题目介绍[1]

有 k 种颜色的涂料和一个包含 n 个栅栏柱的栅栏，请你按下述规则为栅栏设计涂色方案：

每个栅栏柱可以用其中 一种 颜色进行上色。 相邻的栅栏柱 最多连续两个 颜色相同。 给你两个整数 k 和 n ，返回所有有效的涂色方案数 。

示例 1

> 输入：n = 3, k = 2
>
> 输出：6
>
> 解释：所有的可能涂色方案如上图所示。注意，全涂红或者全涂绿的方案属于无效方案，因为相邻的栅栏柱 最多连续两个 颜色相同。

示例 2

> 输入：n = 1, k = 1
>
> 输出：1

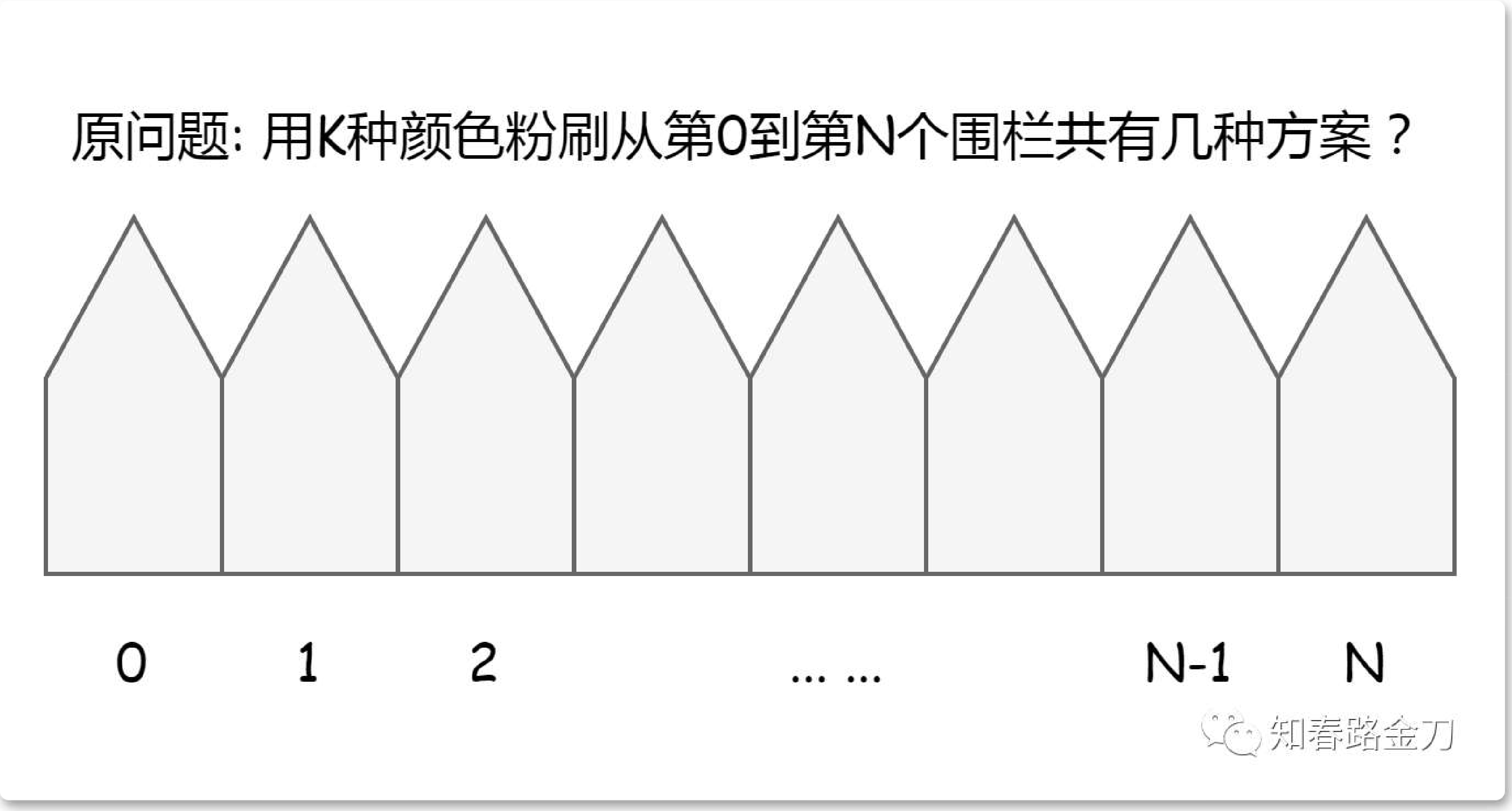
示例 3

> 输入：n = 7, k = 2
>
> 输出：42

微信搜一搜
