练习1 – 找零钱

- leetcode_322_零钱兑换：https://leetcode-cn.com/problems/coin-change/

- 假设有25分、20分、5分、1分的硬币，现要找给客户41分的零钱，如何办到硬币个数最少?
  - 此前用贪心策略得到的并非是最优解（贪心得到的解是 5 枚硬币）

- 假设 dp(n) 是凑到 n 分需要的最少硬币个数
  - 如果第 1 次选择了 25 分的硬币，那么 dp(n) = dp(n – 25) + 1
  - 如果第 1 次选择了 20 分的硬币，那么 dp(n) = dp(n – 20) + 1
  - 如果第 1 次选择了 5 分的硬币，那么 dp(n) = dp(n – 5) + 1
  - 如果第 1 次选择了 1 分的硬币，那么 dp(n) = dp(n – 1) + 1
  - 所以 dp(n) = min { dp(n – 25), dp(n – 20), dp(n – 5), dp(n – 1) } + 1

小码哥教育 SEEMYGO

找零钱 – 暴力递归

```
int coins(int n) {
    if (n < 1) return Integer.MAX_VALUE;
    if (n == 1 || n == 5 || n == 20 || n == 25) return 1;
    int min1 = Math.min(coins(n - 1), coins(n - 5));
    int min2 = Math.min(coins(n - 20), coins(n - 25));
    return Math.min(min1, min2) + 1;
}
```

- 类似于斐波那契数列的递归版，会有大量的重复计算，时间复杂度较高

小码哥教育 SEEMYGO

找零钱 - 记忆化搜索

```
int coins(int n) {
    if (n < 1) return -1;
    int[] dp = new int[n + 1];
    int[] faces = {1, 5, 20, 25};
    for (int face : faces) {
        if (n < face) break;
        dp[face] = 1;
    }
    return coins(n, dp);
}

static int coins(int n, int[] dp) {
    if (n < 1) return Integer.MAX_VALUE;
    if (dp[n] == 0) {
        int min1 = Math.min(coins(n - 25, dp), coins(n - 20, dp));
        int min2 = Math.min(coins(n - 5, dp), coins(n - 1, dp));
        dp[n] = Math.min(min1, min2) + 1;
    }
    return dp[n];
}
```

找零钱 - 递推

```
int coins(int n) {
    if (n < 1) return -1;
    int[] dp = new int[n + 1];
    for (int i = 1; i <= n; i++) {
        int min = dp[i - 1];
        if (i >= 5) min = Math.min(dp[i - 5], min);
        if (i >= 20) min = Math.min(dp[i - 20], min);
        if (i >= 25) min = Math.min(dp[i - 25], min);
        dp[i] = min + 1;
    }
    return dp[n];
}
```

- 时间复杂度、空间复杂度：O(n)

小码哥教育 SEEMYGO

思考题：请输出找零钱的具体方案（具体是用了哪些面值的硬币）

```
int coins(int n) {
    if (n < 1) return -1;
    int[] dp = new int[n + 1];
    int[] faces = new int[dp.length];
    for (int i = 1; i <= n; i++) {
        int min = dp[i - 1];
        faces[i] = 1;
        if (i >= 5 && dp[i - 5] < min) {
            min = dp[i - 5];
            faces[i] = 5;
        }
        if (i >= 20 && dp[i - 20] < min) {
            min = dp[i - 20];
            faces[i] = 20;
        }
        if (i >= 25 && dp[i - 25] < min) {
            min = dp[i - 25];
            faces[i] = 25;
        }
        dp[i] = min + 1;
    }
    print(faces, n);
    return dp[n];
}
```

```
void print(int[] faces, int i) {
    while (i > 0) {
        System.out.print(faces[i] + " ");
        i -= faces[i];
    }
    System.out.println();
}
```

小码哥教育 SEEMYGO

找零钱 - 通用实现

```
int coins(int n, int[] faces) {
    if (n < 1 || faces == null || faces.length == 0) return -1;
    int[] dp = new int[n + 1];
    for (int i = 1; i <= n; i++) {
        int min = Integer.MAX_VALUE;
        for (int face : faces) {
            if (i < face) continue;
            if (dp[i - face] < 0 || dp[i - face] >= min) continue;
            min = dp[i - face];
        }
        if (min == Integer.MAX_VALUE) {
            dp[i] = -1;
        } else {
            dp[i] = min + 1;
        }
    }
    return dp[n];
}
```