

LeetCode 2654. 使数组所有元素变成 1 的最少操作次数

Feijoa_Li

一、 题目描述

给你一个下标从 0 开始的正整数数组 `nums`。你可以对数组执行以下操作任意次：

选择一个满足 $0 \leq i < n - 1$ 的下标 i ，将 `nums[i]` 或者 `nums[i+1]` 两者之一替换成它们的最大公约数。

请你返回使数组 `nums` 中所有元素都等于 1 的最少操作次数。如果无法让数组全部变成 1，请你返回 -1。

两个正整数的最大公约数指的是能整除这两个数的最大正整数。

二、 解题思路

本题的关键在于如何生成第一个1，因为一旦有了一个1，就可以通过它把相邻的元素逐个变成1。

核心观察：

- 如果数组中已经存在1，那么只需要将其他非1元素逐个变成1，操作次数为 $n - \text{count}(1)$
- 如果数组中不存在1，我们需要通过操作生成第一个1
- 生成1的最短方式：找到最短的连续子数组，使得这些数的gcd为1

算法步骤：

1. 检查数组中是否已有1，如果有则直接返回 $n - \text{count}(1)$
2. 寻找最短的连续子数组使得gcd为1，设最短长度为 minLen
3. 如果找不到这样的子数组，返回-1
4. 否则，总操作次数为： $n + \text{minLen} - 2$

数学解释：

- 生成第一个1需要 $\text{minLen} - 1$ 次操作（在最短gcd为1的子数组上操作）
- 用这个1感染其他 $n - 1$ 个元素需要 $n - 1$ 次操作
- 总操作次数： $(\text{minLen} - 1) + (n - 1) = n + \text{minLen} - 2$

三、代码实现

```

class Solution {
public:
    const int inf = 1e9;
    int minOperations(vector<int>& nums) {
        int cnt = inf;
        // 计算从位置x开始，需要多少次操作能得到gcd=1
        auto cul = [&](int x) {
            int GCD = nums[x];
            int cnt = 0;
            for (int i = x + 1; i < nums.size() && GCD != 1; i++) {
                GCD = gcd(GCD, nums[i]);
                cnt++;
            }
            return (GCD == 1 ? cnt : inf);
        };
        // 寻找生成1的最短距离
        for (int i = 0; i < nums.size(); i++) cnt = min(cnt, cul(i));
        if (cnt == inf) return -1;

        // 总操作次数 = 生成第一个1的操作 + 用1感染其他元素的操作
        int res = nums.size() + cnt - 1;
        // 如果已有1，直接计算感染操作
        if (find(nums.begin(), nums.end(), 1) != nums.end())
            res = nums.size() - count(nums.begin(), nums.end(), 1);
        return res;
    }
};

```

四、复杂度分析

- 时间复杂度: $O(n^2)$, 其中 n 是数组长度。最坏情况下需要检查所有子数组。
- 空间复杂度: $O(1)$, 只使用了常数级别的额外空间。

五、算法优化

实际上，我们可以通过预处理来优化gcd的计算，但考虑到 $n \leq 50$ ，当前的解法已经足够高效。

关键洞察：

- 问题的核心是找到生成第一个1的最短路径
- 一旦有了1，剩下的操作就是线性的

- 使用gcd的单调性：连续子数组的gcd随着长度增加是非递增的，从而利用线段树优化到 $O(n \log n)$

