

我们用 $f(i)$ 表示第 i 天最后能得到多少钱，并且第 i 天要做出卖出操作。

当状态 $f(i)$ 被计算完之后，我们用 $x(i), y(i)$ 表示将 $f(i)$ 这么多钱卖出能得到的 A 券和 B 券的数量，这个可以通过解方程得出。那么很容易得到状态转移方程：

$$f(i) = \max_{j=0}^{i-1} [x(j)^* s[j] + b(j)^* y(j)], \text{ 最后的答案} = \text{Max}(f[i], 0 \leq i \leq n)$$

1、SAT 问题 (SATISFIABILITY)

判断合取范式 (有限个简单析取式的合取) 是否可满足)

2、0-1 整数规划 (0-1 INTEGER PROGRAMMING)

对整形矩阵 C 和整形向量 d ，判断是否存在 0-1 向量 x ，
s.t. $Cx=d$ 。

3、最大团 (CLIQUE)

判断图 G 中是否存在规模为 K 的团。

4、(SET PACKING)

判断集合族中是否存在 $|I|$ 个两两不交的集合。

3a4 集合数 $n=|V|, |I|=K$

构建集合 S_i 是从第 i 个点出发的到其它所有点的边中不在 E 中的那些。

于是， $S_i \cap S_j = \emptyset$ 当且仅当 $j \in E$ 。

5、最小点覆盖 (NODE COVER)

判断是否存在 G 中规模 $\leq I$ 的点集覆盖 G 中所有弧 (E)

3a5 未被点覆盖选中的点间在图中无边，在补图中成团。

即补图中存在规模为 $|V|-K$ 的点覆盖则原图中存在规模为 K 的团。

6、集合覆盖 (SET COVERING)

判断有限集 P 有限子集族是否存在规模不超过 I 的覆盖。

5a6 全集是 G 中的边，集族中 $S_i=$ 第 i 个点相邻的边的集合。

7、反馈节点集 (FEEDBACK NODE SET)

对有向图 H 和正整数 k ，问是否存在规模不超过 k 的 V 的子集 C ，s.t. 对 H 中的任一圆，都有 C 中的点。

5a7 将无向图中的每个边都用一组边反向有向代替，即 k 。这样由于点覆盖所有边，且每条边都只有 2 个顶点，所以是等价的。

8、反馈弧集 (FEEDBACK ARC SET)

对有向图 H 和正整数 k ，问是否存在规模不超过 k 的 E 的子集 C ，s.t. 对 H 中的任一圆，都有 C 中的弧。

5a8 将无向图中的每个边拆成两个部分 $(u, v)(v, u)$ ，前者表示入点，后者表示出点。对每个分段的节点，再从入点引出出点。对于无向图中的每条边，分裂成两条边，因为每条边有入点，这样，每条边就成为一个圆，等价于有向图的圆。

9、有向哈密尔顿回路 (DIRECTED HAMILTON CIRCUIT)

判断有向图 H 有没有无重复遍历所有点的有向回路。

10、无向哈密尔顿回路 (UNDIRECTED HAMILTON CIRCUIT)

判断无向图 G 有没有无重复遍历所有点的回路。

9a10 如同书上的做法，将每个点拆分成出点、入点和中间点。

11、3SAT (SATISFIABILITY WITH AT MOST 3 LITERALS PER CLAUSE) 析取式： $D_1 \sim D_r$ 文字集： $u_1 \sim u_m$ 及其非。

12、图着色数 (CHROMATIC NUMBER)

对图 G ，判断是否存在 k 染色，使相邻节点颜色相异。

11a12 不太一般的，假设 $3SAT$ 变元数小于 4。

$V=[u_1, u_2, \dots, u_n] (1 \leq i \leq n)$ $E=\{uv \mid u, v \in V\}$ 为图的两组边， u 和不同角标的文字及非连接，文字和自己所带的反角标。颜色数为 $r+1$ 。

13、分团覆盖 (CLIQUE NUMBER)

判断图 G 是否可分成不多于 k 个团。

12a13 染色之后每种颜色所成的集合在补图中成团。

14、恰好覆盖 (EXACT COVER)

判断是否存在子集族的一个子集是全集的分割。

14a15 HITTING SET 的集合为 EXACT COVER 的集合。HITTING SET 的集合 s 为 EXACT COVER 的集合 S 的集合所构成的集合。于是子集 W 在 EXACT COVER 中对应的子元的交集都有且仅有 1 个。

15、(HITTING SET)

对 U 的子集族 S ，是否存在 U 的子集 W ，使得 W 与 S 中的每个集合的交集规模均为 1。

16、斯坦纳树 (STEINER TREE)

对图 G, V 的子集 R, w 为 G 中的边的权重。求一个 G 的子树，包含 R 中所有点，且总权重不超过 k 。

14a16

给定图 G 和子集 R ，求树 T 。

转换之后， $V=[n_0] \cup [n_1] \cup [n_2] \cup [n_3] \cup [n_4]$ 集合方面，对 n_0 和 n_1 建立边集， n_2 和 n_3 中的两个点 a, b 建立边集， n_4 和 n_5 的权值是 $|S_4|$ ，其余地方权值都是 0。

17、三维匹配 (3-DIMENSIONAL MATCHING)

判断 $T \times T \times T$ 的子集 U 是否存在规模为 $|T|$ 的子集 W ，其中元素在三个维度上的投影均不相同。

14a17

不失一般性，我们假设好覆盖的子集族中每个子集的元素数至少为 2。（这个可以通过增加全集元素来使其划归，或者类似于分割元素。）

18、0-1 背包 (KNAPSACK)

判断对 $r+1$ 元整数组 $a_1 \sim a_r, b$ ，是否存在 0-1 量 x_i ，
s.t. $\sum a_i x_i = b$ 。

14a18

r 为子集族中子集个数， $a_{ij} = \text{if}(u_i \in S_j)$ $a_j = \sum a_{ij} d^i (i-1)$ $b = 1 + d + d^2 + \dots + d^{r-1}$ t 为全集元素数。

19、工作规划 (JOB SEQUENCING)

有 p 个工作，每个工作有它要做的时间、 ddl 、和迟到惩罚三个参数，判断是否存在顺序使得总惩罚不超过 K 。

18a19

0-1 背包就是极特殊的工作规划。代价和时间相同。 ddl 都是 b 。

20、(PARTITION)

判断 s 个正整数组成集合是否可以分成相等的两部分。

18a20

$s=2$ 时，这个数中包含原来背包问题的小重量，剩下两个盖用于配平。需要足够大，比如边配 $+1$ ，另一边配 -1 。

21、最大割 (MAX CUT) (可由 20 推导来)

给定图 G ，权重函数 w ，判断是否有不小于 W 的割。

$$|V|=s, w(i,j)=c_i * c_j W=\text{ceil}((\sum c_i)^2 / 4).$$

1. SAT or 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 中的各个析取范式 c_1, c_2, \dots, c_p 是由符号集 $\{x_1, x_2, \dots, x_n; \bar{x}_1, \bar{x}_2, \dots, \bar{x}_n\}$ 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

$$C_{ij} = \begin{cases} 1, & \text{if } x_j \in c_i \\ -1, & \text{if } \bar{x}_j \in c_i \\ 0, & \text{otherwise} \end{cases} \quad b_i = 1 - (c_i \text{ 中变项的个数}) \\ d_j = b_j + \Phi_i$$

输入多项式时间转化：

转化后的矩阵需要构造矩阵 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。

2. SAT or 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：

$N = \{\sigma_i \mid \sigma_i \in V\}$ 边集为：

$E = \{(\sigma_i, \sigma_j) \mid (\sigma_i, \sigma_j) \in V\}$ 且 $\sigma \neq \bar{\sigma}$

$k=p$

输入多项式时间转化：

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

对没有顶点 $\langle \sigma, i \rangle$ ，可能与之相连的顶点 $\langle \delta, j \rangle$ ，符号可能的取值有 $(2n-1)^2$ 种， j 可能的取值最多有 $(p-1)$ 种。故总共不会超过 $(p^n)^2 \cdot (2n-1)^2 \cdot (p-1)$ 种。

故转化的整体时间是 $O(p^n \cdot n^2)$ 。

3. CLIQUE or SET PACKING

问题回顾：

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

输入对应：

设原图中的节点为 $N=\{1, 2, 3, \dots, n\}$ ，每一个节点 i 对应 SET PACKING 问题中一个输入的集合 $\{i\} \mid i \in N, i \notin A\}$ ， A 是原图的边集。取 $k=|A|$ 。

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的条数，所以元素个数多至 $n(n-1)/2$ ，所以构造相应的输入需要的复杂度是 $O(n^2)$ 。

4. CLIQUE or NODE COVER

问题回顾：

NODE COVER：输入一张图 G 和一个正整数 k ，是否存在一个点集 $R \subseteq N$ 使得 $|R| \leq k$ ，且图 G 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 $k=|N|-k$

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) 所以转化为多项式时间的。

5. NODE COVER or SET COVERING

问题回顾：

SET COVERING：给定有限个集合组成的集族 $\{S_i\}$ ，和一个正整数 k 。判断是否存在 $\{S_i\}$ 的一个子集 $\{S_j\} \subseteq \{S_i\}$ ，使得 $U(S_k) = U(S_j)$ ，且 $|S_j| \leq k$

输入对应：

设 $\{1, 2, \dots, n\}$ 为图 G 中的顶点集，则取集合 S_j 为 NODE COVER 图 G 中与顶点 j 相关的边的并集作为 SET COVERING 问题的输入。且去 $k=1$ 。

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为 $2m$ ，所以输入转化的复杂度为 $O(n+2m)$

6. NODE COVER or FEEDBACK NODE SET

问题回顾：

FEEDBACK NODE SET：给定有向图 H 和正整数 k ，判断是否存在一个点集 $R \subseteq V$ ，使得 H 中一个环在 R 中有一个相关的点，且 $|R| \leq k$ 。

