

递归、分治

(Recursion, Divide & Conquer)



扫码了解极客时间《算法面试通关40讲》视频课程

本节内容

1. Recursion

2. Divide & Conquer

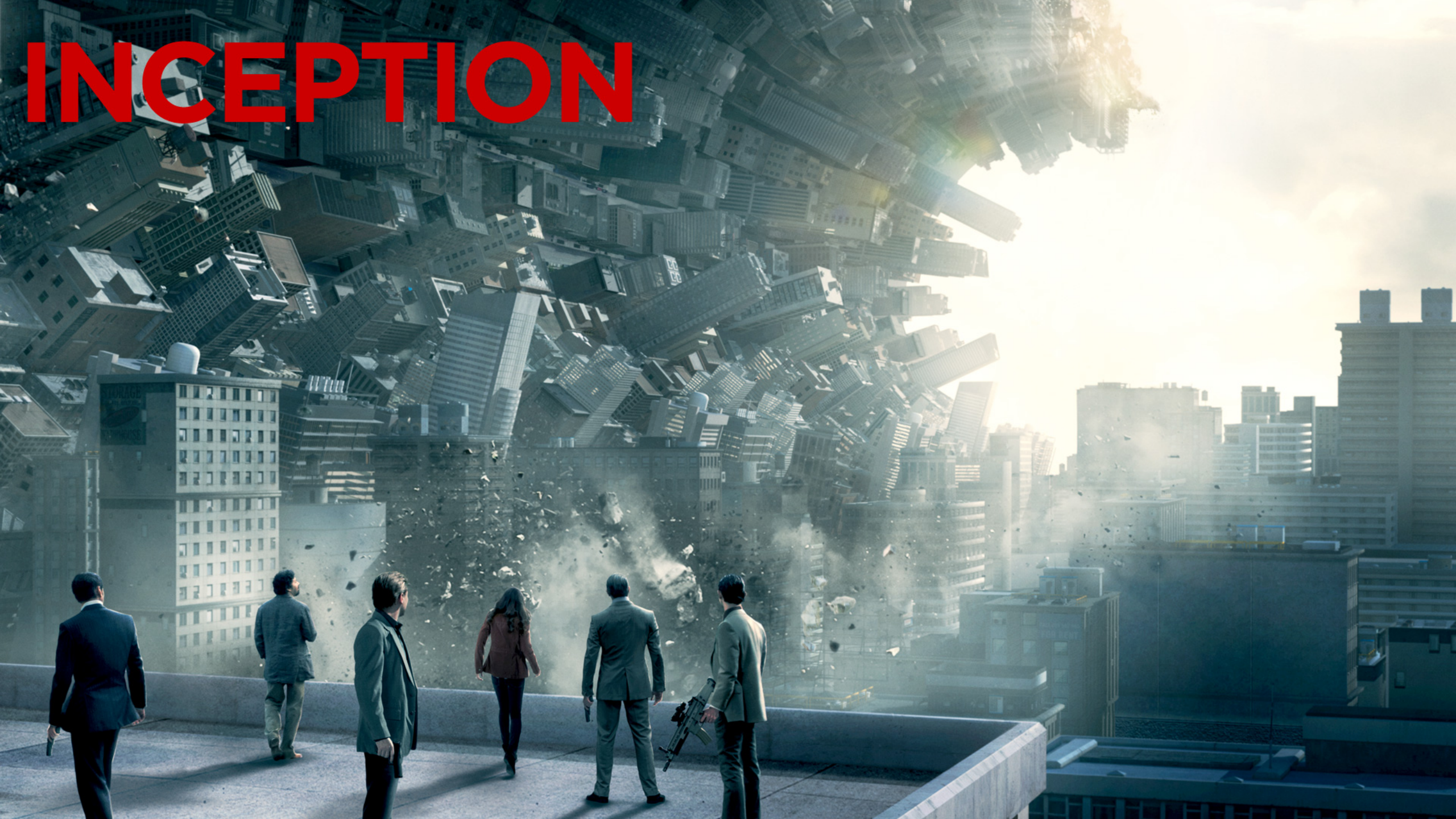
Recursion

递归 — 循环

通过函数体来进行的循环

递归

1. 从前有个山，
2. 山里有个庙，
3. 庙里有个和尚讲故事：



INCEPTION

Recursion

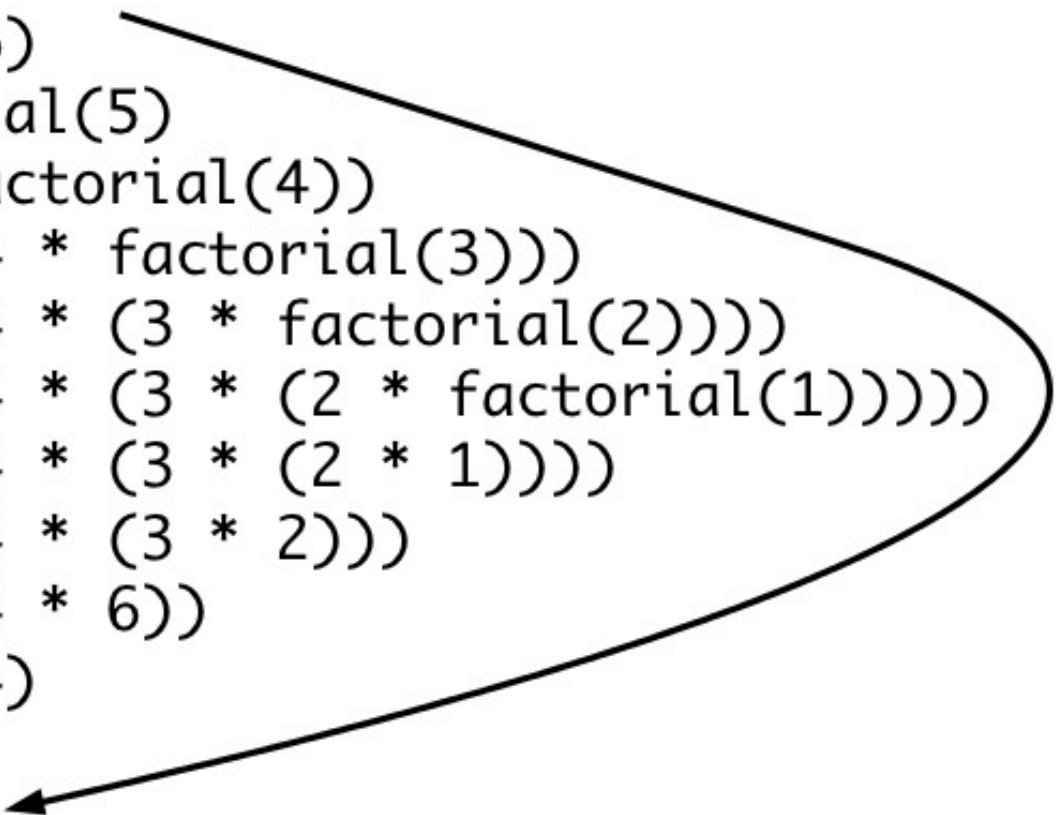
计算 $n!$

$$n! = 1 * 2 * 3 * \dots * n$$

```
def Factorial(n):  
    if n <= 1:  
        return 1  
    return n * Factorial(n - 1)
```


Recursive

```
factorial(6)
6 * factorial(5)
6 * (5 * factorial(4))
6 * (5 * (4 * factorial(3)))
6 * (5 * (4 * (3 * factorial(2))))
6 * (5 * (4 * (3 * (2 * factorial(1)))))
6 * (5 * (4 * (3 * (2 * 1))))
6 * (5 * (4 * (3 * 2)))
6 * (5 * (4 * 6))
6 * (5 * 24)
6 * 120
720
```



