

# 极客大学算法训练营

## 第八课

### 分治、回溯

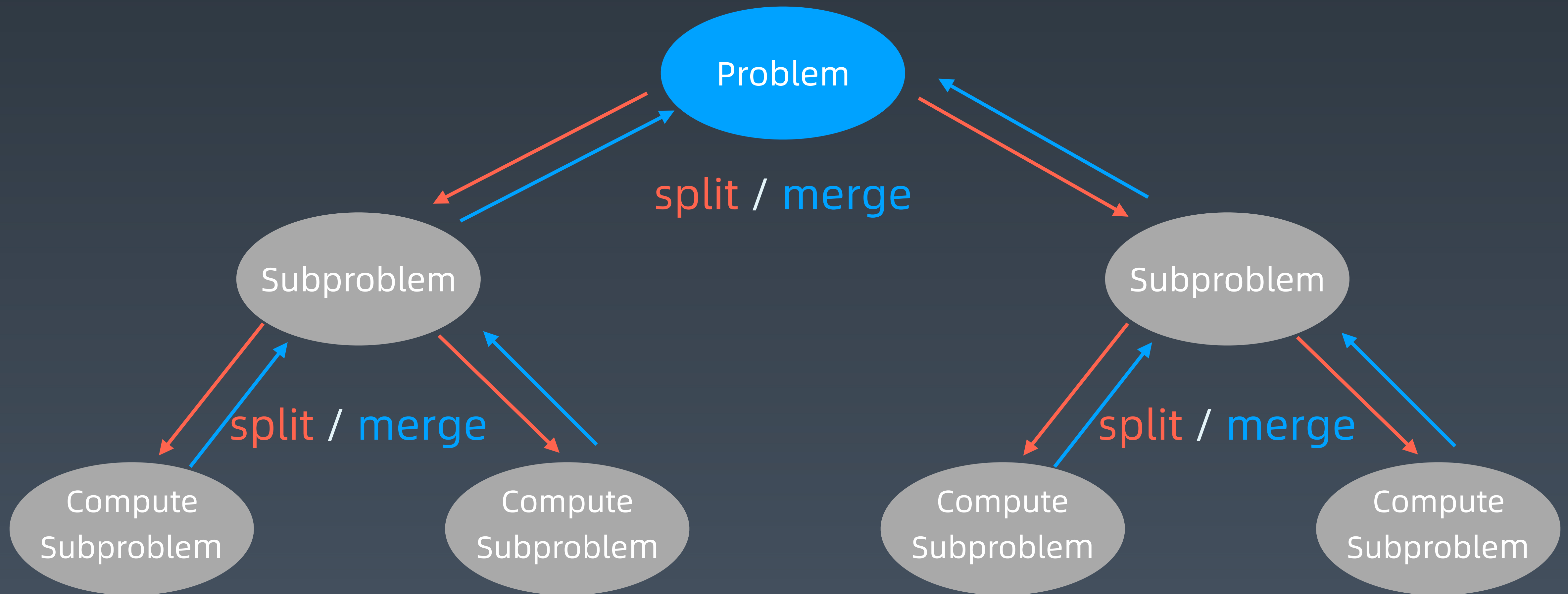
覃超

Sophon Tech 创始人，前 Facebook 工程师

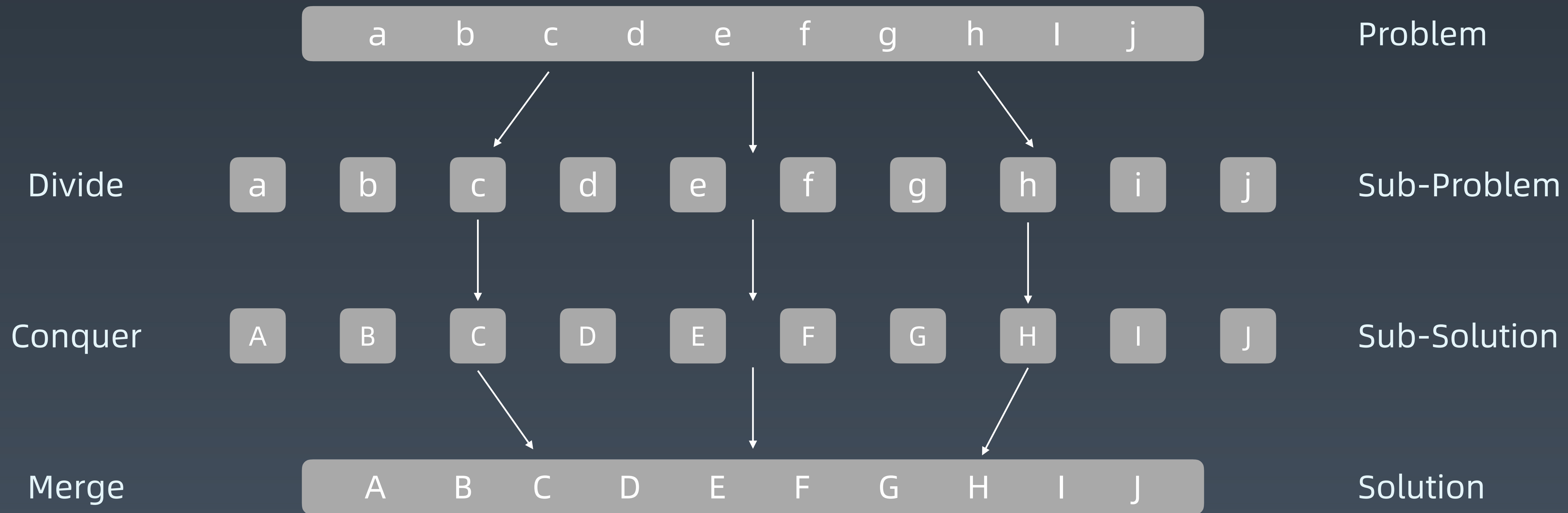
分治

Divide & Conquer

# 递归状态树



# 分治 Divide & Conquer



# 分治代码模板

```
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, ...)  
  
    # revert the current level states
```

回溯

Backtracking

# 回溯

回溯法采用试错的思想，它尝试分步的去解决一个问题。在分步解决问题的过程中，当它通过尝试发现现有的分步答案不能得到有效的正确的解答的时候，它将取消上一步甚至是上几步的计算，再通过其它的可能的分步解答再次尝试寻找问题的答案。

回溯法通常用最简单的递归方法来实现，在反复重复上述的步骤后可能出现两种情况：

- 找到一个可能存在的正确的答案；
- 在尝试了所有可能的分步方法后宣告该问题没有答案。

在最坏的情况下，回溯法会导致一次复杂度为指数时间的计算。

# 预习题目

1. <https://leetcode-cn.com/problems/powx-n/>
2. <https://leetcode-cn.com/problems/subsets/>



# 实战题目

1. <https://leetcode-cn.com/problems/majority-element/description/> （简单、但是高频）
2. <https://leetcode-cn.com/problems/letter-combinations-of-a-phone-number/>
3. <https://leetcode-cn.com/problems/n-queens/>

THANKS! |  极客大学