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 中的反向两向的两条边， $E = \{(u, v) \mid (u, v) \in V\}$

取 $k=1$ 。

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 $O(n+2m)$

7. NODE COVER or FEEDBACK ARC SET

问题回顾：

FEEDBACK ARC SET：给定有向图 H 和正整数 k ，判断是否存在一个边集 $S \subseteq E$ ，使得 H 中一个环在 S 中有一个相关的边，且 $|S| \leq k$ 。

输入对应：

将原图中的每个点分成出点和入点。 $V = N \times \{0, 1\}$

原图所有的边都转化成从出点到入点的两条边，所有的点的入点到出点的边。

即 $E = \{(u, 0) \mid u \in V\} \cup \{(0, 1) \mid u \in V\}$

取 $k=1$ 。

输入多项式时间转化：

有向图 H 中的顶点数为 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 $O(2n+n+2m) = O(3n+2m)$ 。

8. NODE COVER or 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：

该图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 k 。

定义向哈密顿图的顶点集为：(将原图中的边编号从 1 到 m)

$V = \{u_1, u_2, u_3, \dots, u_m\} \cup \{u_0\}$ u 是原图 G 中的顶点， i 是与 u 相连的边。

定义向哈密顿图的边集为：

$E = \{(u_i, u_{i+1}) \mid i \leq m\}$ $i \leq m$ 表示第 i 条边

$U(u_0, u_1, \dots, u_m) = \{u_1, u_2, \dots, u_m\} \cup \{u_0\}$ u 是编号 i 的边的端点， $i \leq m$ 表示第 i 条边

$U(u_1, u_2, \dots, u_{m-1}, u_0) = \{u_1, u_2, \dots, u_{m-1}, u_0\} \cup \{u_0, u_1\}$ u 是编号 i 的边的端点， $i \leq m$ 表示第 i 条边

$U(u_0, u_1, \dots, u_{m-1}, u_0) = \{u_0, u_1, \dots, u_{m-1}\}$ u 是编号 i 的边的端点， $i \leq m$ 表示第 i 条边

输入多项式时间转化：

该图有 n 个顶点，每条边有 2 个端点，所以每条边有 4 个点，所以总点数为 $4n$ 。

分析各条边的条数：每条边有 2 个端点，所以每条边有 2 个点，所以每条边有 2 条边。

第①类边：每条边只有一条边，所以每条边有 2 个点，所以每条边有 2 条边。

第②类边：每条边有两条边，所以每条边有 4 个点，所以每条边有 4 条边。

第③类边：每条边有三条边，所以每条边有 6 个点，所以每条边有 6 条边。

第④类边：每条边有四条边，所以每条边有 8 个点，所以每条边有 8 条边。

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 $2m+n$ 条边。

9. DIRECTED HAMILTON CIRCUIT or UNDIRECTED HAMILTON CIRCUIT

输入对应：

无向哈密顿图中的顶点集为：

$N = \{0, 1, 2, \dots, m\}$ 即将原图中的点分成对应的三个点。

边集定义为：

$A = \{(u_0, 0), (u_1, 1), (u_2, 2)\} \cup \{u, v \in V\}$

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 $2m+n$ 条边。

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

输入对应：

将 $3SAT$ 中的每一个语句做如下的转化：

$\sigma_1 \cup \sigma_2 \cup \dots \cup \sigma_m = (\sigma_1 \cup \sigma_2 \cup \dots \cup \sigma_m) \cup (\bar{\sigma}_1 \cup \bar{\sigma}_2 \cup \dots \cup \bar{\sigma}_m)$

一直每个子句的长度不超过 3 为止。

输入多项式时间转化：

将上面一长句为 m 经过一次转化，得到了 m 个语句，其中有 $(m-1)$ 个语句的长度是不超过 3 的，还有一个长度为 $(m-1)$ 的语句，第二次转化的时候就会将一条长语句为 $(m-1)$ 的语句进行转化。所以将一条长为 $(m-1)$ 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 <math

12. CHROMATIC NUMBER & CLIQUE COVER

问题回顾:
CLIQUE COVER: 属于一个图 G 和一个正整数 k , 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 考虑问题图 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。

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

$\text{NUAU}(\langle u, e, f \rangle \mid \text{顶点 } u \text{ 与边 } e \text{ 相关联}, 1 \leq f \leq k)$

用来覆盖这些元素的, 有两种形式:

$|I| = \{u, \langle u, e, f \rangle \mid e \text{ 与 } u \text{ 相关联}\}$, 对每一个 $1 \leq f \leq k$ 和每一个 $u \in N$ 对应这样一个集合
 $|I| = \{e, \langle u, e, f \rangle, \langle v, e, g \rangle \mid f \neq i, g \neq j, u \text{ 和 } v \text{ 是与 } e \text{ 相关联的两个点}\}$

输入多项式时间转化:

设原图的 n 个顶点和 m 条边

转化为需要覆盖的元素的个数为 $|N| + |A| + |\{ \langle u, e, f \rangle \mid f = 0 \}| = O(n + m + n(n - 1)k)$
第一类集合的个数为 $O(n(n - 1)k)$, 第二类集合的个数为 $O(2m(k - 1))$
所以转化的过程是多项式时间的。

14. EXACT COVER & HITTING SET

问题回顾:
EXACT COVER: 给定一些集合 S_i , 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 , 一个簇集 S_j , U_i 是簇集 S_j 的子集, 有一个 S_j 的子集 W 使得对任意的集合 U_i , $|W \cap U_i| = 1$ 。

输入对应:
对于恰好覆盖的 S_i 和 $\{u_i\}$, 转化成 HITTING SET 问题的输入 $U'_i = \{S_j \mid u_i \in S_j\}$, 簇集 S_j 的每个元素都是恰好覆盖包含的集合。

输入多项式时间转化:

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

15. EXACT COVER & STEINER TREE

问题回顾:
STEINER 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: $A = \{\langle n_0, S_j \rangle\} \cup \{\langle S_j, U_i \rangle \mid U_i \in S_j\}$

边的权重函数: $w(\langle n_0, S_j \rangle) = |S_j|$, $w(\langle S_j, U_i \rangle) = 0$, 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

16. EXACT COVER & 3-DIMENSIONAL MATCHING

问题回顾:

3-DIMENSIONAL MATCHING:

图 T 是一个有限集合, 给出一个集合 $U \subseteq T \times T \times T$, 找出 U 的一个子集 $W \subseteq U$, 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 $T = \{ \langle i_1, j_1, l_1 \rangle \mid i_1 \in S_j\}$, 今 α 是一个从 U_i 到自身的单射, 令 $\pi: T \rightarrow T$ 是使得对每个固定的 j , $\langle i_1, j_1 \rangle \mid i_1 \in S_j \rangle = \pi$ 的一个圆 (即轮换) 的函数, β 是一个与 i_1, j_1 有关的函数。

$U = \{ \langle \beta(i_1), \pi(j_1), l_1 \rangle \mid \text{对所有的 } i_1, j_1, l_1 \text{ 有 } i_1 \neq \pi(i_1) \}$

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 $|T| = \sum |S_j|$, 对 T 中一个元素 $\langle i_1, j_1 \rangle$, 它在 U 中第二个维度出现只有两种形式:

$\langle \beta(i_1), \pi(j_1), l_1 \rangle \text{ 和 } \langle \beta(\pi(i_1)), \pi(j_1), \pi(l_1) \rangle$

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 $O(np)$ 。

17. EXACT COVER & KNAPSACK

问题回顾:

KNAPSACK: 给出一些背包 $\{a_1, a_2, a_3, \dots, a_r\}$ 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 $a_j = \sum_{i \in S_j} a_i$, $b = \frac{d-1}{d-1}$

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 $O(mn)$ 的。

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:

从原问题成了新问题的输入为 $\langle 3p+1 \rangle$, 是多项式时间的。

19. KNAPSACK & PARTITION

问题回顾:

PARTITION: $(c_1, c_2, \dots, c_k) \in Z^k$, 是否存在 $I \subseteq \{1, 2, \dots, k\}$, 使得 $\sum_{i \in I} c_i = \sum_{i \notin I} c_i$

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

20. PARTITION & MAX CUT

问题回顾:

MAXCUT: 输入一个图 G , 邻接矩阵为 A , w 是 A 的权重函数, 给定一个正整数 W , 是否存在原顶点集的一个子集 S , 使得 $\sum_{u \in S, v \in S} w(u, v) \geq W$

输入对应:

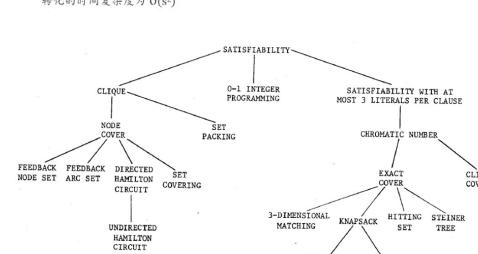
取 $N = \{1, 2, \dots, s\}$, $A = \{ \{i, j\} \mid i \neq j \}$

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

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

输入多项式时间转化:

在 MAXCUT 问题中有个 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 $O(s^2)$



课本作业:

Ex.6.1

最大子段和问题。设 s_j 是以 j 为结束位置的最大子段和, 则有 $s_j = \max(s_{j-1} + a_j, a_j)$, 于是整个串的最大子段和即为 $\max_{1 \leq j \leq n} s_j$ 。

Ex.6.2

设 p_i 表示停留在第 i 个旅馆时的总最小罚值, 其中 $p_0 = 0$ 。枚举上一个落脚点位置 j , 则有 $p_i = \min_{0 \leq j < i} (p_j + (200 - (a_i - a_j))^2)$, 于是 p_n 即为整个旅程的最小罚值。

