238.除自身以外数组的乘积

> 计算学部十大打卡活动——“龙舞编程新春会”编程打卡（2024-1-30）
>
> 力扣——238.除自身以外数组的乘积

一、题目

给你一个整数数组 `nums`，返回 *数组 `answer`，其中 `answer[i]` 等于 `nums` 中除 `nums[i]` 之外其余各元素的乘积*。

题目数据 **保证** 数组 `nums` 之中任意元素的全部前缀元素和后缀的乘积都在 **32 位** 整数范围内。

请 **不要使用除法**，且在 `O(n)` 时间复杂度内完成此题。

示例 1:

```
输入: nums = [1,2,3,4]
输出: [24,12,8,6]
```

示例 2:

```
输入: nums = [-1,1,0,-3,3]
输出: [0,0,9,0,0]
```

提示:

- `2 <= nums.length <= 105`
- `-30 <= nums[i] <= 30`
- **保证** 数组 `nums` 之中任意元素的全部前缀元素和后缀的乘积都在 **32 位** 整数范围内

二、思路

使用左边所有数字的乘积乘以右边所有数字的乘积得到除自己以外所有数字的乘积。

只需要维护两个数组，分别记录左边乘积和右边乘积。

三、代码

```cpp
class Solution {
public:
    vector<int> productExceptSelf(vector<int>& nums) {
        int length = nums.size();

        // L 和 R 分别表示左右两侧的乘积列表
        vector<int> L(length, 0), R(length, 0);

        vector<int> answer(length);

        // L[i] 为索引 i 左侧所有元素的乘积
        // 对于索引为 '0' 的元素，因为左侧没有元素，所以 L[0] = 1
```

```
        L[0] = 1;
        for (int i = 1; i < length; i++) {
            L[i] = nums[i - 1] * L[i - 1];
        }

        // R[i] 为索引 i 右侧所有元素的乘积
        // 对于索引为 'length-1' 的元素，因为右侧没有元素，所以 R[length-1] = 1
        R[length - 1] = 1;
        for (int i = length - 2; i >= 0; i--) {
            R[i] = nums[i + 1] * R[i + 1];
        }

        // 对于索引 i，除 nums[i] 之外其余各元素的乘积就是左侧所有元素的乘积乘以右侧所有元素
的乘积
        for (int i = 0; i < length; i++) {
            answer[i] = L[i] * R[i];
        }

        return answer;
    }
};
```

复杂度分析

时间复杂度：$O(n)$

空间复杂度：$O(n)$