一样，又因为 π 函数是一个轮换，所以 π(σ)也不一样。所以按照这种方法选出来的集合是满足条件的 W。

如果三维匹配问题有对应的解。U 中第一种元素，对每个u_i我们只能取一个<α(u_i), <i, j>, <i, j>>，将这些元素对应的集合s_j作为恰好覆盖问题的解，现在是这些s_j必能覆盖所有u_i，下证它们彼此之间没有交集。由于|T|中元素的个数 = 不同的<i, j>的个数，因为选出来的 W 中元素个数 =T 的元素个数，则 W 的每个维度都由所有不同的<i, j>构成。所以如果现在选出来的s_j中还有其他元素，则这些元素对应的 U 中的元素只能在 U 第二类中，即为<β, <i, j>, π(<i, j>)>，由于 π(<i, j>)是一个与 j 有关的环，所以对于选出来的集合s_j，π 总能将与s_j有关的第二类元素中的关系<i, j>映射到第一类元素中的关系<k, j>，则这样就会使得与这两个关系相关的 W 中的元素在第三个维度上相等。所以是s_j是不存在其他元素的。所以这些选出来的s_j就是一个恰好覆盖。

17. EXACT COVER ∝ KNAPSACK

问题回顾：

KNAPSACK：给出一些背包{a_1, a_2, a_3, … , a_r}和质量 b，是否存在对背包的选择使得 $\sum a_i x_i$ = b.

输入对应：

令 r = |{s_j}|，即原先输入的集合的个数，令 d = r + 1，令$a_j = \sum d^{i-1}(u_i \in s_j)$, b = $\frac{d^t-1}{d-1}$

输入多项式时间对应：

这里计算a_j的时候需要的工作量为s_j包含的元素的个数，所以构造输入为 O(rn)的。

输出对应：

分析这里的a_j，每个a_j其实可以看成一个长度为 n 的 d 进制数(n 为 u 元素的个数)，这个进制的第 i 位为 1 表示u_i在s_j中，否则第 i 位为 0，而这里的 b 写成 d 进制就是 1111…1(共 n 个 1)。

如果恰好覆盖中存在解，则这些选出来的s_j对应的a_j作为 d 进制数时，d 进制数的每一位有且仅有一个a_j使得该位为 1，其他的a_j该位都为 0，所以这些a_j的和为 1111…1(共 n 个 1)，即为 d，所以恰好覆盖的解对应背包问题的一个解。

如果背包问题中存在解，则将选出来的背包作为恰好覆盖问题中选出来的集合，则这些集合对应的a_j的和为 b＝1111…1(共 n 个 1)，如果分析所有的a_j，则同时只能有一个a_j最高位为 1，否则和有 n+1 位，不可能为 b，则将所有a_j的最高位去掉等效于将这个数最高位的 1 变成 0，变成a_j'，同时将 b 的最高位的 1 也变成 0，变成 b'，则仍有a_j'的和为 b' = 111…1(共 n-1 个 1)，同样可以分析到有且只有一个对应的a_k' 第 n-1 位为 1，然后又可以把所有数的第 n-1 位去掉，如此一直下去可知 d 进制数的每一位有且只有一个a_j使该位为 1，而a_j的一位为 1 表示的是该位对应的 u 元素在集合s_j中，所以可知这些集合能覆盖所有 u 元素，且彼此之间没有交集。

18. KNAPSACK ∝ SEQUENCING

问题回顾：

SEQUENCING：输入

任务执行时间：$(T_1, T_2, \dots, T_p) \in Z^p$

任务截止时间：$(D_1, D_2, \dots, D_p) \in Z^p$

任务延时惩罚：$(P_1, P_2, \dots, P_p) \in Z^p$

正整数 k。

是否存在一种任务调度使得总惩罚不超过 k。

输入对应：

取任务的数量为背包问题中的背包数，任务 i 的执行时间和惩罚时间都是a_i，

即$P_i = T_i = a_i$，取每个任务的截止时间都为 b，即$D_i = b$

取 $k = \sum_{i=1}^{p} a_i - b$

输入多项式时间转化：

从原问题生成了新问题的输入为（3p+1），是多项式时间的。

输出对应：

设在截止时间 b 之间完成了任务s_1到s_n，则完成任务的时间为$\sum_{i=1}^{n} T_{s_i}$，$\sum_{i=1}^{n} T_{s_i} \le b$，剩下的任务为$s_{n+1}$到$s_p$，这些任务需要惩罚，所以总惩罚为$\sum_{i=n+1}^{p} P_{s_i} = \sum_{i=1}^{p} T_{s_i} - \sum_{i=1}^{n} T_{s_i} = \sum_{i=1}^{p} a_i - \sum_{i=1}^{n} T_{s_i} \ge \sum_{i=1}^{p} a_i - b = k$

若背包问题中存在一个选择方案，使得选择的背包的重量和为 b，则将这些背包对应的任务先安排，则这些问题执行时间恰好为 b，最后的惩罚恰好为 k，即这是满足条件的任务调度方案。

