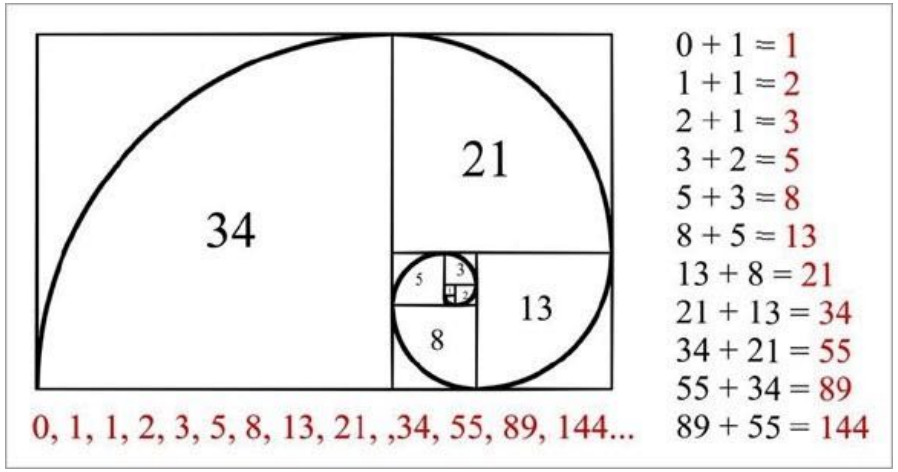
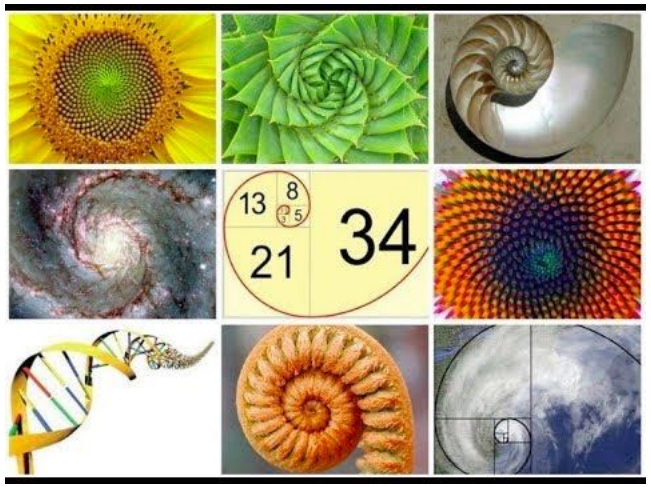
典型例子：斐波那契数列

- 斐波那契数列的发现者，是意大利数学家列昂纳多·斐波那契（Leonardo Fibonacci）
- 从第3项开始，每一项都等于前两项之和
- 相邻两项之比，无限趋近于“黄金分割数”0.618….
- 在自然界中有很多实例
- 黄金分割比例广泛应用于美术





