题目 1:

(a) 算法描述

动态规划方法:

定义数组 dp[0..n]，其中 dp[i]表示长度为 i 的铜管能获得的最大价值。

- 初始条件：dp[0]=0（长度为 0 时价值为 0）
- 递推关系：

$$dp[i] = \max_{\substack{1 \le j \le i \\ j \le n}} \{p[j] + dp[i-j]\}$$

其中 j 表示切割的第一段长度，剩余部分 i−j 递归求解。

伪代码:

```
function cutRod(p, n):
    dp[0] = 0
    for i from 1 to n:
        max_val = -∞
        for j from 1 to min(i, n):   // j 不超过当前长度 i 和最大长度 n
            if j <= i and j <= n:     // 确保 j 有效
                current_val = p[j] + dp[i - j]
                if current_val > max_val:
                    max_val = current_val
        dp[i] = max_val
    return dp[n]
```

(b) 时间复杂度分析

- 外层循环遍历长度 i 从 1 到 n，共 n 次。
- 内层循环遍历 j 从 1 到 min(i,n)，最坏情况下为 O(n)次。
  总时间复杂度为 O(n2)

题目 2:

分支限界法求解:

1. **单位价值排序**（降序）:
   - 物品 1: 40/4=10.040/4=10.0
   - 物品 2: 42/7=6.042/7=6.0
   - 物品 3: 25/5=5.025/5=5.0
   - 物品 4: 12/3=4.012/3=4.0
2. **解空间树与搜索过程**:
   - 优先级队列按上界（分数背包最优值）排序。
   - 上界计算：当前价值 + 剩余容量装剩余物品的最大单位价值部分。

搜索过程:

- **节点 0**（根节点）:

重量 =0, 价值 =0, 上界 =0+10×10.0=100 重量 =0, 价值 =0, 上界 =0+10×10.0=100

- **扩展节点 0**（考虑物品 1）:
  - 选物品 1：重量 = 4，价值 = 40，上界 = 40 + 6 × 6.0 = 76
  - 不选物品 1：重量 = 0，价值 = 0，上界 = 0 + 10 × 6.0 = 60
- **扩展上界最大的节点（选物品 1）**:
  - 考虑物品 2（不可行，重量 4+7>10）
  - 不选物品 2：重量 = 4，价值 = 40，上界 = 40 + 6 × 5.0 = 70
- **扩展节点（不选物品 1）**：上界 60 < 70，剪枝。
- **扩展节点（选物品 1，不选物品 2）**:
  - 选物品 3：重量 = 9，价值 = 65，上界 = 65 + 1 × 4.0 = 69
  - 不选物品 3：重量 = 4，价值 = 40，上界 = 40 + 6 × 4.0 = 64
- **扩展节点（选物品 1 和 3）**:
  - 选物品 4：不可行（重量 12 > 10）
  - 不选物品 4：重量 = 9，价值 = 65（可行解）
- 其他节点上界均 ≤ 65，剪枝。

3. **结果**:
   最优解为选择物品 1 和 3，总价值 **65**（重量 4+5=9 ≤ 10）。

题目 3:

步骤 1: 初始化

- 从节点 1 开始。
- 记录当前路径和路径长度。
- 初始化一个优先队列（最小堆），用于存储部分路径和它们的路径长度。

步骤 2: 构建初始路径

- 从节点 1 开始，选择下一个路径长度最小的节点，例如选择节点 2（距离为 3）。
- 更新当前路径为 [1, 2]，路径长度为 3。

步骤 3: 探索路径

- 从节点 2 继续探索，选择下一个路径长度最小的节点，例如节点 4（距离为 2）。
- 更新当前路径为 [1, 2, 4]，路径长度为 5。

步骤 4: 继续探索

- 从节点 4 继续，选择下一个路径长度最小的节点，例如节点 5（距离为 5）。
- 更新当前路径为 [1, 2, 4, 5]，路径长度为 10。

步骤 5: 检查和更新最优解

- 继续这个过程，直到所有节点都被访问。
- 在每一步，检查当前路径是否是完整的回路（即回到起点）。
- 如果是回路，计算总长度，并与当前已知的最短路径长度比较，更新最优解。

