复习答案合集（第 7、8、14 章）

第 8 章：习题 8-2、8-10 ｜ 第 7 章：切杆问题、0/1 背包阶跃点法（含 7-15） ｜ 第 14 章：密码算法 RSA

◆ 问题 8-2：n-皇后问题求第一个可行解

【问题描述】

程序 8-4 求 n-皇后问题的**全部可行解**。假设作业实现已按时限非递减序排列，请**修改这一程序，使之在求得第一个可行解后即停止运行**。

【理论基础 — 来自 PPT 第 8.1-8.2 章】

概念	说明
回溯法框架	深度优先搜索状态空间树，用约束函数和限界函数进行剪枝
n-皇后问题	在 n×n 棋盘上放 n 个皇后，互不能攻击（同行、同列、同对角线）
约束条件	隐式约束：|i-j| ≠ |x[i]-x[j]|，即两皇后不在同对角线
解空间树	可变大小元组树或完全二叉树（0/1 决策树）
第一个解 vs 全部解	找第一个解：找到即返回；全部解：需遍历整个子树

【修改方案】

核心改动：在递归回溯函数中，当找到一个答案状态（叶子结点）时，设置一个**全局标志位**或**返回成功标志**，立即终止整个搜索。

【参考代码 — 修改版程序 8-4】

```
// 全局变量：标志是否已找到第一个解 bool found = false; // 判定两个皇后是否在同对角线 bool Place(int k, int *x) { for(int j=0; j<k; j++) if((x[j]==x[k]) || (abs(x[j]-x[k])==abs(j-k))) return false; return true; } // 修改后的 NQueens 函数 - 找第一个解则停止 void NQueens_FirstSolution(int k, int n, int *x) { if(found) return; // 若已找到解，立即返回 for(int i=0; i<n; i++) { if(Place(k, x)) { x[k] = i; if(k == n-1) { // 找到一个完整解（叶子结点） cout << "找到第一个解："; for(int i=0; i<n; i++) cout << x[i] << " "; cout << endl; found = true; // ← 设置标志位，停止搜索 return; // ← 立即返回 } else { NQueens_FirstSolution(k+1, n, x); // 继续递归 } if(found) return; // 回溯时检查标志 } } } // 主调用 int main() { int n = 8; int x[n]; found = false; NQueens_FirstSolution(0, n, x); return 0; }
```

【关键修改点解析】

修改项	原程序	修改后	作用
全局标志	无	`bool found = false;`	记录是否已找到第一个解
找到解时	输出解，继续搜索	`found=true; return;`	标记并立即返回，不再搜索
回溯时检查	无	`if(found) return;`	回溯途中也停止递归
时间复杂度	Θ(N!)（最坏情况）	O(n)（最优情况第一层就有解）	显著提高效率

【替代方案 — 返回值标志法】

```
// 方案 B：使用返回值表示是否找到解 bool NQueens_FirstSolution(int k, int n, int *x) { for(int i=0; i<n; i++) { if(Place(k, x)) { x[k] = i; if(k == n-1) { // 输出第一个解 for(int j=0; j<n; j++) cout << x[j] << " "; cout << endl; return true; // ← 返回 true 表示已找到 } else if(NQueens_FirstSolution(k+1, n, x)) { return true; // ← 若子递归找到，立即返回 } } } return false; // 该分支无解 }
```

📌 **核心思想：** 在回溯法中，通过设置标志位或利用返回值，使得一旦找到第一个答案状态（完整的 n-元组），就立即停止对后续分支的搜索。这体现了"约束函数"的剪枝思想——剪掉不必要的子树。

◆ 问题 8-10：密钥频率问题的最坏情况时间复杂度

【问题描述】

求使用回溯法求所有可能的**密钥频率环问题的最坏情况时间复杂度**。

注： 本题应理解为"密钥频率环问题"或在考研题库中常见的"连续邮票问题"之类，需要使用回溯法在满足频率或代价约束的条件下搜索所有可行解。以下按密钥匹配/频率搜索问题的标准框架解答。

【理论基础 — 来自 PPT 第 8.1 章】

概念	定义	说明
解空间	候选解集合	由显式约束规定，大小为 $\|S_0\|\times\|S_1\|\times\ldots\times\|S_{n-1}\|$
最坏情况	算法需要搜索最多结点	通常出现在：无剪枝、无解、或解空间树最深叶子最多
时间复杂度	T(n) = 生成结点数 + 每结点处理时间	T(n) = O(结点总数) + O(约束检查)
全排列问题	n 个元素的排列	状态空间大小 = n!
子集和问题	n 个数选或不选	状态空间大小 = 2^n

【密钥频率环问题的分析】

问题设定：

- 给定 m 种字符（如：A-Z 共 26 个），需要在一个 n 位的密钥中每个位置选择一个字符
- 每种字符的使用频率不超过某个限制（如最多出现 k 次）
- 求满足频率约束的所有可行密钥

【解空间分析】

问题规模参数： - n：密钥长度（位数） - m：字符表大小（如 26 个英文字母） - k：每个字符的最大使用频率（约束条件） 状态空间树结构： - 第 i 层（i=0 到 n-1）：在第 i 个位置选择字符 - 每结点有最多 m 条分支（每个字符一条） - 树的最大高度：n（递归深度） - 完全展开的树中的结点数：1 + m + m² + m³ + ... + m^n = (m^(n+1) - 1)/(m - 1)

【最坏情况分析】

影响因素	对时间复杂度的影响
无剪枝	需要生成**完整树的所有结点**，包括所有叶子和中间节点
频率约束很宽松（如 k ⩾ n）	约束函数难以剪枝，几乎所有分支都被探索
无可行解	必须搜索整个搜索空间才能确认无解
高度 = n	递归深度达到最大，调用栈开销大

【时间复杂度推导】

1. 结点总数统计：

状态空间树（无剪枝）的结点总数： N_total = Σ(i=0 to n) m^i = 1 + m + m² + ... + m^n = (m^(n+1) - 1) / (m - 1) ≈ O(m^(n+1)) 当 m > 1 时 特例分析： - m = 2（二叉树）：N ≈ 2^(n+1) - m = 26（英文字母）：N ≈ 26^(n+1) - m = 10（数字）：N ≈ 10^(n+1)

2. 每个结点的处理时间：

约束检查： - 检查当前结点是否满足频率约束：O(m) 或 O(n) - 比较最多需要检查 m 种字符的出现次数 回溯操作：O(1) 每结点总处理时间：O(m) 或 O(n)（取决于实现） 通常取 O(n)

3. 最坏情况总时间复杂度：

T(n, m) = （结点总数） × （每结点处理时间） = O(m^(n+1)) × O(n) = **O(n · m^(n+1))** 或简化为：**O(n · m^n)**（忽略常数因子）