Ex.6.3

设 r_j 表示在前 j 个位置开分店的最大总利润, $r_0 = 0$ 。若 j 为上一个满足 $m_i - m_j \geq k$ 的位置, 则根据是否在当前位置开分店可以分成两种情况, 取其中能获取较大利润的选择, 也即是 $r_j = \max(r_{j-1}, r_j + p_i)$, 最后返回 r_n 即可。

Ex.6.4

设 v_i 表示前 i 个字符组成的子串是否能被有效的分割, $v_0 = \text{true}$ 。若用 OR 表示连接求或, \wedge 表示逻辑与, $\text{substring}(m, n)$ 表示序号位于 $[m, n]$ 区间内的字符组成的子串, 则有 $v_i = \bigvee_{0 \leq j < i} (v_j \wedge \text{dict}(\text{substring}(j, i)))$ 。最后如要输出分割后的单词, 记录一下状态转移路径即可。

Ex.6.5

a) 设 n_i 为在某一列的前 i 行的格子里放置石子的 pattern 数目, 初始化

$n_0 = 1, n_1 = 2$ 。类似于 Ex.6.3, 根据是否在第 i 个位置放置石子分成两个情况,

并将这两种情况的 pattern 数相加可得 $n_i = n_{i-1} + n_{i-2}$ 。这正好就是 Fibonacci 数

列, 于是可得 $n_4 = 8$ 。

b) 设这 8 种 type 依次编号为 0-7, 并且已有函数 $c(i, t)$ 来判断 i, j 两种 type 是否

兼容。再设 $s(i, t)$ 为在前 i 列放置石子且最后一列的 type 为 t 时的最优值, $v(i, t)$

表示在第 i 列按照 t 样式放置石子时所得到的值。于是有 $s(i, t) = \max_{c(i, t) = \text{true}} (s(i-1, t) + v(i, t))$ 。最后返回 $\max_{0 \leq t \leq 7} (s(n, t))$ 即可。注意在这里因为石子的个数为 $2n$, 所以不需要考虑石子不够用的情况。

Ex.6.6

先建立好查询表 $m[3][3], n[3][3]$, 其中 $m[r][i][l] \cdot n[r][i][l] = r, (1 \leq i \leq 3)$ 。然后设

$b(i, j, t)$ 表示位于 $[i, j]$ 之间的子表达式相乘是否能得到 t 。则有状态转移方程:

$$b(i, j, t) = \bigvee_{i \leq k \leq j} \left(\bigvee_{l \leq t \leq 3} (OR(b(i, k, m[l][i][l]) \wedge b(k+1, j, n[l][i][l]))) \right)$$

式子好像挺复杂的, 其实思路还是很清楚的, 跟矩阵乘法类似。最后返回 $b(1, n, a)$ 即可。

Ex.6.7

定义 $p(i, j)$ 来表示在子串 $x[i, j]$ 是否为回文串。于是有以下关系式:

$$\begin{cases} \text{if } i \geq j : p(i, j) = \text{true} \\ \text{if } x[i] = x[j] : p(i, j) = p(i+1, j-1) \\ \text{otherwise} : p(i, j) = \text{false} \end{cases}$$

枚举所有子串, 按照上面的关系式依次检查其是否为回文串, 然后找出最长的回文子串即可。由于子串的个数为 $O(n^2)$, 所以整个算法的时间复杂度也为 $O(n^2)$ 。

Ex.6.8

注意这里要求子串连续, 所以与最长公共子序列略有不同。定义 $L(i, j)$ 为子串

$x[0, i]$ 和 $y[0, j]$ 的最长右对齐公共子串长度 (右对齐的意思是指这个公共子串是 $x[0, i]$ 和 $y[0, j]$ 的后缀)。于是有:

$$\begin{cases} \text{if } x[i] \neq y[j] : L(i, j) = 0 \\ \text{otherwise} : L(i, j) = L(i-1, j-1) + 1 \end{cases}$$

求得所有的 $L(i, j)$ 之后, 找出最长的一条即可。时间复杂度为 $O(mn)$ 。

Ex.6.9

首先将 m 个分割点按位置排序后存入数组 $d[1, \dots, m]$, 然后定义 $c(i, j, r, s)$ 为分割

子串 $s[i, j]$ 所需要的代价, 且其中所用的分割点为子数组 $d[r, s]$ 。于是有:

$$\begin{cases} c(i, j, r, s) = \min_{r \leq k \leq s} (l(i, j) + c(i, d[k], r, k-1) + c(d[k]+1, j, k+1, s)) \\ l(i, j) = j-i+1 \end{cases}$$

上式中的 $l(i, j)$ 为子串 $s[i, j]$ 的长度, 也即是该子串第一次被划分时所需要的代价。

最后返回 $c(1, n, 1, m)$ 即可。整个算法的时间复杂度为 $O(n^2m^2)$ 。

Ex.6.10

设 $h(i, j)$ 为掷出前 i 个硬币, 出现 j 个人头的概率。于是有:

$$\begin{cases} \text{if } j > i : h(i, j) = 0 \\ \text{else if } j = i : h(i, j) = p_i \cdot h(i-1, j-1) + (1-p_i) \cdot h(i-1, j) \\ \text{else} : h(i, j) = \max(L(i-1, j), L(i, j-1)) \end{cases}$$

最后返回 $L(n, k)$ 即可, 时间复杂度为 $O(nk)$ 。

Ex.6.11

定义 $L(i, j)$ 为子串 $x[0, i]$ 和 $y[0, j]$ 的最长公共子序列长度, 于是有:

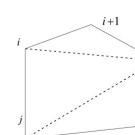
$$\begin{cases} \text{if } i = k = 0 : L(i, j) = 0 \\ \text{else if } x[i] = y[j] : L(i, j) = L(i-1, j-1) + 1 \\ \text{else} : L(i, j) = \max(L(i-1, j), L(i, j-1)) \end{cases}$$

最后返回 $L(n, m)$ 即可, 注意与 Ex.6.8 的区别。

Ex.6.12

首先按照提示进行编号, 再定义函数 $dist(i, j)$ 表示顶点 i , j 之间的距离, 对于子

问题 $A(i, j)$, 考虑边 $e(i, j)$, 显然, 这条边在最优分割后一定在某个三角形内, 如下图:



为了找出这个三角形, 枚举与 $e(i, j)$ 相对的顶点 $k \in e(i, j)$, 于是有以下关系式:

$$\begin{cases} \text{if } k = i+1 : A(i, j) = dist(i, j) + A(k, j) \\ \text{if } k = j+1 : A(i, j) = dist(i, k) + A(i, k) \\ \text{else} : A(i, j) = dist(i, k) + dist(k, j) + A(i, k) + A(k, j) \end{cases}$$

然后取其中代价最优的 k 值即可。整个算法的时间复杂度为 $O(n^2)$ 。

Ex.6.13

a) 比如 $\{1, 2, 10, 3\}$ 。

b) 定义 $r(i, j)$ 表示在子串 $x[i, j]$ 中进行游走时所获得的最大分值, 假设对手也

总是采取最优的策略, 于是有:

$$\begin{cases} r = s_0 + \min(x[i], r(i+1, j)) \\ r_j = s_j + \min(x(i+1, j), r(i, j-1)) \\ r(i, j) = \max(r, r_j) \end{cases}$$

其中 r 是玩家 1 第一步选择卡片 i 时所能得到的分值, r_j 则是选择卡片 j 时所能得到的分值。 $r(i, j)$ 当然是取 r_i 与 r_j 的较大的, 此外还需要记录下这一步的

选择 $c(i, j)$:

$$\begin{cases} \text{if } r > r_j : c(i, j) = \text{pick front} \\ \text{else if } c(i, j) = \text{pick back} \\ \text{else} : c(i, j) = \text{pick middle} \end{cases}$$

作为边界情况, 当 $i = j = 1$ 时, 虽然 $r(i, j) = \max(s_i, s_j)$ 。

Ex.6.14

每次沿着边沿切下一块布, 这样剩余的布料又可以形成最多三块矩形面料。设 $P(w, h)$ 是在宽为 w , 高为 h 布料上所能得到的收益 (显然, $P(w, h) = P(h, w)$)。

考虑每一次切割刀样的两种情况, 于是有:

$$\begin{cases} P(w, h, i) = c_i + P(a_i, h-b_i) + P(w-a_i, b_i) + P(w-a_i, h-b_i) \\ P(w, h, i) = c_i + P(b_i, h-a_i) + P(w-b_i, a_i) + P(w-b_i, h-a_i) \\ P(w, h) = \max \left(\sum_{i=0}^{t-1} (P(w, h, i) + P(w, h, i)) \right) \end{cases}$$

其中, $P(w, h, i)$ 是将第 i 个式样按照给定的形状 (即宽为 a_i , 高为 b_i) 直接切下

来的收益, 而

```

if (i == n) : p(i, j) = 1
if (j == n) : p(i, j) = 0
else : p(i, j) = p(i+1, j)/2 + p(i, j+1)/2

```

Ex.6.16

和 TSP 不同的地方在于本题目的起始点是 home 可以访问多次，而且不必访问所有的点。令 $B(S, i)$ 表示从 g 表示的 garage sale 集为 S ，结束位置为 j 的子问题上的最大收益值。注意 j 的前导点既可能是某个 garage sale，也可能是 home，于是有：

$$\begin{cases} B(S, i)_j = \max_{a \in S, j} (B(S - \{j\}, i) + v_j - d_a) \\ B(S, i)_h = \max_{a \in S} (B(S - \{j\}, i) + v_j - d_a - d_a) \\ B(S, i) = \max_{a \in S} (B(S, i)_j, B(S, i)_h, v_j - d_a) \end{cases}$$