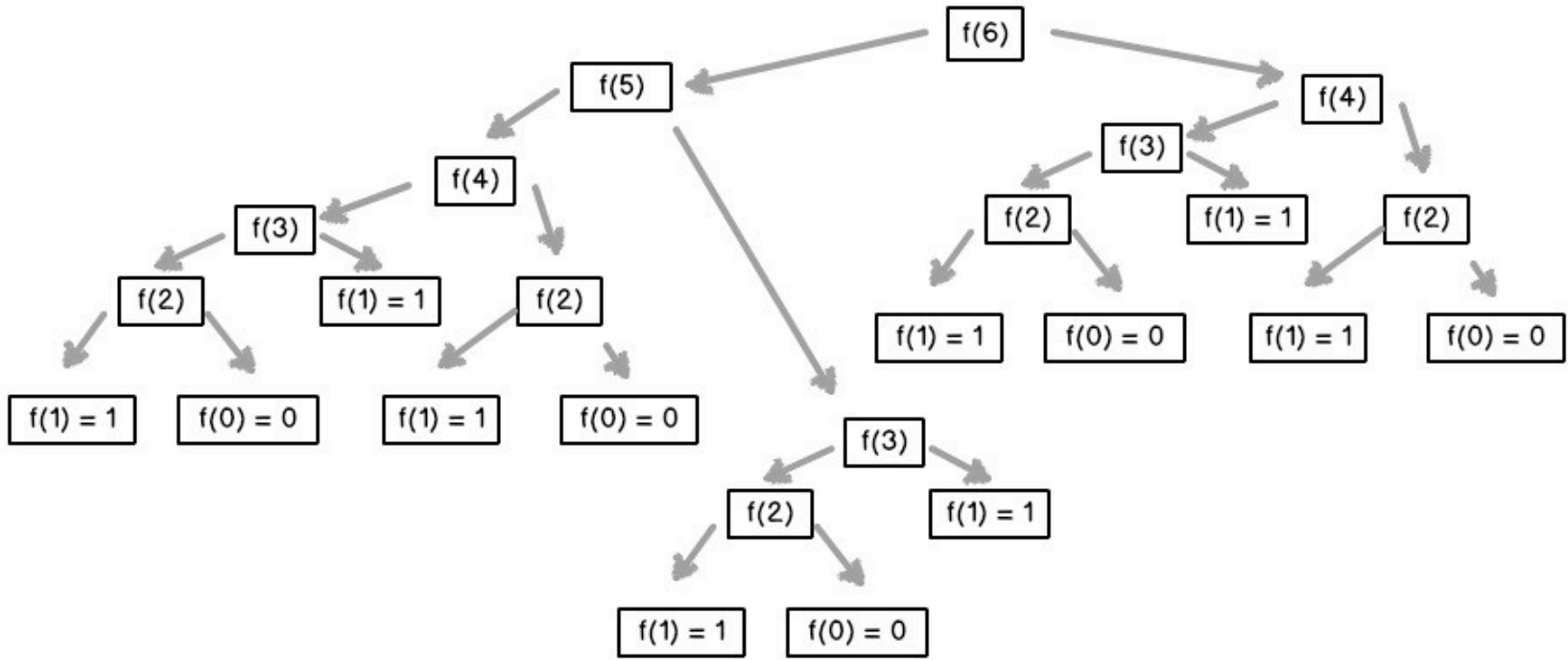
Recursion

Fibonacci array: 1, 1, 2, 3, 5, 13, 21, 34, ...