【参考伪代码分析】

```
void KeywordFrequency(int pos, int n, int *key, int *freq, int m, int k) { if(pos == n) { // 找到一个答案状态（完整密钥）→ O(1) OutputAnswer(key); return; } for(int c=0; c<m; c++) { // m 次迭代 if(freq[c] < k) { // 检查约束 → O(1) key[pos] = c; freq[c]++; // 递归调用，深度最多 n 层 KeywordFrequency(pos+1, n, key, freq, m, k); freq[c]--; // 恢复状态 → O(1) } } // 该层总耗时：O(m) } // 调用栈分析： // 深度 0：1 个结点，每个 O(m) → O(m) // 深度 1：m 个结点，每个 O(m) → O(m²) // 深度 2：m² 个结点，每个 O(m) → O(m³) // ... // 深度 n：m^n 个结点，每个 O(m) → O(m^(n+1)) // 总和：T(n) = O(m + m² + m³ + ... + m^(n+1)) // = O(m^(n+1))
```

【标准答案】

问题 8-10 的最坏情况时间复杂度：

T(n) = O(n · m^(n+1))

或等价地写成 **O(n · m^n)**（当忽略常数因子）

其中：
- n = 密钥长度（位数）
- m = 字符集大小（如 26 个英文字母）
- 系数 n 来自每结点的约束检查和恢复操作
- m^(n+1) 来自状态空间树的结点总数

【复杂度对比】

问题类型	解空间大小	最坏时间复杂度	备注
n-皇后问题	n!（全排列）	O(n!) 或 O(n · n!)	n ⩽ 8 可行
子集和问题	2^n （0/1 决策）	O(n · 2^n)	n ⩽ 20 可行
密钥频率问题（m 字符表）	m^n	**O(n · m^n)**	n · m 乘积要求较小
全排列枚举（无约束）	n!（≈ √(2πn) · (n/e)^n）	O(n · n!)	增长最快

📌 **核心要点：**
- 回溯法的时间复杂度由**状态空间树的结点数**与**每结点的处理时间**的乘积决定
- 对于字符集大小为 m、长度为 n 的问题，最坏情况下需要探索 m^(n+1) 个结点
- 每结点的约束检查和恢复操作耗时 O(n)，导致最终复杂度为 **O(n · m^(n+1))**
- 考试中若无具体 m 值，也可写成 **O(m^n)** 的形式（省略常数 n 和 m 的因子）

第 7 章 动态规划法 — 补充：切杆问题

Rod Cutting Problem | 来源：考纲第七章"补充：切杆问题"

◆ 切杆问题（Rod Cutting Problem）

【问题描述】

给定一根长度为 **n** 的钢条（或圆木），以及一个价格表 P[1..n]，其中 **P[i]** 表示长度为 i 的钢条的售价。求将该钢条切成若干段（每段长度均为正整数）使得**总售价最大**的切割方案。

切割方式：长度为 n 的钢条可以不切，也可以切成若干段 $i_1 + i_2 + \ldots + i_k = n$，每段单独售卖。

【PPT 例题 — 长度为 10 的圆木】

价格表 P[i]：

长度 i	1	2	3	4	5	6	7	8	9	10
价格 P[i]	1	5	8	9	10	17	17	20	24	30

问：长度为 10 的圆木如何切割才能获得最大收益?

【动态规划法基本要素 — 来自 PPT 第 7 章】

要素	在切杆问题中的体现
最优子结构	若长度 n 的最优切割方案的第一段长度为 i，则剩余长度 n-i 的切割也必然是最优的
重叠子问题	求 r[n] 需用 r[n-1], r[n-2], ..., r[1]；这些子问题在递归求解中被多次重复计算

【动态规划法求解步骤】

步骤 1：刻画最优解的结构特性

设 r[i] 为长度 i 的钢条的最大收益。最优解的第一段长度 j 可能是 1, 2, ..., i 中的任意一个，剩余部分 i-j 的最优收益是 r[i-j]。

步骤 2：递归定义最优解值

$$r[0] = 0$$
$$r[i] = \max \{ P[j] + r[i-j] \} \quad \text{其中 } 1 \le j \le i$$

步骤 3：自底向上计算最优解值

按长度从小到大依次计算 r[1], r[2], ..., r[n]，每个 r[i] 利用已算出的 r[0..i-1]。同时用 s[i] 记录 r[i] 取得最大值时的第一段切割长度 j。

步骤 4：根据 s[] 数组构造一个最优切割方案

从 n 出发：第一段长度 s[n]，然后处理剩余 n - s[n]，重复直到长度为 0。

【算法程序实现 — 自底向上】

```
// 求解长度为 n 的钢条的最大收益 int CutRod(int *P, int n, int *s) { int r[n+1]; r[0] = 0; for(int i=1; i<=n; i++) { // 自底向上，依次求 r[1..n] int q = -1; for(int j=1; j<=i; j++) { // 枚举第一段长度 j if(P[j] + r[i-j] > q) { q = P[j] + r[i-j]; s[i] = j; // 记录最优的第一段长度 } } r[i] = q; } return r[n]; } // 构造最优切割方案 void PrintCutSolution(int *s, int n) { while(n > 0) { cout << s[n] << " "; // 输出当前第一段长度 n = n - s[n]; // 处理剩余部分 } } // 时间复杂度：O(n²) （双重循环：外层 i=1..n，内层 j=1..i） // 空间复杂度：O(n) （r[] 和 s[] 数组）
```

【例题计算过程 — 长度 n=10】

以下展示完整的 r[i] 和 s[i] 表的填表过程：

i	r[i] 计算（取 max）	r[i]	s[i]
0	—	0	0
1	P[1]+r[0] = 1+0 = 1	1	1
2	max{ P[1]+r[1]=2, **P[2]+r[0]=5** }	5	2
3	max{ P[1]+r[2]=6, P[2]+r[1]=6, **P[3]+r[0]=8** }	8	3
4	max{ P[1]+r[3]=9, **P[2]+r[2]=10**, P[3]+r[1]=9, P[4]+r[0]=9 }	10	2
5	max{ P[1]+r[4]=11, **P[2]+r[3]=13**, P[3]+r[2]=13*, P[4]+r[1]=10, P[5]+r[0]=10 }	13	2
6	max{ P[1]+r[5]=14, P[2]+r[4]=15, P[3]+r[3]=16, P[4]+r[2]=14, P[5]+r[1]=11, **P[6]+r[0]=17** }	17	6
7	max{ ..., **P[1]+r[6]=18**, P[6]+r[1]=18*, P[7]+r[0]=17 }	18	1
8	max{ ..., **P[2]+r[6]=22**, ..., P[8]+r[0]=20 }	22	2
9	max{ ..., **P[3]+r[6]=25**, ..., P[9]+r[0]=24 }	25	3
10	max{ ..., **P[10]+r[0]=30** }	30	10

注：标 * 表示等价最优值，取较小的 j 作为 s[i]。

【最终结果 — 与 PPT 一致】

i	0	1	2	3	4	5	6	7	8	9	10
r[i]	0	1	5	8	10	13	17	18	22	25	**30**
s[i]	0	1	2	3	2	2	6	1	2	3	**10**