最后找出最大的 $\max_{a \in S} (B(\{g\}, j) - d_a)$ 即可。当然，如果这个值小于 0，则返回 0。

Ex.6.17

设 $b(i)$ 表示值 i 是否能被找齐，于是有：

$$\begin{cases} if (i < 0) : b(i) = false \\ if (i = 0) : b(i) = true \\ else : b(i) = OR(b(i-x_1), b(i-x_2)) \end{cases}$$

最多只有 $O(n)$ 个状态，每次状态转移的代价为 $O(1)$ ，于是整个算法的时间复杂度为 $O(nv)$ 。

Ex.6.18

设 $b(i, j)$ 表示用前 i 个币种时是否能开面值 j ，考虑是否用第 i 个币种可以分成两种情况：如果使用，因为每个币种只能使用一次，有 $b(i, j) = b(i-1, j-x_i)$ ；

如果不使用，则 $b(i, j) = b(i-1, j)$ ，于是整个状态转移方程可表示为：

$$\begin{cases} b(i, j)_s = b(i-1, j-x_i) \\ b(i, j)_h = b(i-1, j) \\ b(i, j) = b(i, j)_s \vee b(i, j)_h \end{cases}$$

其中 \vee 表示逻辑或。最多只有 $O(nv)$ 个状态，状态转移代价为 $O(1)$ ，于是整个算法的时间复杂度为 $O(nv)$ 。

Ex.6.19

同 Ex.6.17，只不过在状态空间中增加一个维度来记录可以使用的钱币数 k ，即是有： $b(k, i, j) = OR(b(k-1, i, x_i))$ 。

Ex.6.20

设 $c(i, j)$ 为单词子集 $w[i:j]$ 所构成的最优查找树的查找代价，注意最优查找树的子树同样是最优查找树，于是有： $c(i, j) = \min\{c(i, k-1) + c(k+1, j)\} + \sum_{i \leq k < j} p_k$ 。

Ex.6.21

即是最大独立集的补集。

Ex.6.22

请参考 Ex.6.18。

Ex.6.23

设 $p(i, j)$ 为只考虑前 i 台机器，且总预算为 j 时的最大正常运行概率。再令

$$t(j) = \sum_{i=1}^j c_i, \text{ 于是有: } p(i, j) = \max_{k \in [i, j], t(i) \leq k \leq t(j)} (p(i-1, k-c_i)(1-(1-t)^i)).$$

Ex.6.24

a) 观察书中 Figure 6.4，由于计算某一个方块的只需要知道该方块的左、上、左上方三个方向上的相邻方块的值，于是在每一时刻只需要存储当前行和上一行的状态就足以足够，为方便可以用滚动数组来存储，空间复杂度为 $O(n)$ 。

b) 为了与书中的代码保持一致，先将 m, n 互换一下，即还是用 m 表示行，而 n 表示列。用一个 $n/2+1$ 列（因为只需要考虑 $j > n/2$ 的情况）的滚动数组 K 来存储当前行和上一行的 k 值，然后在 edit distance 算法中加入以下伪代码：

```

K(1, 1) = 1
if j > n/2
    if E(1, j) == E(1-1, j) + 1
        K(i, j-n/2) = K(i-1, j-n/2)
    else if E(1, j) == E(1-1, j-1) + 1
        K(i, j-n/2) = K(i-1, j-n/2-1)
    else if E(1, j) == E(1-1, j-1) + diff(i, j)
        K(i, j-n/2) = K(i-1, j-n/2-1)

```

c) 由递归关系容易看出，这个算法的时间复杂度为：

$$\begin{aligned} T(m, n) &= O(mn) + \frac{1}{2}O(mn) + \frac{1}{4}O(mn) + \frac{1}{8}O(mn) \dots \\ &= O(mn) \end{aligned}$$

这是个典型的时间换空间的算法，虽然时间复杂度还是为 $O(mn)$ ，但是常数因子却增大了一倍，算法实现的难度也大大提高了。

Ex.6.25

设 $b(i, c_1, c_2, c_3)$ 表示前 i 个元素是否能被分成三堆，且这三堆的分别为 c_1, c_2, c_3

(还可以限制 $c_1 \leq c_2 \leq c_3$ 以缩减状态空间大小)。再令 s_i 表示 $\sum_{j=1}^i a_j$ 。于是有以下状态转移关系：

$$\begin{aligned} b(i, c_1, c_2, c_3) &= b(i-1, c_1-a_1, c_2, c_3) \\ &\quad \vee b(i-1, c_1, c_2-a_2, c_3) \\ &\quad \vee b(i-1, c_1, c_2, c_3-a_3) \end{aligned}$$

其中 \vee 表示逻辑或，最后返回 $b\left(n, \frac{s_n}{3}, \frac{s_n}{3}, \frac{s_n}{3}\right)$ 即可，时间复杂度为 $O(ns^3)$ 。

Ex.6.26

类似于最短编辑距离，有：

$$S(i, j) = \max \begin{cases} S(i-1, j) + \delta(x[i], -) \\ S(i, j-1) + \delta(-, y[j]) \\ S(i-1, j-1) + \delta(x[i], y[j]) \end{cases}$$

Ex.6.27

重定义子问题为 $S(i, j, k)$ ，其中 $k \in \{0, 1, 2\}$ ，分别对应在最末位置上的这三种匹配：

$x[i] \leftrightarrow y[j], - \leftrightarrow y[j], x[i] \leftrightarrow -, \text{ 于是有:}$

$$s(i, j, 0) = \max_{0 \leq k < 3} (s(i-1, j-1, k)) + \delta(x[i], y[j])$$

$$s(i, j, 1) = \max \begin{cases} s(i, j-1, 0) - (c_0 + c_1) \\ s(i, j-1, 1) - c_1 \\ s(i, j-1, 2) - (c_0 + c_1) \end{cases} + \delta(-, y[j])$$

$$s(i, j, 2) = \max \begin{cases} s(i-1, j, 0) - (c_0 + c_1) \\ s(i-1, j, 1) - (c_0 + c_1) \\ s(i-1, j, 2) - c_1 \end{cases} + \delta(x[i], -)$$

Ex.6.28

结合 Ex.6.1 和 Ex.6.26，有：

$$S(i, j) = \max \begin{cases} \max(S(i-1, j) + \delta(x[i], -), \delta(x[i], -)) \\ \max(S(i, j-1) + \delta(-, y[j]), \delta(-, y[j])) \\ \max(S(i-1, j-1) + \delta(x[i], y[j]), \delta(x[i], y[j])) \end{cases}$$

最后找到 $\max_{1 \leq i \leq n, 1 \leq j \leq m} (S(i, j))$ 即可。

Ex.6.29

设 $V[j]$ 表示在只考虑 $x[1:j]$ 的子问题上的最大权值总和。容易看出，一定存在

$x[1:j]$ 的某个后缀 $x[i:j]$ ，使得 $V[j]$ 为以下三个值之一： $V[i-1]$ 、

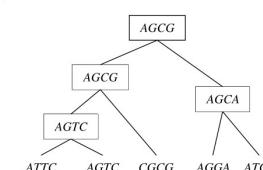
$V[i-1] + w(i, j) \cdot w(i, j)$ 。于是可得以下关系：

$$V[j] = \max_{1 \leq i \leq j} (\max(V[i-1], V[i-1] + w(i, j), w(i, j)))$$

其中 $w(i, j)$ 即是三元组 (i, j, w) 对应的 w 值。算法的时间复杂度为 $O(n^2)$ 。

Ex.6.30

a) 如下图：



设 $T(m, n)$ 为只考虑前 i 台机器，且总预算为 j 时的最大正常运行概率。再令

$$T(m, n) = O(mn) + \frac{1}{2}O(mn) + \frac{1}{4}O(mn) + \frac{1}{8}O(mn) \dots = O(mn)$$

这是一个典型的时间换空间的算法，虽然时间复杂度还是为 $O(mn)$ ，但是常数因子却增大了一倍，算法实现的难度也大大提高了。

b) 根据提示，每次只考虑一位。定义状态 $S(T, c)$ ，其中 $c \in \{A, C, G, T\}$ ，来表示

问题域为树 T ，且 T 的根节点为 c 时的最小分值。于是有：

$$\delta(i, j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

$L = left-subtree(T)$

$R = right-subtree(T)$

$$S(T, c) = \min_{a \in A, b \in B} (S(L, a) + S(R, b) + \delta(c, a) + \delta(c, b))$$

由于每个内结点都有两个儿子，于是可得内结点的数目为叶子节点的个数减一，

也即是说有状态数 $s = O(n)$ ，又因为状态转移的代价为 $O(1)$ ，所以整个算法的

时间复杂度为 $O(nk)$ 。

Ex.7.6

比如下面这个 LP：

$$\begin{aligned} \text{maximize } & 2y - x \\ \text{subject to } & x \geq y \\ & y \leq 100 \\ & x, y \geq 0 \end{aligned}$$

当 $x = 100, y = 100$ 时取得最大值 100。

Ex.7.7

a) 对于任意 a, b ，该 LP 总是可行的。

b) $ab = 0$ 。

c) $ab \neq 0, a \neq b$ 。

Ex.7.8

除 a, b, c 外，再增加 8 个变量， $e_i, 1 \leq i \leq 7$ 和 me ，分别表示在每个采样点的绝对误差和总的绝对误差。于是可得到 LP 如下：

$$\begin{aligned} \text{minimize } & me \\ \text{for } i = 1:7 & e_i \geq ax_i + by_i - c \\ & e_i \geq -(ax_i + by_i - c) \\ \text{me } & \geq e_i \end{aligned}$$

Ex.7.9

比如下面这个简单问题：

$$\begin{aligned} \text{maximize } & xy \\ \text{subject to } & x + y \leq 10 \\ & x, y \geq 0 \end{aligned}$$