知春路金刀

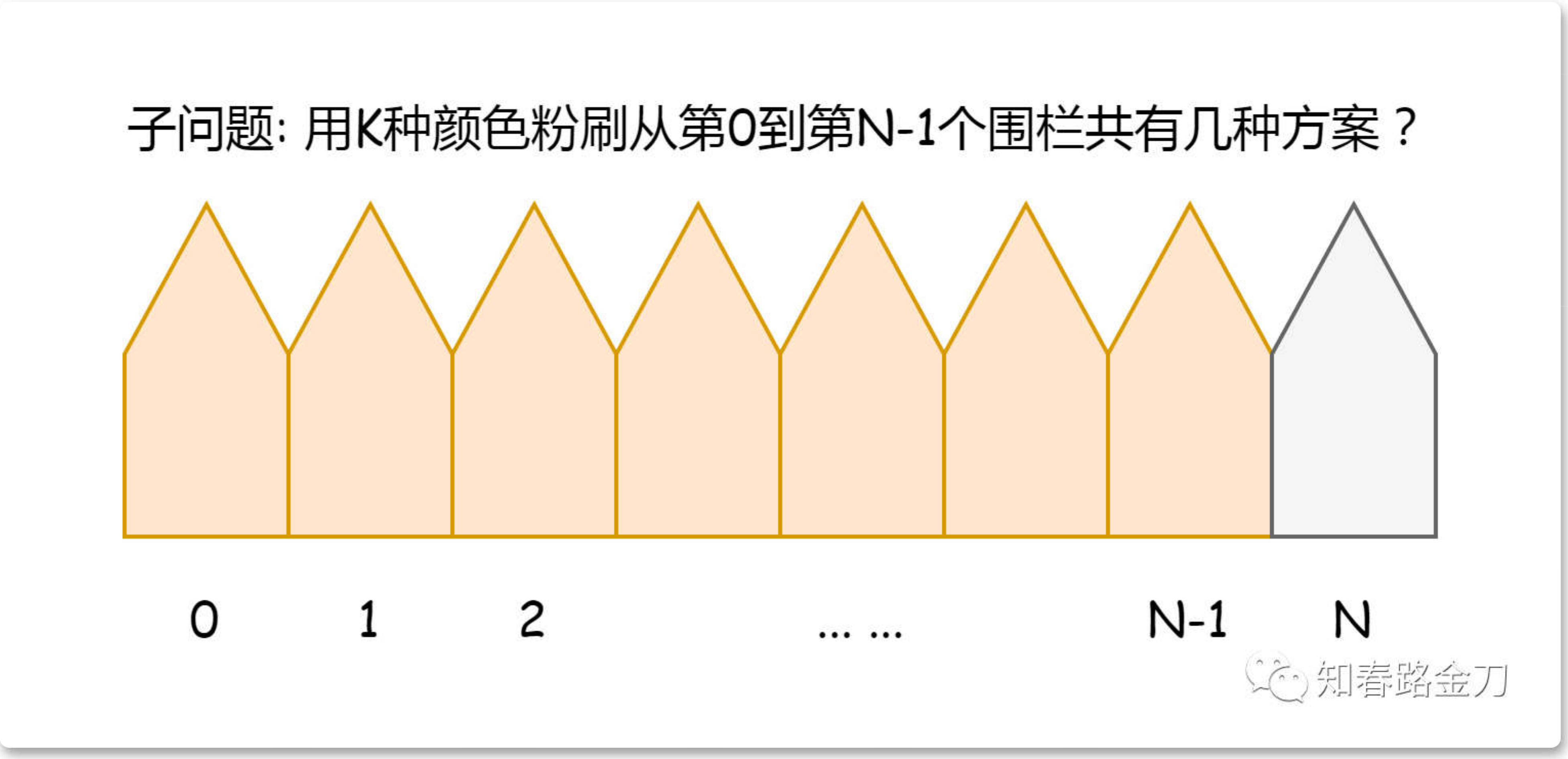
题目解答

首先寻找子问题



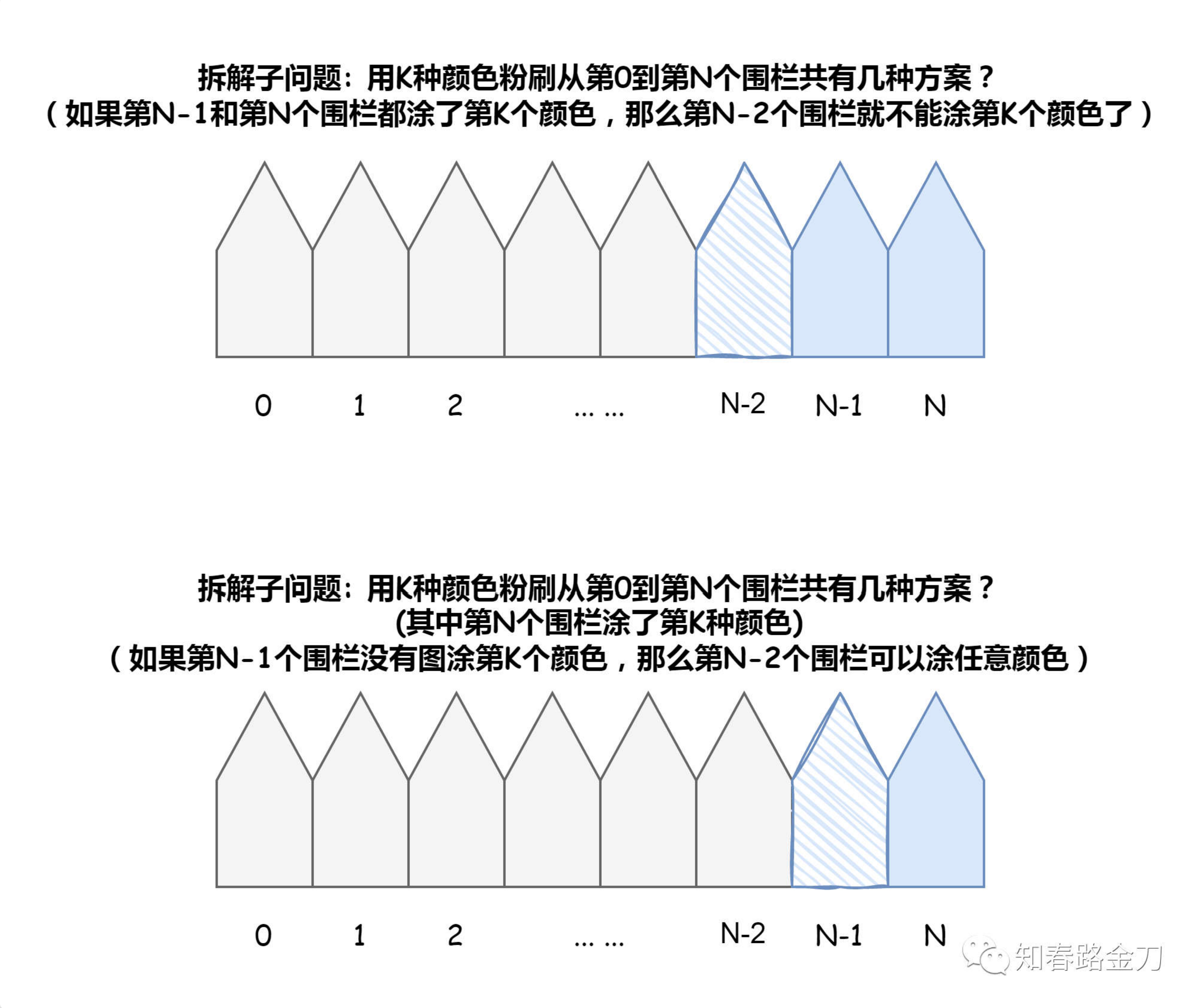
题目的原问题是求解用**「K 种颜色粉刷从第 0 到第 N 个围栏共有几种方案」**，这个问题可以拆成如下 N 个子问题