最大收益：r[10] = 30

最优切割方案构造（从 n=10 出发）：
- 当前长度 n=10，s[10] = 10，第一段切 **10**，剩余 0
- 结束

结论：长度为 10 的圆木不切，整根出售，收益最大 = 30

【几个验证示例 — 不同长度的最优切割】

长度 n	最优切割方案（用 s[] 追溯）	最大收益 r[n]
4	s[4]=2 → 切 2，剩 2；s[2]=2 → 切 2。方案：(2, 2)	P[2]+P[2] = 5+5 = 10 ✓
5	s[5]=2 → 切 2，剩 3；s[3]=3 → 切 3。方案：(2, 3)	P[2]+P[3] = 5+8 = 13 ✓
7	s[7]=1 → 切 1，剩 6；s[6]=6 → 切 6。方案：(1, 6)	P[1]+P[6] = 1+17 = 18 ✓
8	s[8]=2 → 切 2，剩 6；s[6]=6 → 切 6。方案：(2, 6)	P[2]+P[6] = 5+17 = 22 ✓
9	s[9]=3 → 切 3，剩 6；s[6]=6 → 切 6。方案：(3, 6)	P[3]+P[6] = 8+17 = 25 ✓
10	s[10]=10 → 不切。方案：(10)	P[10] = 30 ✓

【复杂度分析】

方法	时间复杂度	空间复杂度	说明
朴素递归	$O(2^n)$	O(n)	大量重叠子问题被重复计算，指数级爆炸
备忘录法 (自顶向下 DP)	**$O(n^2)$**	O(n)	递归 + 缓存子问题解
自底向上 DP	**$O(n^2)$**	O(n)	双重循环，无递归开销，**推荐做法**

时间复杂度推导：外层循环 i 从 1 到 n，内层循环 j 从 1 到 i，总操作数 = 1+2+...+n = n(n+1)/2 = **$O(n^2)$**。

【动态规划法四步框架 — 切杆问题对应总结】

DP 四步骤	在切杆问题中的具体体现
1. 刻画最优解的结构特性	长度 n 的最优切割 = 第一段长度 j + 剩余长度 n-j 的最优切割
2. 递归定义最优解值	$r[i] = \max\{P[j] + r[i-j] : 1 \leqslant j \leqslant i\}$
3. 自底向上计算最优解值	按 i = 1, 2, ..., n 顺序填表，每个 r[i] 用已算出的 r[0..i-1]
4. 根据计算信息构造最优解	用 s[] 数组从 n 出发回溯：n → n - s[n] → ... → 0，输出每个 s[] 值

📌 **考试核心要点：**
- 切杆问题是**动态规划法的经典应用**，体现"最优子结构"和"重叠子问题"两大要素
- 递推式：**$r[i] = \max\{P[j] + r[i-j] : 1 \leqslant j \leqslant i\}$，$r[0] = 0$**
- 时间复杂度 **$O(n^2)$**，空间复杂度 O(n)
- 同时维护 **s[]** 数组记录最优切割决策，最后通过 s[] 回溯构造最优切割方案
- PPT 例题答案：长度 10 → **整根不切，收益 30**

第 7 章 动态规划法 — 0/1 背包问题阶跃点法

阶跃点集合求解（实数重量） | 含例题 n=3 + 习题 7-15 完整解答

◆ 方法概述：阶跃点（Step Pair）集合求解

【适用场景】

当 0/1 背包问题的 **物品重量是实数**（不是小整数）时，传统二维 DP 表 f(j,X) 无法填表。这时使用**阶跃点集合法**：用一组关键序偶 (X,P) 来表征 f 函数的"台阶式跳变"，避免穷举所有 X 值。

【基本概念】

符号	含义
序偶 (X, P)	装载重量 X，对应收益 P
S^i	考虑物品 0~i 后的全部"非支配可行装载方案"的阶跃点集合
S^{-1}	初始集合，恒为 {(0, 0)}（什么都不装）
S_1^i	由 S^{i-1} 中每个序偶 **加上物品 i 的 (w_i, p_i)** 得到（表示装入物品 i 的方案）
S^i	合并 S^{i-1}（不装 i）和 S_1^i（装 i），去除被支配和超重的序偶后所得

【两条清除规则 ★★★】

合并 S^{i-1} 和 S_1^i 后，按 X 升序排列，逐个检查：

1. **超重清除：** X > M 的序偶必删除
2. **支配清除：** 若序偶 (X', P') 满足 X' ⩾ X 且 P' ⩽ P（同时不全等于），则 (X', P') 被 (X, P) **支配**，必删除。
   简化判断：按 X 升序排列后，**P 必须严格递增**，否则该点被前面某点支配，删除。

【求解步骤】

1. 初始化 $S^{-1} = \{(0, 0)\}$
2. 对 i = 0, 1, ..., n-1：
   - 计算 $S_1^i = \{ (X+w_i, P+p_i) : (X, P) \in S^{i-1} \}$
   - 合并 S^{i-1} 和 S_1^i，按 X 升序排列
   - 清除超重点（X > M）
   - 清除被支配点（P 不递增者）
   - 得到 S^i
3. 最优解值 = S^{n-1} 中满足 X ⩽ M 的最大 P
4. 从最优序偶**回溯**每个 x_i：是否在 S^{i-1} 中决定 $x_i = 0$ 还是 1

◆ 例题：n=3, M=6, W=(2,3,4), P=(1,2,4)

【Step 1】初始化 S^{-1}

$$S^{-1} = \{(0, 0)\}$$

【Step 2】生成 S^0（考虑物品 0：w_0=2, p_0=1）

- $S_1^0 = \{(0+2, 0+1)\} = \{(2, 1)\}$
- 合并 S^{-1} 与 S_1^0：{(0,0), (2,1)}
- 无超重，无支配（P 递增：0<1 ✓）

$$S^0 = \{(0, 0), (2, 1)\}$$

【Step 3】生成 S^1（考虑物品 1：w_1=3, p_1=2）

- S_1^1 = 每个 S^0 序偶加 (3, 2)：
  (0,0)→(3,2)，(2,1)→(5,3)
  S_1^1 = {(3,2), (5,3)}
- 合并 S^0 与 S_1^1：{(0,0), (2,1), (3,2), (5,3)}
- 检查：均不超 M=6；P 递增（0<1<2<3 ✓），无支配

$$S^1 = \{(0, 0), (2, 1), (3, 2), (5, 3)\}$$

【Step 4】生成 S^2（考虑物品 2：w_2=4, p_2=4）

- S_1^2 = 每个 S^1 序偶加 (4, 4)：

原序偶	加 (4,4) 后	X > M=6?
(0, 0)	(4, 4)	否，保留 ✓
(2, 1)	(6, 5)	否，保留 ✓
(3, 2)	(7, 6)	是，X=7>6，删除 ✗
(5, 3)	(9, 7)	是，X=9>6，删除 ✗