根据算法几何平均值不等式知当 $x = y = 5$ 时， xy 取得最大值 25。

Ex.7.10

最大流为 13，相应的最小割为 $\{(S, C, F), \{A, B, D, E, G, T\}\}$ 。

Ex.7.11

与之对偶的 LP 如下：

$$\begin{aligned} \text{minimize } & 3s + 5t \\ \text{subject to } & 2s + t \geq 1 \\ & s + 3t \geq 1 \\ & s, t \geq 0 \end{aligned}$$

最优值为 2.2，此时 $x = 0.8, y = 1.4$ （原问题）或者 $s = 0.4, t = 0.2$ （对偶问题）。

Ex.7.12

$\boxed{1} + \boxed{2} \times 1/2 \rightarrow x_1 - x_3 \leq 1.5$ ，又因 $x_3 \geq 0$ ，于是 $x_1 - 2x_3 \leq x_1 - x_3 / 2 \leq 1.5$ 。

Ex.7.13

a) 赢。

b) 设 R 和 C 的策略分别为 $\{x_1, x_2\}$ 和 $\{y_1, y_2\}$ ，其中 x_1, y_1 为出人头的概率， x_2, y_2 为出人脚的概率。于是可得以下 LP 及其对偶形式：

$$\begin{array}{ll} \text{max } z & \text{min } w \\ z \leq x_1 - x_2 & w \geq y_1 - y_2 \\ z \leq -x_1 + x_2 & w \geq -y_1 + y_2 \\ x_1 + x_2 = 1 & y_1 + y_2 = 1 \\ x_1, x_2 \geq 0 & y_1, y_2 \geq 0 \end{array}$$

不难看出，该博弈的 value 为 0，最佳策略 $\{x_1, x_2\} = \{y_1, y_2\} = \left\{\frac{1}{2}, \frac{1}{2}\right\}$ 。

Ex.7.14

设 Joey 和 Tony 的策略分别为 $\{x_1, x_2\}$ ， $\{y_1, y_2, y_3\}$ ，有 LPs：

$$\begin{array}{ll} \text{max } z & \text{min } w \\ z \leq 2x_1 - x_2 & w \geq 2y_1 - 3y_3 \\ z \leq -2x_1 & w \geq -y_1 - 2y_2 + y_3 \\ z \leq -3x_1 + x_2 & y_1 + y_2 + y_3 = 1 \\ x_1 + x_2 = 1 & y_1, y_2, y_3 \geq 0 \end{array}$$

解得该博弈的 value 为 -1，最佳策略 $\{x_1, x_2\} = \left\{\frac{1}{2}, \frac{1}{2}\right\}$ ， $\{y_1, y_2, y_3\} = \left\{0, \frac{2}{3}, \frac{1}{3}\right\}$ 。

Ex.7.17

- a) 最大流为 11，相应的最小割为 $(\{S, A, B\}, \{C, D, T\})$ 。
 b) 图略， S 可达的点集为 $\{A, B\}$ ， T 可达的点集为 $\{S, A, B, C, D\}$ 。
 c) 瓶颈边有： $e(A, C)$, $e(B, C)$ 。
 d) 略。
 e) 设源图为 $G(V, E)$ ，余图为 $R(V, F)$ ，那么一条瓶颈边 $e(i, j)$ 具有以下性质：

$e(i, j) \in E$ ，同时在余图 R 中， $S \rightarrow i$ 且 $j \rightarrow T$ ($S \rightarrow i$ 表示存在一条从顶点 S 到 i 的路径， $j \rightarrow T$ 同)。于是，在 R 中添加 $e(i, j)$ 后就形成了一条增量路径，从而增加了最大流。根据这些性质，可以在线性时间内找到所有的瓶颈边，步骤依次如下：先在余图 R 中，以 S 为源点 DFS 找到 S 可达的顶点集 I ；再在 R 的反图 R^T 中以 T 为源点进行 DFS，得到可达 T 的顶点集 J ；最后遍历集 E ，对于某条边 $e(i, j)$ ，如果 $i \in I$, $j \in J$ ，则说明 $e(i, j)$ 是瓶颈边。

Ex.7.18

- a) 添加一个新顶点 S 连接到原来所有的源，再添加一个新顶点 T ，并将原来的汇都指向 T ，然后求 S 到 T 的最大流即可。当然，这些新添加的边的容量必须足够大，可以设成 INFINITE。
 b) 把每个顶点 v 拆成两个顶点 v_1 , v_2 ，添加一条边 $e(v_1, v_2)$ ，并令 $e(v_1, v_2)$ 的容量为原来该顶点的容量。然后将原图中以 v 为终点的边改成以 v_1 为终点，以 v 为起点的边改成以 v_2 为起点。这样处理后，就成了一个平凡的最大流问题了。
 c) 在 LP 模型中，对每条边多加一个约束。
 d) 修改一下流量守恒的约束即可。

Ex.7.19

首先由给定的最大流解可以很容易的得到余图 $R(V, F)$ ，然后在 R 中以 S 为源点进行 DFS；如果存在着向路径 $S \rightarrow T$ ，根据增量路径定理可知，一定还可以构造出更大的流；否则说明该解的确是最终的。

Ex.7.20

为一条边关联 k 个变量 $f_e^{(1)}, f_e^{(2)}, \dots, f_e^{(k)}$ ，其中 $f_e^{(i)}$ 表示从源 s_i 出发，在边 e 内的流量。那么流 $f^{(i)}$ 的大小即是 s_i 出发，且在 t_i 汇集的各条支流的总和，也即是 $f^{(i)} = \sum_{(s_i, t_i) \in E} f_{(s_i, t_i)}^{(i)}$ 。于是可以建立 LP：

$$\begin{aligned} & \max \sum_e f_e \\ & \forall e \in E: f_e^{(1)} + f_e^{(2)} + \dots + f_e^{(k)} \leq c_e \\ & \forall v \in V: \sum_{e \in E} f_e^{(i)} = \sum_{e \in E} f_e^{(j)} \\ & f^{(i)} = \sum_{(s_i, t_i) \in E} f_{(s_i, t_i)}^{(i)} \geq d_i \\ & f_e^{(i)} \geq 0 \end{aligned}$$

Ex.7.21

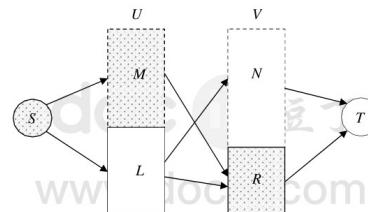
首先先求得最大流，再构造出余图 $R(V, F)$ 。显然，所谓的“关键边” $e(i, j)$ 的容量肯定已被占满了，这意味着在余图 R 中，顶点对 (i, j) 只存在反向边，即 $e(j, i) \in F$, $e(i, j) \notin F$ 。当然，在余图 R 只存在反向边的顶点对不一定都是关键边的顶点，所以需要将这些不是关键边的顶点对剔除掉，方法如下：在 R 中以 S 为源点进行 DFS 搜索，设所经过的反向边集为 E' ，那么对于所有 $e'(j, i) \in E'$, (i, j) 肯定不是关键边 (Why? 自己画图就知道了)。于是，如果在余图 R 中，所有只存在反向边的顶点对集为 L ，在 DFS 后得到的边集 E' 对应的顶点对集为 U ，那么 $C = L - U$ 即是所求的关键边集。

Ex.7.22

参照 Ex.7.21 可知，在最大流已知的情况下，可以在线性时间内判断某条边是否是关键边。所以，如果 $e(u, v)$ 不是关键边，则修正后的最大流 $f' = f$ ，否则 $f' = f - 1$ 。

Ex.7.23

考虑一个二分图 $G(U, E, V)$ ，其中顶点集 U 、 V 互不相交。设图 G 的最小顶点覆盖为 C ，令 $L = U \cap C$, $R = V \cap C$, $M = U - L$, $N = V - R$ 。添加一个源 S 和汇 T 后，可得流网络如下：



在上图中，因为 $L \cup R$ 是最小顶点覆盖，所以从 M 到 N 不存在边。现在观察在上图中用阴影标记出来部分，它与剩余的部分形成了一个割 $(S \cup M \cup R, L \cup N \cup T)$ 。容易看出这个割的割边数目正好就是最小顶点覆盖的顶点数。于是求最小顶点覆盖 $L \cup R$ 也就转化成了求最小割 $(S \cup M \cup R, L \cup N \cup T)$ 。

另外，从这里也可以看出，在二分图中，最小顶点覆盖的顶点数等于最大匹配的边数，因为这两者的流模型是完全一样的。PS. 也可以从另外一个方面来证明：设最小顶点覆盖 C 的顶点数为 m ，最大匹配的边数为 n ，首先由匹配的性质易知 $m \geq n$ 。再回到上面的图，可以发现：对于 L 中任意 k 个顶点，都至少与 N 中的 k 个顶点相连。否则就可以在 N 中找出小于 k 个顶点，替换掉 L 中的 k 个顶点，从而得到一个更小的顶点覆盖。同理，在 R 中的任意 k 个顶点也至少与 M 中的 k 个顶点相连结。由 Hall 定理，我们知道这意味着一个大小为 m 的顶点覆盖必然可以对应一个大小为 m 的匹配，所以又有 $n \geq m$ 。于是可知 $m = n$ ，得证。

Ex.7.24

- a) 比如路径 $D \rightarrow F \rightarrow B \rightarrow E \rightarrow A \rightarrow H \rightarrow C \rightarrow I$ 。
 b) 如果存在交互路径 P ，设在 P 中，属于 M 的边集为 P_M ，不属于 M 的边集为

