1

```python
def fractional_knapsack(P, W, M):
    # P: 物品价值列表
    # W: 物品重量列表
    # M: 背包最大容量
    n = len(P)

    # 计算每个物品的单位重量价值
    items = []
    for i in range(n):
        items.append((P[i], W[i], P[i] / W[i]))  # (价值, 重量, 单位重量价值)

    # 按单位重量价值排序，降序排列
    items.sort(key=lambda x: x[2], reverse=True)

    total_value = 0.0  # 背包内的总价值
    remaining_capacity = M  # 背包剩余容量
    X = [0] * n  # 存储每个物品选择的份额（0-1之间的值）

    for i in range(n):
        value, weight, unit_value = items[i]

        if weight <= remaining_capacity:
            # 当前物品可以完全放入背包
            X[i] = 1
            total_value += value
            remaining_capacity -= weight
        else:
            # 当前物品只能部分放入背包
            X[i] = remaining_capacity / weight  # 选择物品的部分
            total_value += X[i] * value
            remaining_capacity = 0  # 背包已满

        if remaining_capacity == 0:
            break  # 背包已满，退出循环

    return total_value, X  # 返回总价值和每个物品的选择份额
```

算法复杂度分析：

1. **计算单位重量价值**：计算每个物品的单位重量价值的时间复杂度为 $O(n)$。
2. **排序**：排序操作的时间复杂度是 $O(n \log n)$。
3. **遍历物品**：遍历所有物品，时间复杂度为 $O(n)$。

因此，整体的时间复杂度为 $O(n \log n)$，其中 n 是物品的数量。

2

```python
def min_coins_to_make_amount(target_amount, coin_values):
    # 将目标金额转换为整数（以美分为单位）
    target_amount_in_cents = int(target_amount * 100)

    # 排序硬币面值，从大到小
    coin_values.sort(reverse=True)

    # 存储每种硬币的数量
    coin_counts = {}

    # 对每个硬币面值，使用尽可能多的硬币
    for coin in coin_values:
        coin_value_in_cents = coin * 100  # 将硬币面值转换为美分
        if target_amount_in_cents >= coin_value_in_cents:
            num_coins = target_amount_in_cents // coin_value_in_cents  # 可以使用多少个当前面值的硬币
            coin_counts[coin] = num_coins
            target_amount_in_cents -= num_coins * coin_value_in_cents  # 更新剩余金额

        if target_amount_in_cents == 0:
            break  # 如果已经凑足目标金额，结束循环

    return coin_counts

# 主函数：计算最少硬币数
def main():
    target_amount = 3.33  # 目标金额为3.33美元
    coin_values = [0.25, 0.10, 0.05, 0.01]  # 可用的硬币面值
    result = min_coins_to_make_amount(target_amount, coin_values)

    # 输出硬币数量
    print("使用的硬币数量：")
    for coin, count in result.items():
        print(f"{coin}美元：{count}个")

# 调用主函数
main()
```

算法复杂度分析：

1. **排序硬币面值**：排序面值的时间复杂度是 $O(k \log k)$，其中 k 是硬币种类的数量（在本题中为4）。
2. **遍历硬币面值**：对于每个硬币，我们需要计算能够使用多少个该硬币，这个操作是 $O(k)$。
3. **总时间复杂度**：由于排序和遍历硬币的复杂度都取决于硬币种类的数量，因此总的时间复杂度是 $O(k \log k)$。

由于硬币种类固定为4种，复杂度是常数级别的 $O(1)$，在实际应用中非常高效。