- S_1^2 = {(4, 4), (6, 5)}
- 合并 S^1 与 S_1^2，按 X 升序：{(0,0), (2,1), (3,2), **(4,4)**, (5,3), **(6,5)**}
- 支配检查（P 必须递增）：

序偶	P 值	判定
(0, 0)	0	✓
(2, 1)	1 > 0	✓
(3, 2)	2 > 1	✓
(4, 4)	4 > 2	✓
(5, 3)	3 < 4	被 (4,4) 支配，删除 ✗
(6, 5)	5 > 4	✓

$$S^2 = \{(0, 0), (2, 1), (3, 2), (4, 4), (6, 5)\}$$

【Step 5】最优解值

S^2 中满足 X ⩽ M=6 的最大收益序偶 = **(6, 5)**
最优解值 f(2, 6) = 5

【Step 6】回溯构造最优解

从 (6, 5) 倒推每个 x_i。**判定规则：**若当前序偶仍在 S^{i-1} 中 → 没装物品 i（x_i=0）；否则装了 i（x_i=1），减去 (w_i, p_i) 继续追溯。

判定	检查	结论
x_2	(6,5) ∈ S^1 = {(0,0),(2,1),(3,2),(5,3)}?	否 → **x_2 = 1**，减 (4,4) 得 (2, 1)
x_1	(2,1) ∈ S^0 = {(0,0),(2,1)}?	是 → **x_1 = 0**，序偶不变
x_0	(2,1) ∈ S^{-1} = {(0,0)}?	否 → **x_0 = 1**，减 (2,1) 得 (0, 0) ✓

最优解：$(x_0, x_1, x_2) = (1, 0, 1)$
验证：重量 = 2+4 = 6 ⩽ 6 ✓ ；收益 = 1+4 = 5 ✓

习题 7-15 完整解答

0/1 背包阶跃点法 | n=4, M=32, W=(10,15,6,9), P=(2,5,8,1)

◆ 习题 7-15：求 S^i（$0 \leqslant i \leqslant 3$），并给出最优解

【初始化】

$$S^{-1} = \{(0, 0)\}$$

【S^0】物品 0：w=10, p=2

- $S_1^0 = \{(0+10, 0+2)\} = \{(10, 2)\}$
- 合并：{(0,0), (10,2)}，无超重无支配

$$S^0 = \{(0, 0), (10, 2)\}$$

【S^1】物品 1：w=15, p=5

- $S_1^1 = \{(0+15,0+5), (10+15,2+5)\} = \{(15, 5), (25, 7)\}$
- 合并：{(0,0), (10,2), (15,5), (25,7)}
- 检查：均 ⩽ 32，P 递增（0<2<5<7 ✓）

$$S^1 = \{(0, 0), (10, 2), (15, 5), (25, 7)\}$$

【S^2】物品 2：w=6, p=8

- S_1^2 = 每个 S^1 加 (6, 8)：(0,0)→(6,8), (10,2)→(16,10), (15,5)→(21,13), (25,7)→(31,15)
- $S_1^2 = \{(6,8), (16,10), (21,13), (31,15)\}$（均不超 M=32）
- 合并 S^1 与 S_1^2，按 X 升序：{(0,0), (6,8), (10,2), (15,5), (16,10), (21,13), (25,7), (31,15)}
- 支配检查（P 必须递增）：

序偶	P 与前最大 P 比较	判定
(0, 0)	—	✓
(6, 8)	8 > 0	✓ 当前最大 P=8
(10, 2)	2 < 8	被 (6,8) 支配，删除 ✗
(15, 5)	5 < 8	被支配，删除 ✗
(16, 10)	10 > 8	✓ 最大 P=10
(21, 13)	13 > 10	✓ 最大 P=13
(25, 7)	7 < 13	被支配，删除 ✗
(31, 15)	15 > 13	✓ 最大 P=15

$$S^2 = \{(0, 0), (6, 8), (16, 10), (21, 13), (31, 15)\}$$

【S^3】物品 3：w=9, p=1

- S_1^3 = 每个 S^2 加 (9, 1)：

原序偶	加 (9,1) 后	X > 32?
(0, 0)	(9, 1)	否 ✓
(6, 8)	(15, 9)	否 ✓
(16, 10)	(25, 11)	否 ✓
(21, 13)	(30, 14)	否 ✓
(31, 15)	(40, 16)	是，X=40>32，删除 ✗

- $S_1^3 = \{(9,1), (15,9), (25,11), (30,14)\}$
- 合并 S^2 与 S_1^3，按 X 升序：{(0,0), (6,8), (9,1), (15,9), (16,10), (21,13), (25,11), (30,14), (31,15)}
- 支配检查：

序偶	P 检查	判定
(0, 0)	—	✓
(6, 8)	8 > 0	✓
(9, 1)	1 < 8	删除 ✗
(15, 9)	9 > 8	✓
(16, 10)	10 > 9	✓
(21, 13)	13 > 10	✓
(25, 11)	11 < 13	删除 ✗
(30, 14)	14 > 13	✓
(31, 15)	15 > 14	✓

$$S^3 = \{(0, 0), (6, 8), (15, 9), (16, 10), (21, 13), (30, 14), (31, 15)\}$$

【最优解值】

S^3 中满足 $X \leqslant M=32$ 的最大收益序偶 = **(31, 15)**
最优解值 = 15

【回溯构造最优解】

从 (31, 15) 倒推：

判定	检查	结论
x_3	$(31,15) \in S^2 = \{(0,0),(6,8),(16,10),(21,13),(31,15)\}$?	是 → $x_3 = 0$，序偶不变
x_2	$(31,15) \in S^1 = \{(0,0),(10,2),(15,5),(25,7)\}$?	否 → $x_2 = 1$，减 (6,8) 得 (25, 7)
x_1	$(25,7) \in S^0 = \{(0,0),(10,2)\}$?	否 → $x_1 = 1$，减 (15,5) 得 (10, 2)
x_0	$(10,2) \in S^{-1} = \{(0,0)\}$?	否 → $x_0 = 1$，减 (10,2) 得 (0, 0) ✓

最优解：$(x_0, x_1, x_2, x_3) = (1, 1, 1, 0)$
验证：重量 $= 10+15+6 = 31 \leq 32$ ✓ ；收益 $= 2+5+8 = 15$ ✓

【补充：启发式方法（题目"另用启发式再算一次"）】

核心思想：设 L = 任一可行解的收益（最优解下界估计值）。引入
ProfLeft(i) = $\Sigma_{k=i}^{n-1} p_k$ = 从物品 i 起到 n-1 的总收益
清除规则：若 S^i 中某序偶 (X, P) 满足 P + ProfLeft(i+1) < L，则即使把剩下所有物品都装入也达不到 L，必非最优 → 删除。