P_{NM} ，由交互路径的定义知， P_{NM} 中的边刚好比在 P_M 中的边多一条，也即是 $|P_{NM}| - |P_M| = 1$ 。于是在 M 中，将 P_M 替换为 P_{NM} 就可以构造一个比原来多一个边的匹配。这就说明了在最大匹配中不可能存在交互路径，但这只是证明了命题的一个方向，另外还需要证明的是：如果匹配 M 不存在交互路径，它一定就是最大匹配？注意到交互路径其实是与最大流模型中的增量路径等价的：设匹配 M 在流模型中的余图为 R ，若 M 不存在交互路径，那么在 R 中也就不存在增量路径，从而得知 M 一定是最匹配。

c) 首先，为保证交互路径的交互性质，需要将所有的边加上方向。对于任意边 $e(i, j) \in E$ ，如果 $e(i, j) \notin M$ ，则规定方向为 $i \rightarrow j$ ，否则规定方向为 $j \rightarrow i$ (这样处理后得到的图其实就是在流模型中匹配 M 对应的余图)。然后从 V_1 中未被

M 覆盖的顶点集出发进行 BFS，若访问到 V_2 中的某个未被覆盖的顶点，即找到了一条交互路径。

d) 迭代的进行交互路径的搜索，每次找到一条交互路径都将使 M 的匹配数加 1。

由于 $|M| \leq \min\left(\frac{|V|}{2}, |E|\right)$ ，所以最多只需迭代 $\min\left(\frac{|V|}{2}, |E|\right)$ 次。由于每次迭代需时 $O(|V| + |E|)$ ，所以总时间复杂度为 $O(|V| \cdot |E|)$ 。

Ex.7.25

- a) 由给定流网络可得 LP 及与其对偶的 LP 如下：

$$\begin{array}{ll} \max f_{SA} + f_{SB} & \min y_1 + 3y_2 + y_3 + 2y_4 + y_5 \\ f_{SA}, f_{AB}, f_{BT} \leq 1 & y_1 + y_6 \geq 1 \\ f_{AT} \leq 2 & y_2 + y_7 \geq 1 \\ f_{SB} \leq 3 & y_3 - y_6 + y_7 \geq 0 \\ f_{SA} - f_{AB} - f_{AT} = 0 & y_4 - y_6 \geq 0 \\ f_{SB} - f_{AB} - f_{BT} = 0 & y_5 - y_7 \geq 0 \\ f_{SA}, f_{SB}, f_{AB}, f_{AT}, f_{BT} \geq 0 & y_1, y_2, y_3, y_4, y_5, y_6, y_7 \geq 0 \end{array}$$

b) 如上，用 lingo 可解得两者的最优值都为 2。

c) 先把答案写出来，至于为什么是这样的形式，看了后两问就知道了：

$$\begin{aligned} & \min \sum_{e \in E} y_e c_e \\ & \forall e(s, u) \in E: y_e + y_u \geq 1 \\ & \forall e(u, t) \in E: y_e - y_u \geq 0 \\ & \forall e(u, v) \in E, u \neq s, v \neq t: y_e - y_u + y_v \geq 0 \end{aligned}$$

d) 若一条路径为 $s \rightarrow u_1 \rightarrow u_2 \rightarrow u_3 \rightarrow \dots \rightarrow u_n \rightarrow t$ ，由上面的 LP 可得：

$$\begin{aligned} & y_{(s, u_1)} + y_{u_1} \geq 1 \\ & y_{(u_1, u_2)} - y_{u_1} + y_{u_2} \geq 0 \\ & y_{(u_2, u_3)} - y_{u_2} + y_{u_3} \geq 0 \\ & \vdots \\ & y_{(u_{n-1}, t)} - y_{u_{n-1}} \geq 0 \end{aligned}$$

将上面所有不等式相加即得 $y_{(s, u_1)} + y_{(u_1, u_2)} + y_{(u_2, u_3)} + \dots + y_{(u_{n-1}, t)} \geq n$ 。

e) 设流网格的一个割为 (L, R) ，其中 $s \in L$, $t \in R$ ，那么这些变量的意义可以是这样：对于边 e ，若 $y_e = 1$ ，那么表示 e 从 L 到 R 的一条边；对于顶点 u ， $y_u = 1$ 即表示 $u \in L$ ，而 $y_u = 0$ 表示 $u \in R$ 。容易观察到，当这些变量被赋予这样的意义后，的确是与对偶 LP 中的约束关系相对应的，而目标函数恰好就是求最小割。

Ex.7.26

a) 首先需要明确的是，由于各个约束条件都为线性，所以实际上每个约束条件都对应着一个半空间，这些半空间并起来形成了最后的可行域，于是可知这个可行域必然是非凹的。这也表明了，如果 $\{x\}$ 、 $\{y\}$ 都是可行解，那么在它们之间的连线上任意一点 $\{z\} = \omega\{x\} + (1-\omega)\{y\}$ 也是可行解。现在设在可满足系统 S 中的 not forced-equal 不等式为 $ie_1, ie_2, ie_3, \dots, ie_4$ ，由 not forced-equal 不等式的定义知必然存在一个解 $\{x_i\}$ 使得 ie_i 不取等号，同样也存在 $\{x_1\}, \{x_2\}, \dots, \{x_n\}$ 分别使得 ie_2, ie_3, \dots, ie_4 不取等号。于是我们可以取 $\{x_a\}$ 为 $\{x_1\}, \{x_2\}, \dots, \{x_n\}$ 的算术平均，由前面结论知 $\{x_a\}$ 也肯定为 S 的一个解，也容易看出 $\{x_a\}$ 可以使得 ie_1, ie_2, \dots, ie_n 都不取等号。于是得证。

b) 分别在第 i 个不等式左边加上一个变量 b_i (且有 $b_i \geq 0$)，然后以目标函数 $\max \sum b_i$ 建立 LP。由于这个 LP 可能是无边界的，可以再加上约束条件 $b_i \leq c_i$ ，其中 c_i 是某个正常数。如果在最优解中， $b_i = 0$ ，那么说明第 i 个不等式是 forced-equal 的，否则为 not forced-equal (待证明)。

Ex.7.27

设而值为 x_i 的硬币的使用数量为 c_i ，于是可得整数线性规划问题如下：

$$\begin{aligned} & \min v - \sum x_i c_i \\ & \sum x_i c_i \leq v \\ & c_i \geq 0 \end{aligned}$$

当然不能用一般的 LP 模型来求解，那样很有可能只能得到没有意义的分数解。

Ex.7.28

a) 设最短路径长度为 s ，于是只需证明 $s \leq \sum_e f_e$ ，且在某种情况下可以取得等号即可。我们知道， f 可以分解成一条或者多条路径 $p_1, p_2, p_3, \dots, p_n$ (见 Ex.7.31.c)。由于 $\text{size}(f) = 1$ ，这意味着 $\sum_e f_e$ 不小于这些路径的长度的某种加权平均，再由于这些路径的长度不小于 s ，从而推出 $s \leq \sum_e f_e$ 。另外，若取最短路径为流路径，此时 $s = \sum_e f_e$ ，这说明了上面的不等式可以取得等号。

b) 比划着写就行，如下：

$$\begin{aligned} & \min \sum_e f_e \\ & \sum_{(s, u) \in E} f_{(s, u)} = 1 \\ & \sum_{(v, t) \in E} f_{(v, t)} = 1 \\ & \forall u \in V, u \neq s, u \neq t: \\ & \quad \sum_{(w, u) \in E} f_{(w, u)} - \sum_{(u, v) \in E} f_{(u, v)} = 0 \\ & \forall e \in E: f_e \geq 0 \end{aligned}$$

c) 注意在上面的 LP 中，除最后的 $f_e \geq 0$ 之外，每个顶点各对应一个约束条件，共 $|V|$ 个约束条件。对于每个顶点 $v \in E$ ，将该顶点对应的约束条件乘上一个系数 x_v ，最后再把所有的约束方程加起来可得 $\sum_{(w, u) \in E} (x_w - x_u) f_{(w, u)} = x_s - s$ ，联系到上面 LP 中的目标函数 $\min \sum_e f_e$ ，容易看出其对偶 LP 即为题中所示。

d) 本题是求有向图的最短路，而前面是求无向图的最短路。

Ex.7.29

a) 用 x_i 来表示是否雇佣演员 i ， y_j 表示是否得到老板 j 的投资，于是有 ILP：

$$\begin{aligned} & \max \sum_j y_j p_j - \sum_i x_i s_i \\ & \text{for } j \in [1, n] \\ & \text{for } i \in L \\ & y_j \leq x_i \\ & x_i, y_j \in \{0, 1\} \end{aligned}$$

b) 在这里只需要说明，对于任意一个非整数解，都存在比它更优的整数解。由于 y_i 是由 x_i 决定的 (最优情况下)，因此可以只考虑 x_i 即可。首先，若非整数解 x_i 均小于 1，若此时的收益值为负，那么直接令 $x_i = 0$ 就能得到一组更优的整数解。否则若此时的收益为正，设 $r = \max x_i$ ，显然 $0 < r < 1$ 。现在我们把每个 x_i 乘上一个放大系数 $\frac{1}{r}$ ，可以得到另一组解 x'_i ，注意在这个新解中，有部分元素变成了 1。容易看出 x'_i 对应的收益值也是 x_i 对应的收益值的 $\frac{1}{r}$ 倍，因为此时的 y_i 也可以提高为原来的 $\frac{1}{r}$ 倍。所以 x'_i 比 x_i 更优。现在来考虑 x_i 有部分元素为 1 的情况，设 $f = \max x_i$ ，那么将 x_i 中小于 1 的所有元素都乘于放大系数 $\frac{1}{f}$ ，或者将 x_i 中小于 1 的所有元素都设为 0，二者取其一，也必能得到一个更优的解 x'_i (这里就不再详细推导了)。通过前面的叙述可知，对于任意一个非整数解，经过上述若干次的变换之后，最终必然能得到一个更优的整数解。