步骤 6: 回溯和剪枝

- 在探索过程中，如果发现当前路径的总长度已经超过已知的最短路径长度，可以剪枝，即不再继续探索该路径。
- 使用优先队列来存储和选择最有希望的路径进行探索。

步骤 7: 完成搜索

- 继续这个过程，直到所有可能的路径都被探索或剪枝。

- 最终，优先队列中剩余的路径即为最优解。

最优解
通过上述步骤，我们可以找到最优路径。在这个图中，最优路径是 [1, 2, 4, 5, 3, 1]，总长度为 22。

题目 4:

线性规划问题:
标准型（引入松弛变量）:

$$\begin{aligned} \max \quad & Z = 2x_1 + 3x_2 + 0s_1 + 0s_2 + 0s_3 + 0s_4 \\ \text{s.t.} \quad & 2x_1 + 2x_2 + s_1 = 12 \\ & x_1 + 2x_2 + s_2 = 8 \\ & 4x_1 + s_3 = 16 \\ & 4x_2 + s_4 = 12 \\ & x_1, x_2, s_1, s_2, s_3, s_4 \geq 0 \end{aligned}$$

单纯形表迭代:

1. **初始表**:

基	**x1**	**x2**	**s1**	**s2**	**s3**	**s4*s***	**解**
s1	2	2	1	0	0	0	12
s2	1	2	0	1	0	0	8
s3	4	0	0	0	1	0	16
s4	0	4	0	0	0	1	12
Z	-2	-3	0	0	0	0	0

   - 进基变量：x2（最小系数 -3）
   - 出基变量：s4（最小比率 12/4=3）

2. **迭代 1**（主元 x2 列，s4 行）:
   - 更新 s4 行：[0,1,0,0,0,0.25,3][0,1,0,0,0,0.25,3]
   - 更新其他行:
     - s1: 2x1+2x2+s1−2 × new s4=6 → [2,0,1,0,0,−0.5,6]$2x1+2x2+s1-2\times$new $s4$=6→[2,0,1,0,0,−0.5,6]
     - s2: x1+2x2+s2−2 × new s4=2 → [1,0,0,1,0,−0.5,2]$x1+2x2+s2-2\times$new $s4$=2→[1,0,0,1,0,−0.5,2]
     - s3: 不变
     - Z: Z−(−3) × new s4=9 → [−2,0,0,0,0,0.75,9]$Z-(-3)\times$new $s4$=9→[−2,0,0,0,0,0.75,9]
       | 基| x1| x2| s1| s2| s3| s4| 解 |
       | s1 | 2 | 0 | 1 | 0 | 0 | -0.5 | 6 |

| s2 | 1 | 0 | 0 | 1 | 0 | -0.5 | 2 |

| s3 | 4 | 0 | 0 | 0 | 1 | 0 | 16 |

| x2 | 0 | 1 | 0 | 0 | 0 | 0.25 | 3 |

| Z | -2 | 0 | 0 | 0 | 0 | 0.75 | 9 |

- ○ 进基变量：x1（系数 -2）
- ○ 出基变量：s2（最小比率 2/1=2）

3. **迭代 2**（主元 x1 列，s2 行）：
   - ○ 更新 s2 行：[1,0,0,1,0,−0.5,2][1,0,0,1,0,−0.5,2]
   - ○ 更新其他行：
     - ▪ s1: 2x1+s1−2 × new s2=2 → [0,0,1,−2,0,0.5,2]$2x1+s1-2\times \text{new } s2=2\rightarrow[0,0,1,-2,0,0.5,2]$
     - ▪ s3: 4x1+s3−4 × new s2=8 → [0,0,0,−4,1,2,8]$4x1+s3-4\times \text{new } s2=8\rightarrow[0,0,0,-4,1,2,8]$
     - ▪ x2：不变
     - ▪ Z: Z−(−2) × new s2=13 → [0,0,0,2,0,−0.25,13]$Z-(-2)\times \text{new } s2=13\rightarrow[0,0,0,2,0,-0.25,13]$

       |基 | x1| x2| s1| s2| s3| s4| 解 |

       | s1 | 0 | 0 | 1 | -2 | 0 | 0.5 | 2 |

       | x1 | 1 | 0 | 0 | 1 | 0 | -0.5 | 2 |

       | s3 | 0 | 0 | 0 | -4 | 1 | 2 | 8 |

       | x2 | 0 | 1 | 0 | 0 | 0 | 0.25 | 3 |

       | Z | 0 | 0 | 0 | 2 | 0 | -0.25 | 13 |
   - ○ 进基变量：s4$s4$（系数 -0.25）
   - ○ 出基变量：s1$s1$（最小比率 2/0.5=4）