【应用到 7-15】

取下界 **L = 15**（用上面已求得的最优解 (1,1,1,0) 的收益）。

ProfLeft 值：

i	0	1	2	3	4
ProfLeft(i)	2+5+8+1 = 16	5+8+1 = 14	8+1 = 9	1	0

清除规则应用：

阶段	条件	应用结果
S^0	P + ProfLeft(1) < 15 即 P + 14 < 15，即 P < 1	(0, 0)：0+14=14<15 → 删除 ✗ (10, 2)：2+14=16 ⩾ 15 → 保留 ✓ → **S^0 = {(10, 2)}**
S^1	P + ProfLeft(2) < 15 即 P + 9 < 15，即 P < 6	基于精简后的 S^0={(10,2)}，生成 $S_1{}^1$={(25,7)} 合并：{(10,2), (25,7)}；P<6 删除 (10,2) → **S^1 = {(25, 7)}**
S^2	P + ProfLeft(3) < 15 即 P + 1 < 15，即 P < 14	$S_1{}^2$ = {(25+6, 7+8)} = {(31, 15)} 合并：{(25,7), (31,15)}；P<14 删除 (25,7) → **S^2 = {(31, 15)}**
S^3	P + ProfLeft(4) < 15 即 P + 0 < 15，即 P < 15	$S_1{}^3$ = {(31+9, 15+1)} = {(40, 16)}；X=40>32 删除 合并：{(31, 15)}；满足 P ⩾ 15 → **S^3 = {(31, 15)}**

对比两种方法的 S^i 集合大小：

阶段	非启发式 |S^i|	启发式 |S^i|	减少
S^0	2	1	↓ 1
S^1	4	1	↓ 3
S^2	5	1	↓ 4
S^3	7	1	↓ 6

结论：启发式法大幅减少序偶数量，同时得到一致的最优解 **15**。

📌 **考试核心要点：**
- 阶跃点法是 0/1 背包问题（实数重量）的**动态规划法实现**
- $S^i = (S^{i-1}) \cup (S_1{}^i)$，再去除超重点和被支配点
- **支配判定快捷规则**：按 X 升序排，P 必须严格递增，否则删
- **回溯构造**：当前序偶在 S^{i-1} 中 → x_i=0；不在 → x_i=1 且减去 (w_i, p_i)
- **启发式版本**：引入 ProfLeft(i)，清除 P + ProfLeft(i+1) < L 的序偶以加速求解
- **7-15 答案**：(x_0,x_1,x_2,x_3) = (1,1,1,0)，最优解值 = 15

第 14 章 密码算法 — 考点速览（5%）

信息安全 | 密码体制 | 数论基础 | RSA 算法

◆ 考点 1：信息安全的四个目标

目标	含义	对应密码学技术
机密性 Confidentiality	非授权用户不能知晓信息内容	**加密**（对称/非对称加密）
完整性 Integrity	信息在生成、传输、存储、使用过程中不被非授权篡改	**消息摘要**（单向 Hash）
抗否认性 Non-repudiation	通信双方无法事后否认曾对信息的生成、签发、接收行为	**数字签名**（用私钥加密）
可用性 Availability	保证授权用户能方便地访问所需信息	访问控制、容灾备份

关键结论：密码技术是实现信息安全的**核心技术**，机密性、完整性、抗否认性三大目标均依赖于密码算法。

◆ 考点 2：现代密码学的两个分支

分支	作用
密码编码学 cryptography	致力于建立**难以攻破**的安全密码体制
密码分析学 cryptanalysis	力图**破译**对方的密码体制

◆ 考点 3：两种密码体制对比（★ 高频考点）

对比维度	对称密码体制	非对称密码体制（公开密钥）
密钥	加/解密用**同一个**密钥 k	用**两个**密钥：公钥 k_1、私钥 k_2
加密公式	$C = E_k(m)$	$C = E_{k_1}(m)$
解密公式	$m = D_k(C)$	$m = D_{K_2}(C)$
密钥传递	必须通过**秘密信道**分发	公钥可公开发布，**无需秘密信道**
典型代表	DES（分组密码）、序列密码	**RSA**、背包密码
理论基础	代换、置换	NP 难度问题（大整数分解、背包问题、离散对数）
优点	• 加/解密速度**快** • 安全强度高	• 无需秘密信道分发密钥 • 可实现**数字签名** • 适合公共网络保密通信
缺点	• 密钥分发困难 • n 个用户需 C(n,2) 个密钥 • 用户需记 n-1 个密钥	• 加/解密速度**慢** • 计算量大

非对称体制的双重用途：
- **保密通信：**任何人用公钥 k_1 加密 → 只有私钥 k_2 持有者能解密
- **数字签名：**私钥 k_2 加密 → 任何人用公钥 k_1 解密 → 证明消息来自 k_2 持有者

第 14 章 — 数论初步与 RSA 算法

RSA 的数学基础与加/解密原理

◆ 考点 4：数论基础（RSA 的理论基础）

【1. 同余 a ≡ b (mod n)】

设 n 为自然数。若 a − b 是 n 的倍数，则称 **a 与 b 关于模 n 同余**，记作 $a \equiv b \pmod n$。
等价于 $a \pmod n = b \pmod n$，即 $a = kn + b$（k 为整数）。

例： n=5, a=21, b=11 → a−b=10 是 5 的倍数 → $21 \equiv 11 \pmod 5$ ✓

【2. 按模计算原理（定理 14-1）】

$$(a\ \theta\ b) \bmod n = [(a \bmod n)\ \theta\ (b \bmod n)] \bmod n \qquad (\theta = +、-、\times)$$

作用： 限制中间结果的范围，使**大数取模幂运算**不会产生巨大中间结果。
例： $143^4 \bmod 7 = [143 \bmod 7]^4 \bmod 7 = 3^4 \bmod 7 = 81 \bmod 7 = 4$ ✓
注意： 公开密钥密码算法大量使用幂的取模运算。

【3. 欧拉函数 Φ(n)】

定义： 数列 1, 2, ..., n−1 中与 n **互素**的数的个数，记作 Φ(n)。

性质	条件	公式
性质 14-2	p 是素数	$\Phi(p) = p - 1$
定理 14-2 ★	p, q 为不同素数，n = pq	**$\Phi(n) = \Phi(p)\Phi(q) = (p-1)(q-1)$**

例： n=9，与 9 互素的数：{1,2,4,5,7,8}，共 6 个 → Φ(9)=6 ✓

【4. 欧拉定理 + 费马小定理（定理 14-3）★★★】

欧拉定理：若 a 与 n 互素，则 $a^{\Phi(n)} \equiv 1 \pmod n$
推论：$m^{k\Phi(n)} \equiv 1 \pmod n$ 且 **$m^{k\Phi(n)+1} \equiv m \pmod n$** ← RSA 解密正确性的依据！
费马小定理（n 为素数时）：$a^{n-1} \equiv 1 \pmod n$