- 用 K 种颜色粉刷第 0 个围栏共有几种方案
- 用 K 种颜色粉刷从第 0 到第 1 个围栏共有几种方案
- … …
- 用 K 种颜色粉刷从第 0 到第 N-1 个围栏共有几种方案



注意这题有一个限制就是相邻的栅栏**「最多连续两个颜色相同」**，所以小粉刷匠每刷一个围栏的时候，他需要思考这个房子要刷哪种颜色，刷第 1 种颜色？刷第 2 种颜色？刷第 K 种颜色？这样每一个子问题又可以继续拆解变成如下 K*N 个子问题

微信搜一搜
知春路金刀



- 用 K 种颜色粉刷从第 0 到第 0 个围栏共有几种方案
  - 给第 0 个围栏刷第 1 种颜色时，粉刷从第 0 到第 0 个围栏共有几种方案？
  - 给第 0 个围栏刷第 2 种颜色时，粉刷从第 0 到第 0 个围栏共有几种方案？
  - … …
  - 给第 0 个围栏刷第 K 种颜色时，粉刷从第 0 到第 0 个围栏共有几种方案？
- … …
- 用 K 种颜色粉刷从第 0 到第 N-1 个围栏共有几种方案
  - 给第 N-1 个围栏刷第 1 种颜色时，粉刷从第 0 到第 N-1 个围栏共有几种方案？
  - 给第 N-1 个围栏刷第 2 种颜色时，粉刷从第 0 到第 N-1 个围栏共有几种方案？
  - … …
  - 给第 N-1 个围栏刷第 K 种颜色时，粉刷从第 0 到第 N-1 个围栏共有几种方案？