递归函数：fibonacci

- 观察fibonacci数列的定义
- 一个典型的递归定义
- 可以写出递归函数fibonacci

0，1，1，2，3，5，8，13，21，34，55，……

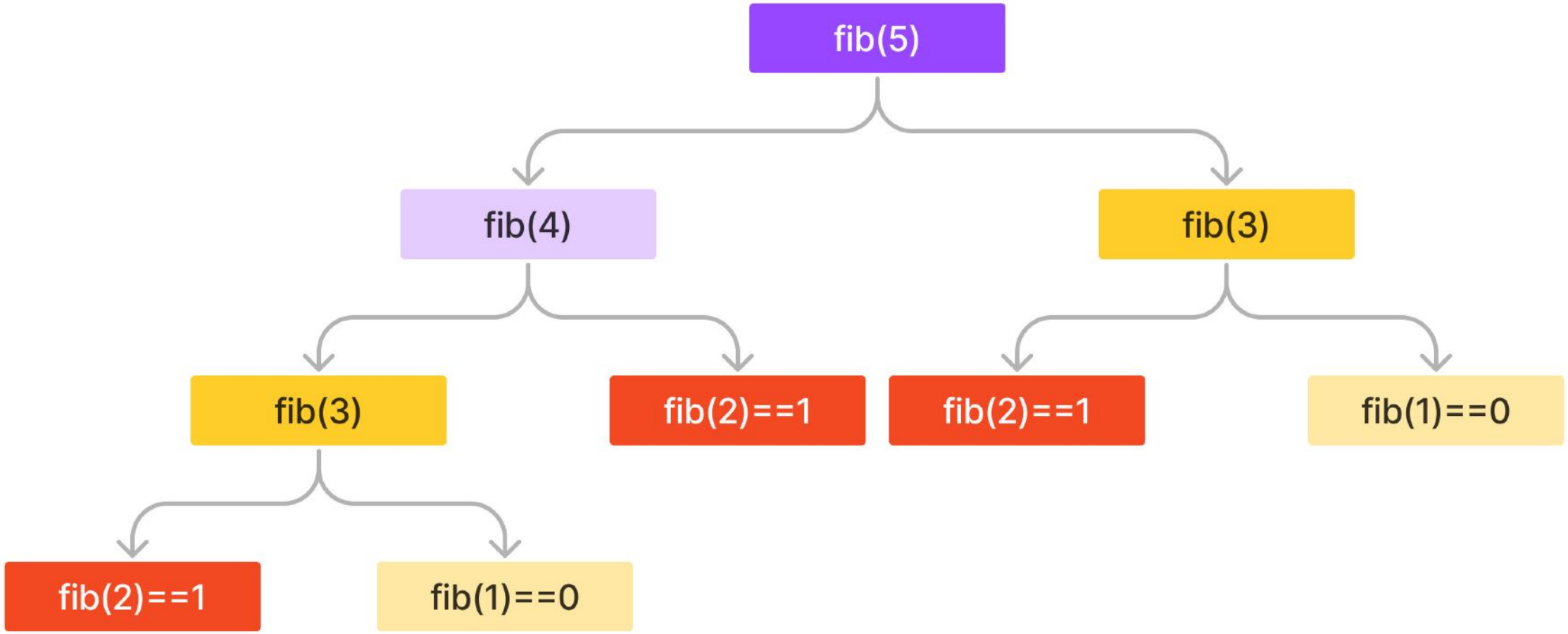
$$F_0 = 0$$

$$F_1 = 1$$

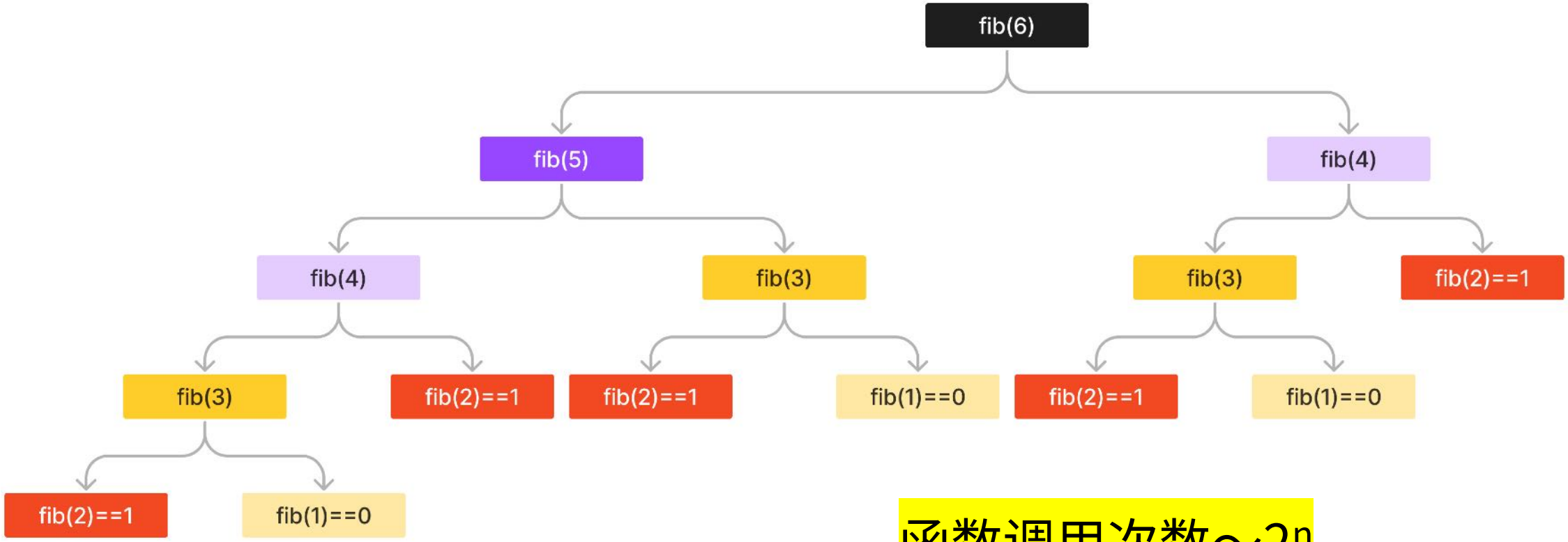
$$F_n = F_{n-1} + F_{n-2}, n > 1$$

```
1  def fibonacci(n):
2      if n == 1:
3          return 0
4      elif n == 2:
5          return 1
6      else:
7          return fibonacci(n - 1) + fibonacci(n - 2)
```

