

G4-11 课后练习 —— 递推算法

姓名： ____ 日期： ____

>

回家先念三遍口诀：

"斐波那契兔子数列，前两个一后面加前！"

"递推往前推，递归往回拆，递推快如飞，递归慢慢来！"

一、基础练习（必做）

1.1 填空题

- (1) 递推的核心思想是：从 ____ 出发，根据 ____ 一步步推出后面的结果。
- (2) 斐波那契数列的递推公式是： $f(n) = \text{ } + \text{ }$ ，其中 $f(1) = \text{ }$ ， $f(2) = \text{ }$ 。
- (3) 爬楼梯问题（每次走1步或2步）的递推公式也是： $f(n) = \text{ } + \text{ }$ ，但初始条件不同： $f(1) = \text{ }$ ， $f(2) = \text{ }$ 。
- (4) 斐波那契数列的第12项是 ____。
- (5) 递推和递归相比， ____ 更快，因为 ____。

1.2 选择题

- (1) 斐波那契数列的第1项和第2项分别是？
 - A. 0, 1
 - B. 1, 1
 - C. 1, 2
 - D. 2, 3
- (2) 在斐波那契数列中， $f(8)$ 等于多少？(1,1,2,3,5,8,13...)
 - A. 13
 - B. 21

- C. 34
- D. 8

(3) 爬楼梯问题中，到第5阶有几种走法（每次1或2步）？

- A. 5
- B. 8
- C. 13
- D. 3

(4) 下面哪个不是递推算法需要的要素？

- A. 初始条件
- B. 递推公式
- C. 循环次数
- D. 随机数

(5) 用递推法计算f(50)和用递归法计算f(50)，哪个更快？

- A. 递推法
- B. 递归法
- C. 一样快
- D. 都算不出来

1.3 数列推导题

题目1：已知 $f(1)=2, f(2)=3, f(n)=f(n-1)+f(n-2)$ ，写出前8项。

n	1	2	3	4	5	6	7	8
f(n)	2	3						

题目2：已知 $f(1)=1, f(2)=3, f(n)=f(n-1)-f(n-2)$ ，写出前8项。

n	1	2	3	4	5	6	7	8
f(n)	1	3						

题目3：你能发现题目2中的规律吗？

1.4 应用题

题目：小明存零花钱。第1天存1元，第2天存1元，从第3天开始，每天存的钱是前两天的和。

请问第10天小明要存多少钱？10天一共存了多少钱？

解题思路：
这就是斐波那契数列！

天数	1	2	3	4	5	6	7	8	9	10
存钱	1	1	2	3	5					

- 第10天存： ____ 元
 - 10天总共存： ____ 元（提示：把所有天的钱加起来）
-

二、编程练习（必做）

2.1 写斐波那契函数

```
def my_fibonacci(n):  
    """  
    计算斐波那契数列的第n项  
    用递推法实现  
    """  
  
    if n <= 0:  
        return 0  
    if n == 1 or n == 2:  
        return 1  
  
    # 在这里用循环实现递推  
    a, b = 1, 1  
    for i in range(3, n + 1):  
        # 你的代码  
        pass  
  
    return b  
  
# 测试  
for i in range(1, 15):  
    print(f"f({i}) = {my_fibonacci(i)}")
```

2.2 写爬楼梯函数

```
def my_climb_stairs(n):  
    """  
    爬楼梯：每次走1步或2步  
    """  
  
    if n == 1:  
        return 1  
    if n == 2:
```

```
return 2
```

```
# 用递推实现
a, b = 1, 2
for i in range(3, n + 1):
    # 你的代码
    pass
```

```
return b
```

```
# 测试
for i in range(1, 11):
    print(f"{i}阶楼梯: {my_climb_stairs(i)}种走法")
```

2.3 生成前N项列表

```
def generate_fibonacci(n):
    """
    返回斐波那契数列的前n项（列表形式）
    """
    if n <= 0:
        return []
    if n == 1:
        return [1]
```

```
result = [1, 1]
# 用循环生成后续项
for i in range(2, n):
    # 你的代码（用result[-1]和result[-2]）
    pass
```

```
return result
```

```
# 测试
fibs = generate_fibonacci(20)
print(f"前20项: {fibs}")
print(f"长度: {len(fibs)}")
```