4. **迭代 3**（主元 s4$s4$ 列，s1$s1$ 行）：
   - ○ 更新 s1$s1$ 行：[0,0,2,−4,0,1,4][0,0,2,−4,0,1,4]
   - ○ 更新其他行：
     - ▪ x1$x1$: x1+s2−(−0.5) × new s1=4 → [1,0,1,−1,0,0,4]$x1+s2-(-0.5)\times \text{new } s1=4\rightarrow[1,0,1,-1,0,0,4]$
     - ▪ s3$s3$: s3−2 × new s1=0 → [0,0,−4,4,1,0,0]$s3-2\times \text{new } s1=0\rightarrow[0,0,-4,4,1,0,0]$
     - ▪ x2$x2$: x2−0.25 × new s1=2 → [0,1,−0.5,1,0,0,2]$x2-0.25\times \text{new } s1=2\rightarrow[0,1,-0.5,1,0,0,2]$
     - ▪ ZZ: Z−(−0.25) × new s1=14 → [0,0,0.5,1,0,0,14]$Z-(-0.25)\times \text{new } s1=14\rightarrow[0,0,0.5,1,0,0,14]$

       | 基 | x1$x1$ | x2$x2$ | s1$s1$ | s2$s2$ | s3$s3$ | s4$s4$ | 解 |

       |------|----------|----------|----------|----------|----------|----------|-----|

       | s4$s4$ | 0 | 0 | 2 | -4 | 0 | 1 | 4 |

       | x1$x1$ | 1 | 0 | 1 | -1 | 0 | 0 | 4 |

       | s3$s3$ | 0 | 0 | -4 | 4 | 1 | 0 | 0 |

       | x2$x2$ | 0 | 1 | -0.5 | 1 | 0 | 0 | 2 |

       | ZZ | 0 | 0 | 0.5 | 1 | 0 | 0 | 14 |
   - ○ ZZ 行系数均非负，达到最优解。

最优解:
x1=4, x2=2, Z=2×4+3×2=14

题目 5:

(a) 基础情况

- f(1,m): 只有 1 层，只需测试 1 次（m⩾1）→ **1**
- f(0,m): 0 层，无需测试 → **0**
- f(n,1): 只有 1 个鸡蛋，需线性扫描，最坏情况测试 n 次 → **n**
- f(n,0): 无鸡蛋，无法测试（n>0）→ **∞**（或未定义）

(b) 从第 h*h* 层扔鸡蛋

- **鸡蛋破裂**：安全楼层在 [0,h−1]，剩余鸡蛋 m−1，次数为 f(h−1,m−1).
- **鸡蛋未破**：安全楼层在 [h,n]，剩余鸡蛋 m，次数为 f(n−h,m)
- **最坏情况**：max(f(h−1,m−1),f(n−h,m))
- **总次数**：1+max(f(h−1,m−1),f(n−h,m)).

(c) 递推关系

$$f(n,m) = \min_{1\le h\le n} \{1 + \max(f(h-1,m-1), f(n-h,m))\}$$

边界条件:

- f(0,m)=0
- f(n,0)=∞（n>0)
- f(1,m)=1（m⩾1)
- f(n,1)=n

(d) 时间复杂度
状态数 O(n×m)，每状态计算 O(n) → O(n2m).
(e) 空间复杂度
二维表存储 f(n,m) → O(nm).
(f) 空间优化
优化方法:
用一维数组 dp_prev[0..n] 存储 m−1 的结果，dp_curr[0..n] 计算当前 m.

- 空间复杂度降至 O(n).

递推过程:

```
# 初始化
dp_prev = [0] + [float('inf')] * n    # m=0 的情况
for m_val in range(1, m+1):
    dp_curr = [0]    # dp_curr[0] = 0
    for i in range(1, n+1):
        min_drops = float('inf')
        for h in range(1, i+1):
            cost = 1 + max(dp_prev[h-1], dp_curr[i-h])
            min_drops = min(min_drops, cost)
```

```
        dp_curr.append(min_drops)
    dp_prev = dp_curr
return dp_prev[n]
```