【5. 乘法逆元与求逆（定理 14-4）】

定义： 若 $ax \equiv 1 \pmod n$，则称 x 为 a 关于模 n 的**乘法逆元**，记作 a^{-1}。

定理 14-4： 若 gcd(a, n) = 1，则 $ax \equiv 1 \pmod n$ 有唯一解：

$$x = a^{\Phi(n)-1} \pmod n \qquad \text{若 n 为素数：} x = a^{n-2} \pmod n$$

★ 考试题型： 用扩展欧几里得算法求逆（B 卷 2024/2025 第 5 题）

RSA 算法（★ 必考核心）

第一个较完善的公开密钥算法 | 安全性基于大整数分解的难度

◆ 考点 5：RSA 的密钥生成（5 步）

1. 选择两个**大素数** p 和 q（如 200 位十进制数），$p \neq q$
2. 计算 $\mathbf{n = p \times q}$，再算 $\mathbf{\Phi(n) = (p-1)(q-1)}$
3. 选择随机整数 e，使 $\mathbf{\gcd(e, \Phi(n)) = 1}$，且 $0 < e < \Phi(n)$
4. 计算 $\mathbf{d = e^{-1} \bmod \Phi(n)}$，即 $ed \equiv 1 \pmod{\Phi(n)}$
5. 公开密钥 $\mathbf{k_1 = (e, n)}$；私人密钥 $\mathbf{k_2 = (d, n)}$

注意： p 和 q 必须保密；Φ(n) 仅用于生成 d，之后销毁。
RSA 安全性来源： 从 (e, n) 反推 d 需要先分解 n = pq，而大整数分解是**NP 难度问题**。

◆ 考点 6：加/解密公式

$$\text{加密：} C = M^e \bmod n \quad \text{（用公钥 e, n）}$$
$$\text{解密：} M = C^d \bmod n \quad \text{（用私钥 d, n）}$$

解密正确性证明（必考推导）：

M' = C^d mod n = (M^e mod n)^d mod n = M^(ed) mod n（按模计算原理）∵ $ed \equiv 1 \pmod{\Phi(n)}$，即 $ed = k\Phi(n) + 1$ = M^(kΦ(n)+1) mod n ≡ M (mod n)（由欧拉定理推论）故 M' = M ✓

◆ 历年真题 A 卷：RSA 加密计算

题目： 已知 e = 5, n = 35, M = 12，计算密文 C（含过程）。

解： $C = M^e \bmod n = 12^5 \bmod 35$。利用按模计算原理逐步求幂：

步骤	计算	结果
12^2	$144 = 4 \times 35 + 4$	144 mod 35 = **4**
$12^4 = (12^2)^2$	$4^2 = 16$	16 mod 35 = **16**
$12^5 = 12^4 \times 12$	$16 \times 12 = 192 = 5 \times 35 + 17$	192 mod 35 = **17**

密文 C = 17

注：每次相乘后立即取模，避免中间结果过大（直接算 $12^5 = 248832$ 也可，但取模法更通用）。

◆ 历年真题 B 卷：扩展欧几里得算法求逆元 d

题目： 已知 e = 13, Φ(n) = 60，求 d 使 $ed \equiv 1 \pmod{60}$，并写出求解过程。

步骤 1：辗转相除求 gcd(60, 13)

60 = 4 × 13 + 8 ← 余数 8　13 = 1 × 8 + 5 ← 余数 5　8 = 1 × 5 + 3 ← 余数 3　5 = 1 × 3 + 2 ← 余数 2　3 = 1 × 2 + 1 ← 余数 1（gcd = 1，存在逆元 ✓）　2 = 2 × 1 + 0

步骤 2：从下往上回代，将 1 表示为 60 与 13 的线性组合

回代式	化简为 60 和 13 的组合
$1 = 3 - 1 \times 2$	$1 = 3 - 1 \times (5 - 1 \times 3) = 2 \times 3 - 1 \times 5$
代入 $3 = 8 - 1 \times 5$	$1 = 2 \times (8 - 5) - 5 = 2 \times 8 - 3 \times 5$
代入 $5 = 13 - 1 \times 8$	$1 = 2 \times 8 - 3 \times (13 - 8) = 5 \times 8 - 3 \times 13$
代入 $8 = 60 - 4 \times 13$	$1 = 5 \times (60 - 4 \times 13) - 3 \times 13 = \mathbf{5 \times 60 - 23 \times 13}$

步骤 3：取模 60 得到 d

由 1 = 5×60 − 23×13，两边对 60 取模：$1 \equiv -23 \times 13 \pmod{60}$ 即 $13 \times (-23) \equiv 1 \pmod{60}$　−23 为负数，正化：d = −23 + 60 = **37**

d = 37
验证：$13 \times 37 = 481 = 8 \times 60 + 1$，即 $13 \times 37 \equiv 1 \pmod{60}$ ✓

RSA 完整运行范例（PPT 例题）

e=3, p=5, q=11, M=14 — 完整密钥生成 + 加密 + 解密 + 验证

◆ PPT 例题：RSA 完整流程演示

【密钥生成】

步骤	计算
选 p, q	p = 5, q = 11
计算 n	n = p × q = 5 × 11 = **55**
计算 Φ(n)	Φ(n) = (5−1)(11−1) = 4 × 10 = **40**
选 e	e = 3（gcd(3, 40) = 1 ✓）
计算 d	$d = e^{-1} \bmod \Phi(n) = 3^{-1} \bmod 40 = \mathbf{27}$ 验证：3 × 27 = 81 = 2×40 + 1 ≡ 1 (mod 40) ✓
公开密钥	**k_1 = (e, n) = (3, 55)**
私人密钥	**k_2 = (d, n) = (27, 55)**

【加密 M = 14】

$C = M^e \bmod n = 14^3 \bmod 55$

步骤	计算	结果
14^2	196 = 3×55 + 31	196 mod 55 = **31**
$14^3 = 14^2 \times 14$	31 × 14 = 434 = 7×55 + 49	434 mod 55 = **49**

密文 C = 49

【解密 C = 49】

$M' = C^d \bmod n = 49^{27} \bmod 55$。用**反复平方法（快速幂取模）**：

$27 = 16 + 8 + 2 + 1$，所以 $49^{27} = 49^{16} \times 49^8 \times 49^2 \times 49^1$

幂次	计算	mod 55 结果
49^1	49	**49**
49^2	$49^2 = 2401 = 43 \times 55 + 36$	**36**
$49^4 = (49^2)^2$	$36^2 = 1296 = 23 \times 55 + 31$	**31**
$49^8 = (49^4)^2$	$31^2 = 961 = 17 \times 55 + 26$	**26**
$49^{16} = (49^8)^2$	$26^2 = 676 = 12 \times 55 + 16$	**16**

组合：

```
49²⁷ mod 55 = 49¹⁶ × 49⁸ × 49² × 49¹ (mod 55) = 16 × 26 × 36 × 49 (mod 55) 逐步乘后取模： 16 × 26 = 416 = 7×55 + 31 → 31 31 × 36 = 1116 = 20×55 + 16 → 16 16 × 49 = 784 = 14×55 + 14 → 14
```