fibonacci(5)的调用序列



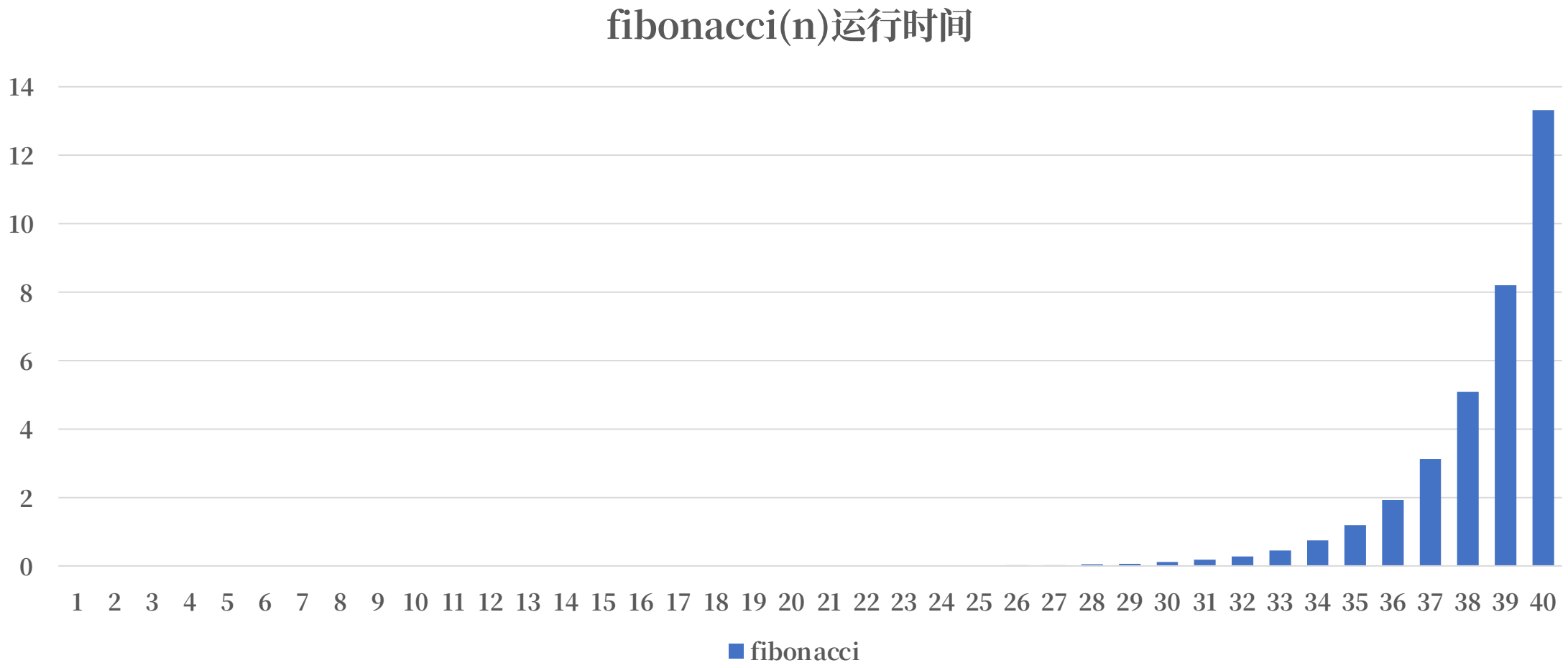
fibonacci(6)的调用序列



函数调用次数～2^n

所以，你的机器上计算fibonacci(40)用了多长时间?