$$F(n) = F(n-1) + F(n-2)$$

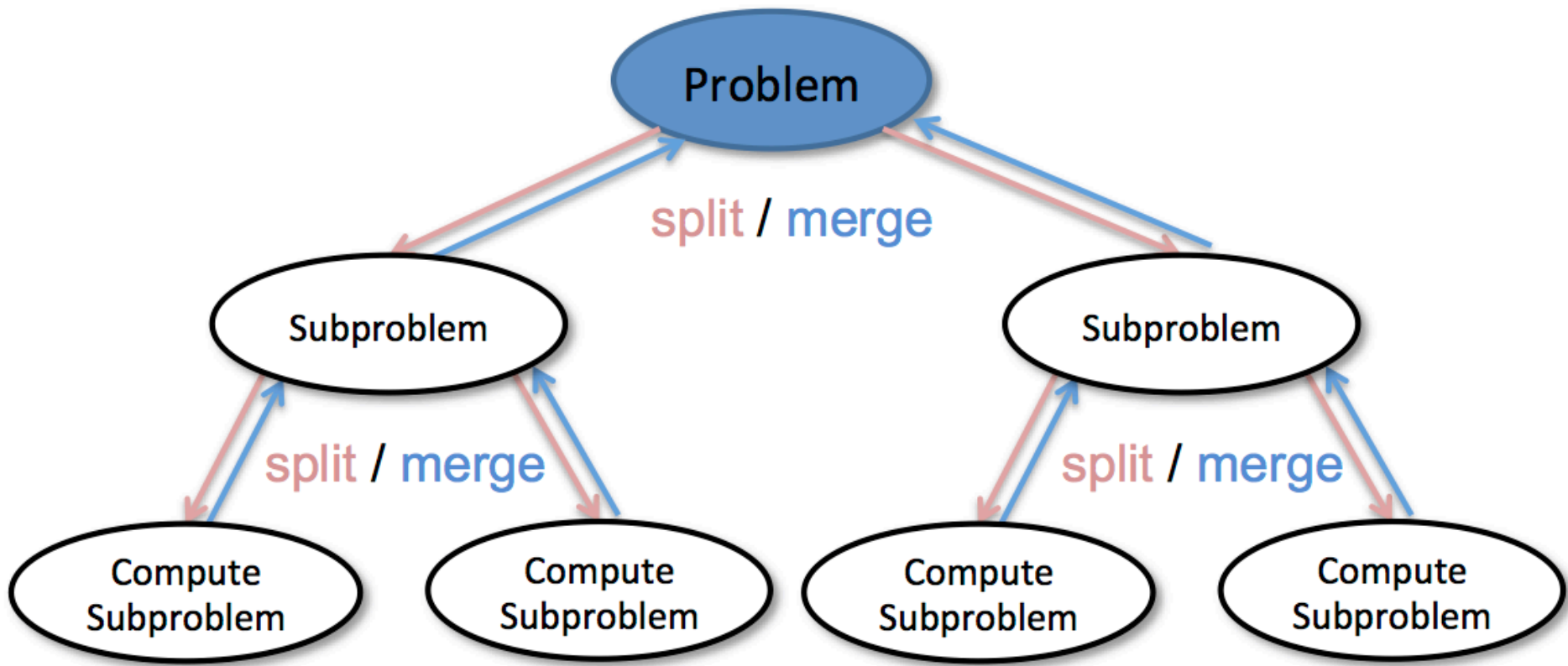
```
def fib(n):  
    if n == 0 or n == 1:  
        return n  
    return fib(n - 1) + fib(n - 2)
```

Fib(6)