恢复明文 M' = 14 = M ✓

◆ 考点 7：RSA 的安全性

核心：RSA 的安全性**依赖于大整数分解的难度**。

- 公开 n = pq 和 e，但 p、q 保密
- 要从 (e, n) 反推 d，必须先分解 n 求出 Φ(n) = (p−1)(q−1)，再算 $d = e^{-1} \bmod \Phi(n)$
- 大整数分解是 NP 难度问题，目前没有多项式时间算法

三种可能的攻击方式：

攻击方式	说明
大整数分解	129 位十进制数已被分布式计算分解，所以 n 应远大于此（通常 2048 位以上）
时间攻击	根据加密运行时间反推私钥内容（侧信道攻击）
密钥分配攻击	公钥分发过程被"中间人"取代，攻击者用自己的公钥替换

◆ 考点 8：RSA 的用途

用途	使用方式	作用
保密通信	发送方用接收方**公钥**加密 接收方用自己**私钥**解密	实现机密性
数字签名	发送方用自己**私钥**加密 接收方用发送方**公钥**验证	实现抗否认性、身份认证

◆ 第 14 章核心知识总结表

考点	关键内容
信息安全 4 目标	机密性（加密）、完整性（消息摘要）、抗否认性（数字签名）、可用性
密码学 2 分支	密码编码学（建立体制）、密码分析学（破译体制）
对称体制	同一密钥，快但分发难，代表 DES
非对称体制	公钥 + 私钥，慢但无需秘密信道，代表 RSA
欧拉函数	$\Phi(p) = p - 1$；$\Phi(pq) = (p-1)(q-1)$
欧拉定理	$\gcd(a, n)=1$ 时 $a^{\Phi(n)} \equiv 1 \pmod n$；推论：$m^{k\Phi(n)+1} \equiv m \pmod n$
费马小定理	n 为素数时 $a^{n-1} \equiv 1 \pmod n$
RSA 密钥	公钥 (e, n)，私钥 (d, n)；$ed \equiv 1 \pmod{\Phi(n)}$
RSA 公式	加密 $C = M^e \bmod n$；解密 $M = C^d \bmod n$
RSA 安全性	基于大整数分解的难度（NP 难度问题）
常见题型	① 给 e, n, M 求 C（按模计算） ② 给 e, Φ(n) 求 d（扩展欧几里得算法） ③ 证明 RSA 解密正确性（用欧拉定理推论）

📌 **考前速记口诀：**
- **四目标**：机、完、抗、可
- **两分支**：编码学（建）+ 分析学（破）
- **RSA 五步**：选 p,q → 算 n,Φ(n) → 选 e → 求 d → 公开 (e,n)，保密 (d,n)
- **加解密**：加用 e（公钥）、解用 d（私钥）
- **安全性**：基于"**大整数分解难**"

第 10 章 NP 完全问题 — 考点速览（5-10%）

P 类 | NP 类 | 多项式约化 | NP 难度 | NP 完全

◆ 考点 1：多项式时间算法 vs 指数时间算法

算法类型	定义与特点
多项式时间 Polynomial Time	规模为 n 的输入，最坏情况运行时间为 $O(n^k)$（k 为常数） **例：** Prim 的 $O(n^3)$、Dijkstra 的 $O(n^2)$、Floyd 的 $O(n^3)$ **认为"很好地解决"了问题**
指数时间 Exponential Time	规模为 n 的输入，最坏情况运行时间为 $O(k^n)$ **例：** 回溯法、分枝限界法、n-皇后问题 **实用性差，规模大时不可行**

◆ 考点 2：易处理 vs 难处理问题（★ 考题 10-6 关键）

问题分类	定义
易处理问题 Tractable	能在**多项式时间**求解的问题 典型代表：排序、最短路径、最小生成树等
难处理问题 Intractable	至今**尚未找到多项式时间算法**求解的问题 对于指数时间算法，规模大时失去实用性 典型代表：TSP、背包问题、n-皇后等
不可判定问题 Undecidable	无论消耗多少时间和空间也**不能求解**的问题 例："停机问题"——判断程序是否会终止 **本章不讨论**

★ **关键认识：**难处理问题虽未找到多项式算法，但也未证明其不存在。这些问题在**计算上相关**，称为 NP 难度或 NP 完全问题。

P 类与 NP 类 — 核心概念

定义 | 关系 | 判定问题

◆ 考点 3：P 类和 NP 类问题（★★★ 必考）

【1. 定义】

P 类问题：所有可在**多项式时间内用确定算法**求解的**判定问题**的集合。

NP 类问题（Non-Deterministic Polynomial）：所有可在**多项式时间内用不确定算法**求解的**判定问题**的集合。

判定问题：只要求产生"0"或"1"作为输出的问题（是/否问题）。

【2. 不确定算法的三个新函数】

函数	作用
Choice(S)	从集合 S 中任意选择一个元素（**不确定选择**）
Success()	发出成功完成信号后算法终止
Failure()	发出不成功完成信号后算法终止

关键特征：不确定算法时间 = 任意一个选择序列导致**成功完成**的最少时间（如不存在成功路径则为 O(1)）

【3. P 与 NP 的关系】

$$\mathbf{P} \subseteq \mathbf{NP}$$

因为确定算法是不确定算法当 Choice 函数只有一种选择时的特例

P = NP 问题：当今计算机科学最大的开放问题。
普遍猜想：P ≠ NP（即存在 NP 类问题不能在多项式时间内求解）

◆ 考点 4：多项式约化（★★ 核心概念，考题 10-6 重点）

【定义 10-3：多项式约化 $Q_1 \propto Q_2$】

若存在确定算法 A 求解 Q_1，算法 A 以**多项式时间**调用求解 Q_2 的确定算法 B，且**不计 B 的工作量**，A 仍是多项式时间的，则称 **Q_1 约化为 Q_2**，记作 **$Q_1 \propto Q_2$**。

图示逻辑： Q_1 的求解 ↓ 调用 Q_2 算法 B （多项式时间调用） ↓ 若 B 也是多项式时间的 ↓ 则 Q_1 就能多项式时间求解

【重要性质】

性质 10-1：若 $Q_1 \propto Q_2$ 且 $Q_2 \in P$，则 $Q_1 \in P$
性质 10-2（传递性）：若 $Q_1 \propto Q_2$ 且 $Q_2 \propto Q_3$，则 $Q_1 \propto Q_3$