Ex.7.30

该条件的必要性很显然：如果存在着完美匹配，那么任意一个男生子集 S 至少与 $|S|$ 个女生有一腿，否则必然出现一女伺候几男。现在来证明充分性，即如果任意一个男生子集 S 都至少与 $|S|$ 个不同的女生相连结，那么一定存在完美匹配。考虑该二分图对应的流模型的任意一个割 (L, R) ，其中 $s \in L$, $t \in R$ 。假设在 L 中，有 b 个男生， g 个女生，于是在 R 中即有 $n-b$ 个男生， $n-g$ 个女生。于是， s 与 R 中的 $n-b$ 个男生都相连结， L 中的 g 个女生都与 t 相连。此外， L 中的 b 个男生总共至少与 b 个女生相连，扣掉在 L 中的 g 个女生，于是这 b 个男生至少与 R 中的 $b-g$ 个女生相连。将上面的连结边总和起来，可得这个割 (L, R) 之间的前向边数至少为 $(n-b) + g + (b-g) = n$ 。既然这个割是任意的，这说明了任意割都至少为 n 。由最大流最小割定理可知该模型的最大流即为 n ，也就是说一定存在完美匹配。

Ex.7.31

a) Ford-Fulkerson 算法即是通过在余图中搜寻增量路径来求最大流（好像书中也没提到这个算法的名字）。如果第一次找到的增量路径为 $A \rightarrow B \rightarrow T$, 接下来再是 $B \rightarrow A \rightarrow T$, 这样每次只找到一条为 1 的增量路径，总共需要迭代 2000 次。
 b) 类似于 Ex.4.13，在这里可以使用 Dijkstra 算法依赖于以下事实，如果一条路径 $s \rightarrow \dots \rightarrow r \rightarrow t$ 的容量为 c ，那么路径 $s \rightarrow \dots \rightarrow r$ 的容量必定大于等于 c 。现在来修改 Dijkstra 算法，设 $cp(v)$ 表从 s 到 v 的所有路径的最大容量值。首先将所有的 $cp(v)$ 初始化为 0，然后修改 update 过程如下：

```
for all edges  $(u,v) \in E$ :
  if  $cp(v) < \min(cp(u), c(u,v))$ 
     $cp(v) = \min(cp(u), c(u,v))$ 
  ...
  ...
```

c) 考虑一个最小割，容易看出每条割边 e 都恰好对应着一条路径 p_e ，使得 p_e 经过边 e ，且有 p_e 上的流等于 $c(e)$ 。这些 p_e 汇总起来就成了最大流。因为割边的数量不会超过 $|E|$ ，于是可知，最大流可以由不超过 $|E|$ 条路径汇总而成。

d) 既然最大流 F 可以由不超过 $|E|$ 条路径汇总而成，那么“最肥”的那条路径的流一定不小于 $\frac{F}{|E|}$ ，令 c_i 为第 i 次迭代后的余图的最大流，于是有关系式：

$$c_{i+1} \leq c_i - \frac{c_i}{|E|}$$

参考书中 5.4 节，可知迭代次数为 $O(|E| \cdot \log F)$ 。

8 NP-complete problems

Ex.8.1

若 S 为图 G 中所有边的长度之和，显然所求最短回路的长度不会超过 S 。倘若 TSP 能在多项式时间内解决，那么通过在区间 $[0, S]$ 内二分的进行 TSP 询问，从而也就能在多项式时间内解决 TSP-OPT。

Ex.8.2

如果图 G 中存在哈密顿回路（亦即是 Rudrata path），那么对于图 G 中的每一条边 e ，询问 G 除边 e 之后是否还存在哈密顿回路。如果答案是“否”，说明 e 正好是回路中的一条边。如果答案是“是”，则将 e 从图 G 中去掉。在上述过程完成之后，图 G 中剩余的边即是所求的回路。显然，若每询问都是多项式时间的，那么整个过程可以在多项式时间内完成。

Ex.8.3

首先，易知 STINGY SAT 的解是可在多项式时间内验证的，因此属于 NP。另外，很容易将其 SAT 归约到 STINGY SAT（将 k 设为所有变量的总个数即可），于是可知 STINGY SAT 为 NP 完全问题。

Ex.8.4

a) 显然 CLIQUE-3 的解是可以多项式时间内验证的，于是属于 NP。
 b) 归约的方向反了，应该是 CLIQUE \rightarrow CLIQUE-3，而不是 CLIQUE-3 \rightarrow CLIQUE。
 c) 注意求出 G 的团之后只能得到 \bar{G} 的顶点覆盖，而非 G 的顶点覆盖。
 d) 若图 G 中所有的顶点的度不大于 3，那么可知在图 G 的最大团中不会超过 4 个顶点。于是，直接枚举所有的顶点四元组即可求解，时间复杂度为 $O(|V|^4)$ 。

Ex.8.5

3D MATCHING \rightarrow SAT：首先，对于任意两个互相冲突的三元组 t_i, t_j ，最多只能选择其中一个，即 $(\bar{t}_i \vee \bar{t}_j)$ 。然后对于任意一个顶点 v ，假设与该顶点相连的三元组为 $t_1^i, t_2^i, \dots, t_n^i$ ，那么至少要选择其中一个，即 $(\bar{t}_i^1 \vee \bar{t}_i^2 \vee \dots \vee \bar{t}_i^n)$ 。最后将这些 clauses 组合起来即构成一个 SAT 问题。这个转换过程显然是多项式的。

RUDRATA CYCLE \rightarrow SAT：假设顶点的总数为 n ，对于任意顶点 i ，设它的邻接点为 $p_1^i, p_2^i, \dots, p_m^i$ 。首先，顶点 i 必在哈密顿回路的某个位置上，即 $(x_1 \vee x_2 \vee \dots \vee x_n)$ 。然后，顶点 i 在哈密顿回路中不能出现多次，于是对于任意 $r \neq s$ ，有 $(\bar{x}_r \vee \bar{x}_s)$ 。再次，在顶点 i 的邻接点中必然有某个顶点是它的后继。于是对于任意位置 j ，设该位置的后继位置为 $k = j + 1 \bmod n$ ，如果顶点 i 位于位置 j ，那么它的邻接点中必有顶点位于位置 k ，即 $(\bar{x}_j \vee x_{p_1^k} \vee x_{p_2^k} \vee \dots \vee x_{p_m^k})$ 。最后将这些 clauses 组合起来即构成一个 SAT 问题，这个转换过程当然也是多项式的。

Ex.8.6

- a) 若每个 literal 最多出现一次，那么任意变量 v 出现的方式大致可以分为以下五种：
 1. 只在子句 c 中出现 v ，此时可以令 v 为 true，同时将 c 去掉。
 2. 只在子句 c 中出现 \bar{v} ，此时可以令 v 为 false，同时将 c 去掉。
 3. 只在子句 c 中出现 v 和 \bar{v} ，此时 v 可以为任意值，同时将 c 去掉。
 4. 在两个子句中分别出现 v 和 \bar{v} ，比如 $(v \wedge a \vee b) \wedge (\bar{v} \wedge c \vee d)$ ，此时可将 v 设定，并将这两个子句合并为 $(a \vee b \vee c \vee d)$ 。
 5. 在两个子句中出现只出现 v 和 \bar{v} ，即 $(v) \wedge (\bar{v})$ ，此时无解。

对每个变量都这样分情况处理，显然整个过程可在多项式时间内可以完成。

- b) 回顾 3SAT 到 INDEPENDENT SET 的归约过程，如果每个 literal 最多出现两次，那么在构造的图 G 中，任何顶点的度都不会超过 4。

Ex.8.7

按提示构造二分图 G 。因为每个子句（clause）恰好包含三个文字（literal），所以任意 n 个子句总共包含了 $3n$ 个文字（可能有重复）。又因每个变量最多对应三个文字，所以这 $3n$ 个文字至少对应了 n 个变量。于是在图 G 中，任意 n 个子句都与不少于 n 个变量相连，由 Hall 定理知图 G 存在完美匹配。找出一个完美匹配，假设子句 c 与变量 v 对配，若 c 中包含了 v 的肯定，则将 v 设为 true，若 c 包含了 v 的否定，则将 v 设为 false。若 c 同时包含了 v 的肯定及否定，那么 v 可以任意取值。分别为每个配对进行上述赋值过程，即可找到一组可满足解。

Ex.8.8

首先很显然，EXACT 4SAT 属于 NP。现在通过将 3SAT 归约到 EXACT 4SAT 来证明后者的 NP 完全性。对于任意一个 3SAT 实例，其中某个子句中包含了同一个文字多次，那么可以缩减为一次；如果同时包含了某个变量的肯定及否定，那么可以将这个变量去掉。然后，可以再在每个子句中可以添加一些哑变量（即没用的辅助变量），这样就可以将每个子句所包含的文字数目扩充到四个。至此，即已将该 3SAT 实例转化成了一个 EXACT 4SAT 问题。

Ex.8.9

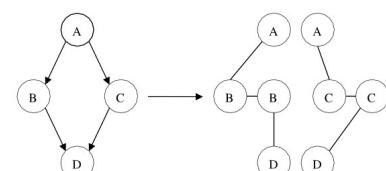
很容易将最小顶点覆盖归约到 HITTING SET。假设要求图 G 的最小顶点覆盖，可以建立一个 HITTING SET 实例，其中 S_1, S_2, \dots, S_n 即是图 G 的各条边，比如 $\{v_1, v_2\}, \{v_3, v_4\}, \dots$ ，通过二分式的询问（见 Ex.8.1），可以找到一个与 S_i 都相交的最小集合 H ，这正好就是图 G 的最小顶点覆盖。

Ex.8.10

- a) 令图 G 为一个环，环上的顶点数等于图 H 的顶点数。那么若 G 是 H 的同构子图，则说明 H 存在 Rudrata 回路。于是知 Rudrata 回路事实上是子图同构问题的一个特例。
 b) 如果令 $g = |V| - 1$ ，即得到一条 Rudrata 路径。
 c) 令 g 为子句的总数，即成 SAT。
 d) 令 $b = \frac{a(a-1)}{2}$ ，此时这 a 个顶点两两相连，于是即成最大团问题。
 e) 令 $b = 0$ ，即成最大独立集问题。
 f) 显然是最小顶点覆盖的一个推广。
 g) Hint 中所描述的特例即是一个 TSP。