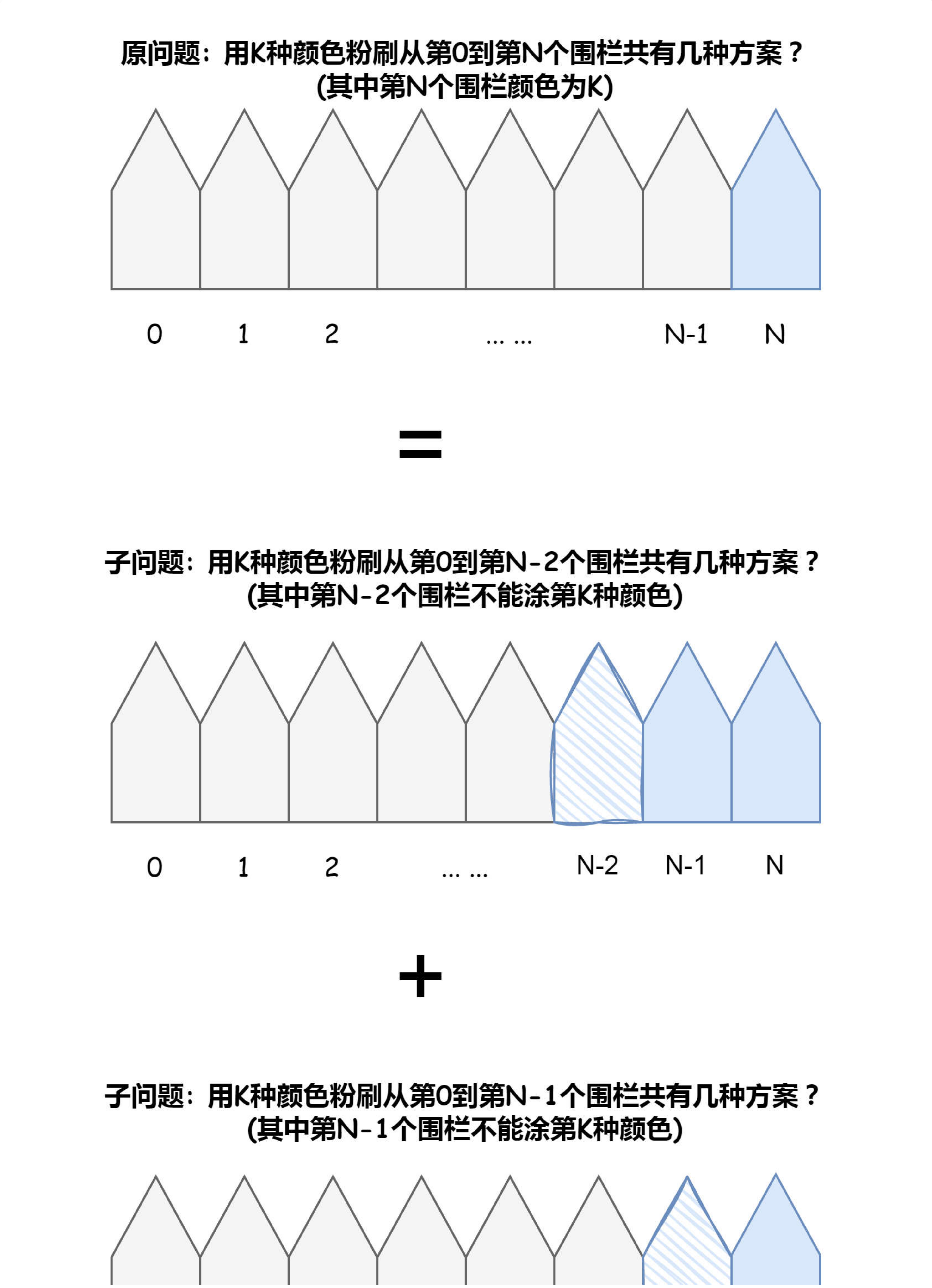
微信搜一搜
知春路金刀

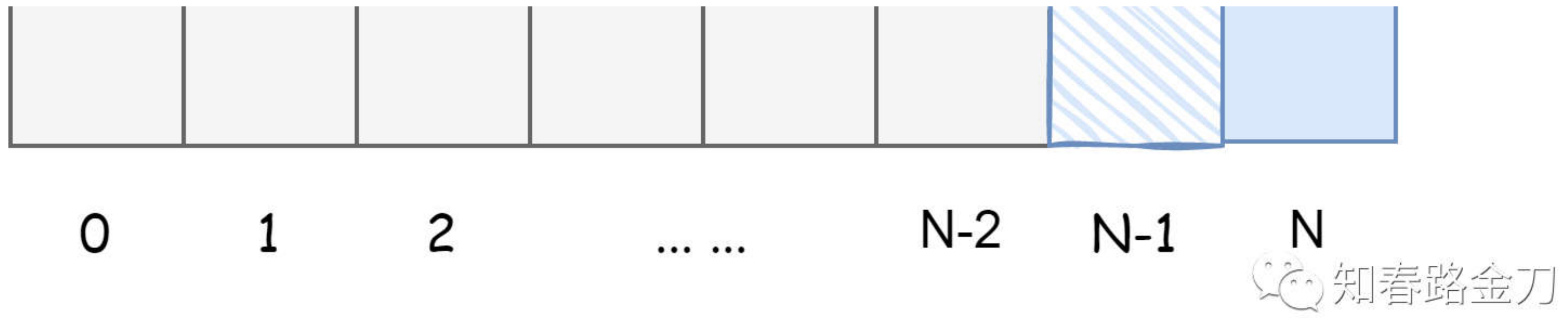
确定状态转移方程

子问题已经确定出来了，那么如果我们知道了如何求解子问题**「用 K 种颜色粉刷从第 0 到第 N-1 个围栏共有几种方案」**，那么我们如何根据这个子问题来算出原问题**「用 K 种颜色粉刷从第 0 到第 N 个围栏共有几种方案」**呢？