如果 SEQUENCING 问题中存在一种调度方式使得惩罚不超过 k，由又上面的分析可知惩罚是不低于 k 的，所以惩罚只能为 k。当惩罚为 k 时，即存在一些对任务的选择使得$\sum_{i=1}^{n} T_{s_i} = b$，则将这些任务对应的背包作为背包问题的选择，则这些选出来的背包重量和为 b，即这种选择也是一种满足要求的实例。

19. KNAPSACK ∝ PARTITION

问题回顾：

PARTITION：$(c_1, c_2, \dots, c_s) \in Z^s$，是否存在I ⊆ {1,2, … , s}，使得$\sum_{h\in I} c_h = \sum_{h\notin I} c_h$

输入对应：

取 s = r+2，$c_i = a_i (i = 1, 2, \dots, r), c_{r+1} = b + 1, c_{r+2} = \sum_{i=1}^{r} a_i + 1 - b$

输入多项式时间转化：

原先的输入转化成这里的输入，只需要 O(r+2)步

输出对应：

若背包问题存在解，即存在一些背包的重量和为 b，则将这些背包对应的c_i和c_{r+2}放在一组，则它们的和为$\sum_{i=1}^{r} a_i + 1$，$\sum_{i=1}^{r+2} c_i = \sum_{i=1}^{r} c_i + (\mathrm{b}+1) + (\sum_{i=1}^{r} a_i + 1 - b) = 2(\sum_{i=1}^{r} a_i + 1)$，所以剩下的一部元素和也为$\sum_{i=1}^{r} a_i + 1$，所以这样可以对应 PARTITION 问题的一个解。

若 PARTITION 问题存在解。由上面的分析可知$\sum_{i=1}^{r+2} c_i = 2(\sum_{i=1}^{r} a_i + 1)$，所以$(c_1, c_2, \dots, c_{r+2})$能分成两组，每组中元素的和为$\sum_{i=1}^{r} a_i + 1$。首先，$c_{r+1}$和$c_{r+2}$不在一组中，因为$c_{r+2} + c_{r+2} = (\mathrm{b}+1) + (\sum_{i=1}^{r} a_i + 1 - b) = \sum_{i=1}^{r} a_i + 2$。则与$c_{r+2}$在一组的元素之和为$\sum_{i=1}^{r} a_i + 1 - c_{r+2} = \sum_{i=1}^{r} a_i + 1 - (\sum_{i=1}^{r} a_i + 1 - b) = \mathrm{b}$，而这些元素在$(c_1, c_2, \dots, c_r)$中，所以它们对应了原问题的背包的重量。所以选出来的这些元素对应的背包重量和为 b，即背包问题有解。

20. PARTITION ∝ MAX CUT

问题回顾：

MAX CUT：输入一个图 G，边集记为 A，w 是 A 的权重函数，给定一个正整数 W，是否存在原图顶点的一个子集 S，使得$\sum_{\substack{\{u,v\}\in A \\ u\in S \\ v\notin S}} w(\{u,v\}) \geq \mathrm{W}$

输入对应：

取 N={1, 2, … , s}　A = {{i, j} | i ≠ j}

w({i, j}) = $c_i * c_j$

W = $\lceil \frac{1}{4}(\sum c_i)^2 \rceil$

输入多项式时间转化：

在 MAX CUT 问题中有 s 个顶点，有 s(s-1)/2 条边，W 的计算至多需要 s 步。所以输入转化的时间复杂度为 O(s²)

输出对应：

设 S 是图 G 的顶点的一个子集，则$\sum_{\substack{\{u,v\}\in A \\ u\in S \\ v\notin S}} w(\{u,v\}) = \sum_{\substack{i\in S \\ j\notin S}} c_i * c_j = (\sum_{i\in S} c_i) * (\sum_{j\notin S} c_j)$

设 $(\sum_{i\in S} c_i) = x, (\sum_{i\in N} c_i) = Total, then\ (\sum_{j\notin S} c_j) = Total - x$, so $\sum_{\substack{\{u,v\}\in A \\ u\in S \\ v\notin S}} w(\{u,v\}) = x(Total - x) = -\left(x - \frac{Total}{2}\right)^2 + \left(\frac{\mathrm{Total}}{2}\right)^2 = -\left(x - \frac{Total}{2}\right)^2 + \mathrm{W}$。

若 PARTITION 存在一个符合要求的划分，则取这个划分对应的顶点组成集合 S，则在 MAX CUT 中$x - \frac{Total}{2} = 0$，$\sum_{\substack{\{u,v\}\in A \\ u\in S \\ v\notin S}} w(\{u,v\}) = W$，满足要求，即 MAX CUT 问题中也存在对应要求的实例。

若在 MAX CUT 问题中存在一个使得$\sum_{\substack{\{u,v\}\in A \\ u\in S \\ v\notin S}} w(\{u,v\}) \geq W$的子集 S，由$\sum_{\substack{\{u,v\}\in A \\ u\in S \\ v\notin S}} w(\{u,v\}) \leq W$，可知$\sum_{\substack{\{u,v\}\in A \\ u\in S \\ v\notin S}} w(\{u,v\}) = W$，所以$x - \frac{Total}{2} = 0$，即 S 中点对应的权重和为$\frac{Total}{2}$，而 S 的补集中的点对应的权重和也为$\frac{Total}{2}$，所以 S 对应在 PARTITION 问题中的元素满足要求。即此时也能找到对应的满足要求的实例。