Ex.8.11

a) 假设有向图 G ，对于 G 中的每个顶点 v ，将其分裂成 $i + o$ 新顶点，其中 i 是 v 的入度， o 是 v 的出度，然后将每个与 v 连结的有向边都转换成连向这 $i + o$ 个新顶点的无向边。为方便表示，在这些新顶点中，将与入边相连的顶点称为入顶点，与出边相连的顶点称为出顶点，这时还需要做的是，在每个入顶点与每个出顶点之间添加一条无向边，这样就将图 G 转化成了一个无向图 G' 。现在只需在这个无向图 G' 上寻找 Rudrata 路径即可。一个示例如下图：



- b) 在上一问的基础上，枚举所有顶点对 (s, t) 即可。

Ex.8.12

- a) 显然 k -SPANNING TREE 问题是可在多项式时间内验证的，因此是搜索问题。
 b) 若 $k = 2$ ，此时的 2-SPANNING TREE 实际上就是一条 Rudrata 路径。另外，这里好像有点问题，不知道我是否理解错了意思，因为当 $k \geq |V|$ 时，显然只要一次 DFS 就能找出解。

Ex.8.13

- a) 这个问题是属于 P 的，解法如下：选取任意一个 $V - L$ 中的顶点 s ，以 s 为源点进行 DFS。每当访问到 L 中的顶点时，停止向下扩展，这样 L 中的顶点就成了叶节点。若 DFS 完成后，所有顶点都被访问到，那么此时就找到了一棵满足要求的生成树。否则说明这样的生成树不存在。
 b) 是 NP 完全的，因为恰好有一个叶节点的生成树即是一条 Rudrata 路径。
 c) 是 NP 完全的，原因同 b)。
 d) 是 NP 完全的，原因同 b)。
 e) 是 NP 完全的，最多叶节点生成树的非叶子节点即构成最小连通支配集。最小支配集的 NP 完全性证明参考 Ex.8.20，而最小连通支配集的 NP 完全性证明与之类似，只要在构造图 G' 时同时再将所有顶点（辅助顶点除外）丙两相连即可。
 f) 是 NP 完全的，原因同 b)。

Ex.8.14

可以将最大团问题归约到此问题。假设要求任意图 $G(V, E)$ 中大小为 k 的团，可以在图 G 中添加 k 个相互独立的顶点，得到新图 G' ，这新加的 k 个顶点保证了图 G'

存在大小为 k 的独立集，同时又不影响到原图的团。

Ex.8.15

可以将最大独立集问题归约到此问题。比如若要求任意图 $G(V, E)$ 中大小为 d 的独立集，可以令 $G_1 = G(V, E)$ ，再令 $G_2(V, \emptyset)$ 的顶点集与 G 相同，但是边集为空，也即是各个顶点相互独立。于是 G_1 与 G_2 存在着大小为 d 的公共子图，当且仅当图 G 存在着大小为 d 的独立集。

Ex.8.16

考虑怎么把一个 3SAT 实例转化成一个 EXPERIMENTAL CUISINE 问题：对于任意一个 3SAT 问题，若其某一子句 c 为 $(U \vee V \vee W)$ ，我们可以相应地建立 7 种 ingredients，分别为 $\{\bar{U}\bar{V}\bar{W}, \bar{U}\bar{V}W, \bar{U}V\bar{W}, \bar{U}VW, \bar{U}V\bar{W}, \bar{U}VW, UV\bar{W}\}$ 。这 7 种 ingredients 代表了使得子句 c 成立的七种不同情况，比如 $UV\bar{W}$ 表示 U 和 V 同时为真，且 W 为假。它们之间当然完全不兼容的，因此将其中两两之间的 discord 值设为 1。现在再考虑子句与子句之间的情况，对于任意两个子句 i 和 j ，将这两个子句中相矛盾的成分之间的 discord 值设为 1，比如 UAB 与 $\bar{U}CD_j$ 等。设子句总数为 n ，再令 $p = 0$ 。若此时能选择的成份数为 n ，那么即说明此 3SAT 能满足。另外，判断一个 3SAT 是否能满足与找出这个 3SAT 的解实际上是等价的（可以参考 Ex.8.2），因此若 EXPERIMENTAL CUISINE 在多项式时间内可解的话，3SAT 亦然。

Ex.8.17

由 NP 问题的等价性可知，任何一个输入规模为 n 的 NP 问题，可以在多项式时间内转化为一个规模为 $p(n)$ 的 SAT 问题，而后者显然可以用穷举法在 $2^{p(n)}$ 时间内解决，于是得证。

Ex.8.18

虽然素因子分解属于 NP，若 P = NP 便意味着素因子分解可以在多项式时间内完成。在 RSA 加密算法的应用中，若某个用户的公钥为 (N, e) ，如果我们可以在多项式时

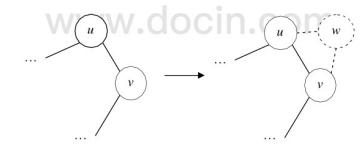
间内将 N 分解为 pq ，那么很容易就能获得其密钥 $d = e^{-1} \bmod (p-1)(q-1)$ 。

Ex.8.19

可以将团问题归约到 KITE 问题。若要求图 $G(V, E)$ 的最大团，类似于 Ex.8.14，可以在图 G 中添加 $|V|$ 个新顶点，并将每个新顶点都连向原图中不同的某个顶点，共形成了 $|V|$ 条新边，这样就得了一个新图 G' 。容易看出，在 G' 中存在大小为 $2g$ 的 kite 当且仅当 G 中存在大小为 g 的团。

Ex.8.20

可以将顶点覆盖问题归约到支配集问题。若要在图 $G(V, E)$ 中求得不大于 b 的一个顶点覆盖，可以先对图 G 做一个预处理：对每条边 $(u, v) \in E$ ，添加一个辅助顶点 w ，及两条边 (u, w) 和 (v, w) ，如下图所示：



对每条边都这样处理后得到一个新图 G' 。容易看出，若原图 G 中存在不大于 b 的顶点覆盖，这个顶点覆盖也是新图 G' 的一个支配集。反过来，若新图 G' 中存在一个不大于 b 的支配集，那么对这个支配集进行一些处理后也能得到一个图 G 的不大于 b 的顶点覆盖。处理过程如下：设该支配集为 D ，对于每条边 (u, v) 及相应的辅助顶点 w ，若 $w \notin D$ ，则不用做任何处理，若 $w \in D$ 且 $u, v \notin D$ ，那么可以在 D 中将 w 替换为 u 或 v ，若 $w \in D$ 同时 $u \in D \vee v \in D$ ，则直接将 w 从 D 中删掉即可。

Ex.8.21

- a) 提示已经足够明显了，这里只需要注意到有向图上的 Rudrata 路径问题可以归约到无向图上的 Rudrata 路径问题（见 Ex.8.11）。

- b) 对于每一个 k -string x ，把它表示成一条向边 (u, v) ，其中 $u = x[1:k-1]$ ，

$v = [2:k]$ ，对所有的 k -string 依次这样处理，并合并相同的顶点，得到一个有向图 G' 。此时，还原单串 x 即相当于找出图 G 的欧拉路径。关于有向图的欧拉路径问题可以参考 Ex.3.26。

Ex.8.22

- a) 显然 FAS 是可在多项式时间内验证的，因此属于 NP。
 b) 设 G 的一个大小为 b 的顶点覆盖为 C ，对于任意顶点 $v_i \in C$ ，设其在 G' 中相对应的顶点为 w_i 和 w'_i ，则将边 (w_i, w'_i) 加入到 E' 。对 G 中的每个顶点都这样处理后，所得的边集 E' 即是 G' 的一个大小为 b 的 feedback arc set。因为对于顶点 w_i 和 w'_i ，若将边 (w_i, w'_i) 后，所有与 w_i 相连的边都不可能位于任何一个环中，因为 w_i 不存在出边。同样，所有与 w'_i 相连的边也不可能位于任何一个环中，因为 w'_i 不存在入边。
 c) 对于 G 中的任意一条边 (v_i, v_j) ，设其在 G' 中相对应的顶点为 w_i, w_j, w_i', w_j' ，相对应的边为 $(w_i, w_j'), (w_i', w_j), (w_i', w_j')$ 。若 E' 是 G' 的一个大小为 b 的 feedback arc set，显然，在这四条边中至少有一条边 e 属于 E' ，否则就会形成环，而边 e 必然有个端点属于 $\{w_i, w_j\}$ 。若 w_i 是 e 的端点，则将 w_i 加入到 C ，否则将 w_j 加入到 C 。容易看出，在经过上述处理后， C 即是 G 的一个大小不超过 b 的顶点覆盖。

Ex.8.23

在此提示的基础上再做以下处理：对任意子句 $c = (l_1 \vee l_2 \vee l_3)$ ，分别将 s_i 与 l_1, l_2 与 l_3 联结起来形成三条路径，因为 l_1 为真使得 c 为真。另外，对于每个变量 v ，将 s_i 与所有 v 联结起来，再连向 t_i ，从而形成一条路径，再将 s_i 与所有 \bar{v} 联结起来，连向 t_i ，又形成一条路径。在这两条路径中，必然要选择其中一条，这保证了所有变量的一致性，即如果有任意子句选择了 v ，则其余子句就不能再选择 \bar{v} 。下面举一个简单的例子，假设要验证 CNF: $(x \vee y \vee z) \wedge (\bar{x} \vee y \vee \bar{z})$ 是否可以被满足，令 $u = (x \vee y \vee z)$, $v = (\bar{x} \vee y \vee \bar{z})$ ，可得 NODE-DISJOINT PATH 问题如下图：