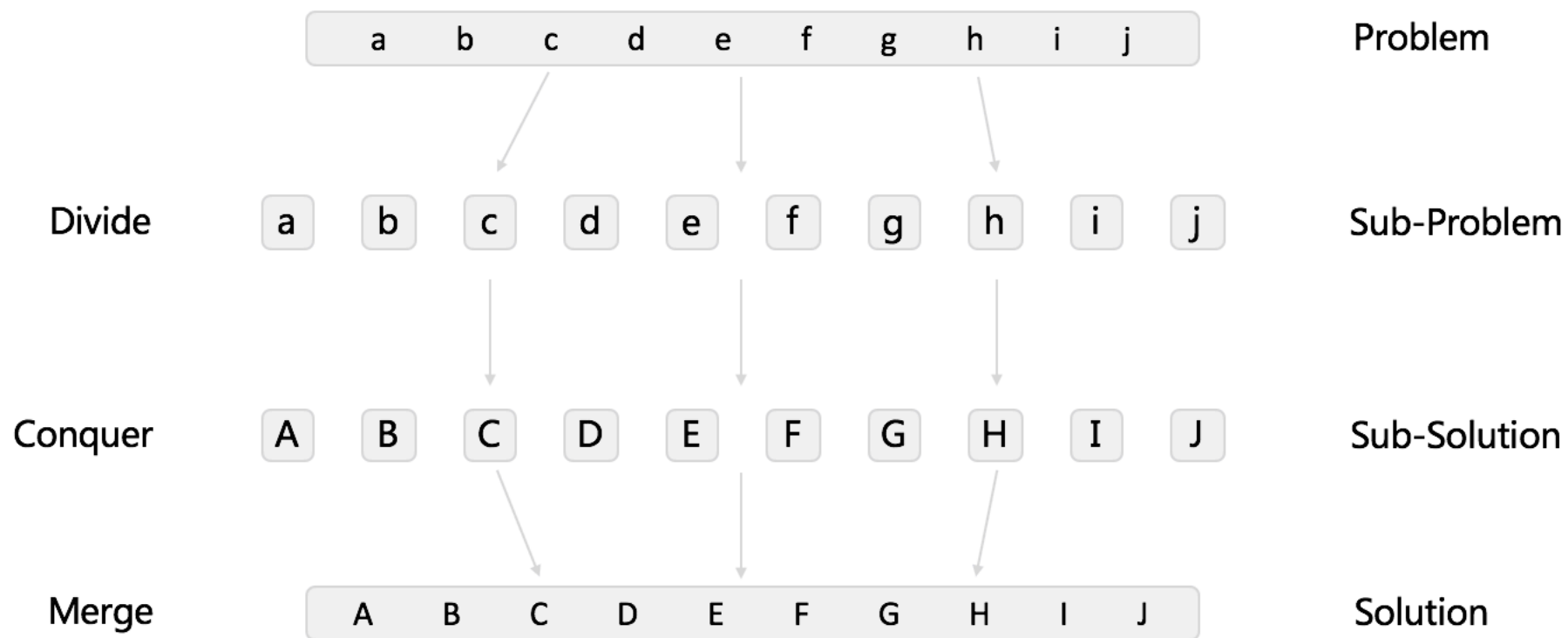


```
def recursion(level, param1, param2, ...):  
  
    # recursion terminator  
    if level > MAX_LEVEL:  
        print_result  
        return  
  
    # process logic in current level  
    process_data(level, data...)   
  
    # drill down  
    self.recursion(level + 1, p1, ...)   
  
    # reverse the current level status if needed  
    reverse_state(level)
```

分治 - Divide & Conquer



Divide & Conquer




```
def divide_conquer(problem, param1, param2, ...):  
  
    # recursion terminator  
    if problem is None:  
        print_result  
        return  
  
    # prepare data  
    data = prepare_data(problem)  
    subproblems = split_problem(problem, data)  
  
    # conquer subproblems  
    subresult1 = self.divide_conquer(subproblems[0], p1, ...)  
    subresult2 = self.divide_conquer(subproblems[1], p1, ...)  
    subresult3 = self.divide_conquer(subproblems[2], p1, ...)  
    ...  
  
    # process and generate the final result  
    result = process_result(subresult1, subresult2, subresult3, ...)
```

实战题目

1. <https://leetcode.com/problems/powx-n/description/>
2. <https://leetcode.com/problems/majority-element/description/>
3. <https://leetcode.com/problems/maximum-subarray/description/>
4. <https://leetcode.com/problems/valid-anagram/#/description>
5. <https://leetcode.com/problems/find-all-anagrams-in-a-string/#/description>
6. <https://leetcode.com/problems/anagrams/#/description>



扫码了解极客时间《算法面试通关40讲》视频课程