重复计算：时间指数增长


fibonacci(n)运行时间
14
12
10
8
6
4
2
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
fibonacci

函数值缓存：用空间换取时间

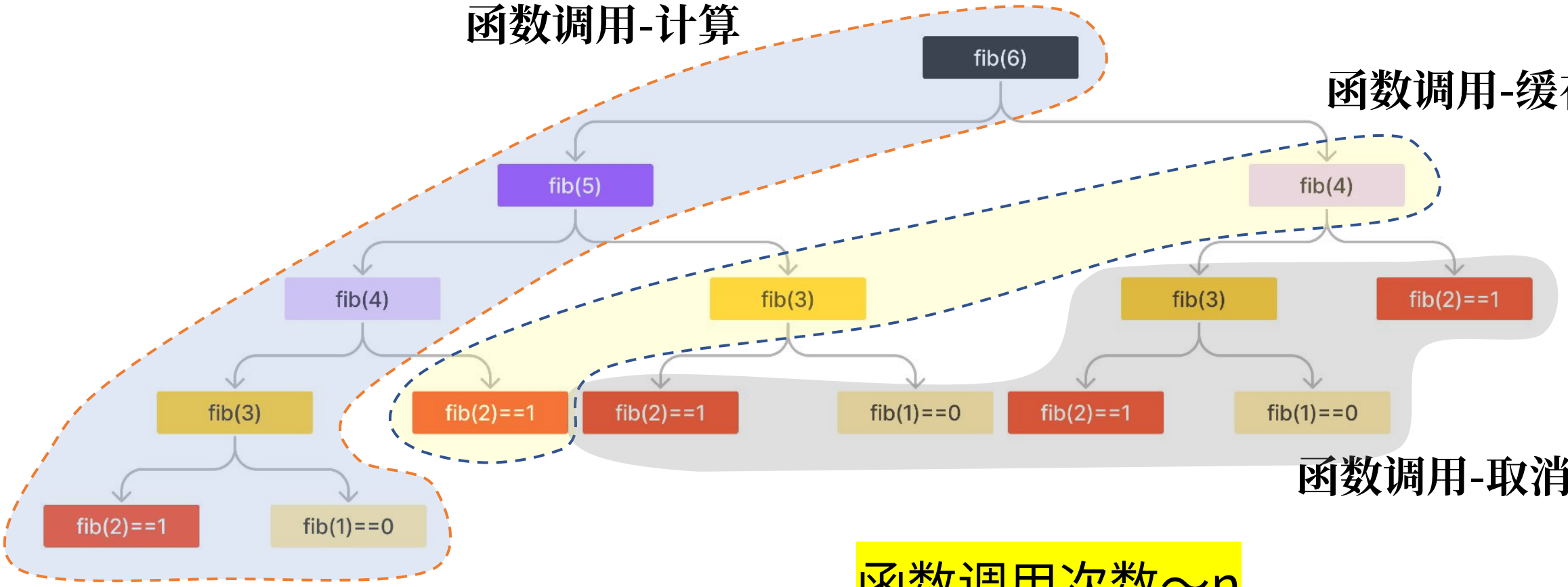
```python
cache = [-1] * 120


def fibonacci(n):
    if cache[n] == -1:  # 是否命中缓存
        if n == 1:
            f = 0
        elif n == 2:
            f = 1
        else:
            f = fibonacci(n - 1) + fibonacci(n - 2)
        cache[n] = f
    return cache[n]


print(fibonacci(100))
```

缓存（记录本）

查找缓存

记录到缓存

函数值缓存版调用序列



函数调用次数～n

函数值缓存：lru_cache装饰器

- 来自functools模块的lru_cache装饰器
- 对产生大量重复计算的递归函数自动提供函数值缓存

```
1  from functools import lru_cache
2
3  @lru_cache(maxsize=128)
4  def fibonacci(n):
5      if n == 1:
6          return 0
7      elif n == 2:
8          return 1
9      else:
10         return fibonacci(n - 1) + fibonacci(n - 2)
11
12 print(fibonacci(100))
```

- 函数参数必须是不可变类型（Hashable）
- 函数满足“单值性”