思考：为什么用「result[-1]」和「result[-2]」可以取到最后两个元素？

2.4 运行螺旋线代码

打开「03_递推算法.py」，做以下实验：

- 修改「NUM_SQUARES」为 6、8、10，观察螺旋线变化
- 修改「SCALE」为 10、15、20，观察大小变化
- 修改「SQUARE_COLORS」为你喜欢的颜色

三、进阶练习（选做）

3.1 递推与递归速度对比

写一个对比程序：

```
import time
```

```
def fib_recursive(n):  
    """递归版"""  
    if n <= 0: return 0  
    if n == 1 or n == 2: return 1  
    return fib_recursive(n-1) + fib_recursive(n-2)
```

```
def fib_iterative(n):  
    """递推（迭代）版"""  
    if n <= 0: return 0  
    if n == 1 or n == 2: return 1  
    a, b = 1, 1  
    for _ in range(3, n + 1):  
        a, b = b, a + b  
    return b
```

```
# 测试不同的n  
for n in [10, 20, 30, 35, 40]:  
    # 递推法  
    start = time.time()  
    fib_iterative(n)  
    iter_time = time.time() - start
```

```
# 递归法（n太大时可能会很慢）  
start = time.time()  
fib_recursive(n)  
rec_time = time.time() - start
```

```
print(f"n={n}: 递推={iter_time:.6f}s 递归={rec_time:.6f}s 倍数={rec_time/iter_time:.0f}x")
```

问题：为什么n越大，递归比递推慢得越多？

3.2 杨辉三角

杨辉三角也是一个经典的递推问题！

```
1  
1 1  
1 2 1  
1 3 3 1  
1 4 6 4 1
```

1 5 10 10 5 1

规律：每个数 = 上一行对应位置的数 + 上一行前一个位置的数

```
def yanghui_triangle(n):
    """打印杨辉三角的前n行"""
    triangle = []
    for i in range(n):
        row = [1] * (i + 1) # 每行首位都是1
        for j in range(1, i):
            # 递推公式
            row[j] = triangle[i-1][j-1] + triangle[i-1][j]
        triangle.append(row)
    return triangle
```

```
# 打印前10行
for row in yanghui_triangle(10):
    print(row)
```

3.3 铺砖问题

用 1×2 的砖铺满 $2 \times n$ 的地面，有多少种铺法？

提示：

- $n=1$ ：1种（竖着放一块） $f(1)=1$
- $n=2$ ：2种（竖竖 或 横横） $f(2)=2$
- $n=3$ ： $f(3)=f(2)+f(1)=3$ 种
- 递推公式： $f(n)=f(n-1)+f(n-2)$

问题：这是哪个我们学过的数列？ ____

四、挑战题（超级选做）

4.1 斐波那契查找

在斐波那契数列中，如果我们要找到第一个大于1000的斐波那契数，应该怎么做？

```
def find_first_greater_than(limit):
    """找到第一个大于limit的斐波那契数"""
    a, b = 1, 1
    n = 2
    while b <= limit:
        # 你的代码
```

```
pass
return n, b
```

```
# 测试
n, value = find_first_greater_than(1000)
print(f"第{n}项: {value}") # 应该输出第17项: 1597
```

4.2 环保植树问题

小明种树，第1天种1棵，第2天种1棵，从第3天开始，每天种的树是前两天的和。

请问第30天一共种了多少棵树？

提示：这是求和问题！不是求第30项，而是求前30项的和。

```
def total_trees(days):
    """计算days天一共种了多少棵树"""
    # 首先生成斐波那契数列的前days项
    # 然后求和
    pass
```

孩子完成了以上练习（请家长在完成的项目前打√）：

>

☐ 基础练习（1.1~1.4）

☐ 编程练习（2.1~2.4）

☐ 进阶练习（3.1~3.3）

☐ 挑战题（4.1~4.2）

>

>

孩子的话：今天学了递推算法，我知道了_____

>

自我评价：☐ 全都会了 ☐ 大部分会了 ☐ 还要再练练