粉刷匠为了计算出方案个数，自学了编程然后搞了 K 个数组 color1、color2 … colorK，其中 color1[n]表示**「用 K 种颜色粉刷从第 0 到第 N 个围栏共有几种方案，且第 N 个围栏粉刷为第 1 种颜色」**。 color2[n] … colorK[n]亦然，粉刷匠每到达一个围栏的时候，都会去更新 color1[n]、color2 [n] … colorK[n]，当粉刷匠来到了第 N 个围栏时心里可能这么想：

- 我要把第 N 个围栏刷为第 K 种颜色
  - 粉刷匠决定把第 N 个围栏刷为第 K 种颜色，并记录下当前**「第 N 个围栏刷为第 K 种颜色共有几种方案」**colorK[n] = (color1[n-1]+ ... + colorK-1[n-1]) + (color1[n-2] + ... + colorK-1[n-2])。
  - **「解释」**：第 N 个围栏刷为第 K 种颜色时，**「用 K 种颜色粉刷从第 0 到第 N 个围栏共有几种方案数量」**等于**「第 N-2 个围栏不能是第 K 种颜色的数量」**加上**「第 N-1 个围栏不能是第 K 种颜色的数量」**，如下图所示。





我们又注意到 无论第 N 个围栏是哪种颜色，**「用 K 种颜色粉刷从第 0 到第 N 个围栏的方法数量，且第 N 个围栏粉刷为第 K 种颜色」**其实都是相等的，跟第 N 个围栏是哪种颜色无关，也就是说

> color1[n] == color2[n] == … == colorK[n]

所以上述**「用 K 种颜色粉刷从第 0 到第 N 个围栏共有几种方案，其第 N 个围栏粉刷为第 K 种颜色」**

> colorK[n] = (color1[n-1]+ … + colorK-1[n-1]) + (color1[n-2] + … + colorK-1[n-2])

可以化简为

> color[n] = color[n-1](K-1) + color[n-2](K-1)

其中 color[n]表示**「用 K 种颜色粉刷从第 0 到第 N 个围栏共有几种方案」**。

因此状态转移方程如下

> dp[n] = dp[n-1](K-1) + dp[n-2](K-1)

其中 dp 有两个初始状态

> dp[0] = k，dp[1] = k*k

方法一：一维动态规划

代码实现

```
class Solution {
    public int numWays(int n, int k) {
        if (n == 1) {
            return k;
        }
        int[] dp = new int[n];
        dp[0] = k;
        dp[1] = k*k;
        for (int i = 2; i < dp.length; i++) {
            dp[i] = (k-1) * (dp[i-1] + dp[i-2]);
        }
        return dp[n-1];
    }
}
```

复杂度分析

- 时间复杂度:O(N)
- 空间复杂度:O(N)

方法二：动态规划优化版

在方法一中，dp[i]的状态总是依赖于 dp[i-1]与 dp[i-2]，可以用两个变量来 dp1 和 dp2 来代替 dp 数组。

代码实现

```
class Solution {
    public int numWays(int n, int k) {
        if (n == 1) {
            return k;
        }
        int dp2 = k;
        int dp1 = k*k;
        for (int i = 2; i < n; i++) {
            int dp1t = dp1;
            dp1 = (k-1) * (dp1 + dp2);
            dp2 = dp1t;
        }
        return dp1;
    }
}
```

复杂度分析

- 时间复杂度:O(N)
- 空间复杂度:O(1)

其他

> **「图解大厂面试高频算法题」**专题文章主旨是：根据二八法则的原理，以付出 20%的时间成本，获得 80%的刷题的收益，让那些想进互联网大厂或心仪公司的人少走些弯路。
>
> 本专题还在持续更新 ing~ 所有文章、图解和代码全部是金刀亲手完成。内容全部放在了**github**[2]和**gitee**[3]方便小伙伴们阅读和调试，另外还有更多小惊喜等你发现~
>
> 如果你喜欢本篇文章，PLZ 一键三连（关注、点赞、在看）。

参考资料

[1] **原题链接:** https://leetcode-cn.com/problems/paint-fence/

[2] **github:** https://github.com/glodknife

[3] **gitee:** https://gitee.com/goldknife6

微信搜一搜

知春路金刀