考试意义：如果能证明问题 Q_1 能约化到已知的 NP 难度问题 Q，就能证明 Q_1 也是 NP 难度的。

NP 难度与 NP 完全（★★★ 必考）

考题 10-7 直考 | Cook 定理 | 证明步骤（考题 10-8）

◆ 考点 5：NP 难度和 NP 完全问题（定义 10-4、10-5）

【定义 10-4：NP 难度（NP Hard）】

对于问题 Q，**任意问题 $Q_1 \in NP$，都有 $Q_1 \propto Q$**，则称 Q 是 **NP 难度**的。

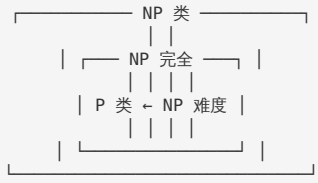
解读：所有 NP 类问题都能约化到 Q → Q "至少一样难"

【定义 10-5：NP 完全（NP Complete）】

若问题 Q ∈ NP 且 Q 是 NP 难度的，则称 Q 是 **NP 完全**的。

> NP 完全 = NP 类 ∩ NP 难度
> （既要是 NP 类的，又要是 NP 难度的）

【关系图】



- 所有 NP 完全问题都是 NP 难度的，但**反之不然**
- NP 难度问题不一定是 NP 类的（可能超出 NP）
- 若 NP 难度问题是 NP 类的 → NP 完全

◆ 考点 6：Cook 定理（第一个 NP 完全问题）

斯蒂芬·库克（Steven Cook，1971）：证明了第一个 NP 完全问题。

> **定理 10-1 (Cook 定理)**：
> 可满足性问题在 P 中，当且仅当 P = NP
> **定理 10-2**：CNF 可满足性问题是 NP 完全的

历史意义：此后至少 300+ 个问题被证明为 NP 完全问题，但尚未证明其中任何一个属于 P。

◆ 考点 7：证明问题是 NP 完全的一般步骤（★ 考题 10-8）

要证明问题 Q 是 NP 完全的，需要以下 6 步：

1. 选择一个**已知**的 NP 难度问题 Q_1（如 CNF 可满足性、最大集团等）
2. **证明能从 Q_1 的实例 I_1 在多项式时间内构造 Q 的实例 I**
3. **证明能从 I 的解在多项式时间内确定 I_1 的解**
4. 从第 2、3 步结论：$Q_1 \propto Q$
5. 由约化的**传递性**：所有 NP 类问题都能约化到 Q，故 Q 是 NP 难度的
6. **若 Q ∈ NP**（能用不确定算法多项式时间求解），则 **Q 是 NP 完全的**

> **★ 关键点（考题 10-8）：**证明约化 $Q_1 \propto Q$ 需要两个方向：
> ・构造方向：$I_1 \rightarrow I$（多项式时间）
> ・反推方向：I 的解 → I_1 的解（多项式时间）

典型 NP 完全问题（考题 10-9）

最大集团 | 顶点覆盖 | 3SAT | 图着色

◆ 典型问题 1：最大集团判定问题（定理 10-3）

【概念】

集团（完全子图）：无向图 G=(V,E) 的顶点子集 S ⊆ V，使得 S 中任意两个顶点都相邻（(u,v) ∈ E）。
最大集团问题：求 G 的最大集团规模 k
最大集团判定问题 (G,k)：判定 G 是否存在规模至少为 k 的集团

【证明其为 NP 完全】

定理 10-3：CNF 可满足性 ∝ 最大集团判定问题

证明思路（2 步）：

步骤	内容
第一步	从任意 CNF 公式 F 构造无向图 G=(V,E)（多项式时间内） 顶点集 V = {(l, i) | l 是子句 C_i 的一个文字} 边集 E = {((l, i), (l', j)) | i≠j 且 l≠-l'}
第二步	**关键：**F 可满足 ⟺ G 有规模 ⩾k 的集团 其中 k = CNF 的子句个数

★ 考题 10-9 问这个问题的约化！

◆ 典型问题 2：顶点覆盖判定问题（定理 10-4）

顶点覆盖：无向图 G 的顶点子集 S ⊆ V，使得 E 中所有边至少有一个顶点在 S 中
顶点覆盖判定问题：判定 G 是否存在规模至多为 k 的顶点覆盖

定理 10-4：最大集团判定问题 ∝ 顶点覆盖判定问题

核心构造：将 G 的边集互补成 G'，则：
G 有规模 ⩾k 的集团 ⟺ G' 有规模 ⩽n-k 的顶点覆盖

◆ 典型问题 3：3SAT（3 元 CNF 可满足性，定理 10-5）

3SAT：每个子句恰好包含 3 个文字的 CNF 可满足性问题
例：$(x_1 \vee x_2 \vee x_3) \wedge (\bar{x_1} \vee x_2 \vee \bar{x_3})$

定理 10-5：CNF 可满足性 ∝ 3SAT
关键转换技巧：用 3 个辅助结论将任意 CNF 转换成 3SAT（多项式时间）

结论：3SAT 是 NP 完全的

◆ 典型问题 4：图着色数判定问题（定理 10-6）

图着色：对图的顶点着色，使任意相邻顶点颜色不同
着色数判定问题：判定图 G 是否能用 k 种颜色着色

定理 10-6：3SAT ∝ 着色数判定问题
核心构造：从 n 个变量的 3SAT 公式 F 构造图 G，使
F 可满足 ⟺ G 是 (n+1) 可着色的

结论：图着色数判定问题是 NP 完全的

◆ NP 完全问题树

学习意义与核心总结

◆ 为什么要学习 NP 完全问题?

- **实践意义：**如果问题已被证明是 NP 难度的，恐怕很难指望找到多项式时间有效算法
- **算法选择：**对于 NP 难度问题，应该采用以下策略而非穷举：
  - 启发式算法（分枝限界法、贪心法）
  - 随机算法
  - 近似算法
- **理论意义：**P = NP 是千禧年七大数学难题之一，$100 万奖金

◆ 第 10 章核心知识总结表

考点	关键内容
多项式 vs 指数	$O(n^k)$ 易处理，$O(k^n)$ 难处理
P 类	多项式时间确定算法求解的判定问题
NP 类	多项式时间不确定算法求解的判定问题
P ⊆ NP	关系；P = NP 未证明（猜想 P ≠ NP）
多项式约化 $Q_1 \propto Q_2$	Q_2 多项式可解 $\Longrightarrow$ Q_1 多项式可解
NP 难度	所有 NP 类问题都能约化到它
NP 完全	NP 类 ∩ NP 难度
Cook 定理	CNF 可满足性是第一个 NP 完全问题
证明步骤（10-8）	选已知问题 → 构造 → 反推 → 约化 → 难度 → 完全
集团 ∝ 覆盖（10-9）	G 有大小 k 的集团 $\Longleftrightarrow$ G' 有大小 n-k 的覆盖
典型 NP 完全问题	最大集团、顶点覆盖、3SAT、图着色等 1000+ 个

📌 考前速记口诀：
· 多项式易处理，指数难处理
· P 类确定解，NP 类不确定算
· 约化是关键，$Q_1 \propto Q_2$ 传递
· 难度 + NP 类 = 完全
· 一个难，全难；一个易，全易（待证明）