# 题目

给你一个二进制数组 nums ，你需要从中删掉一个元素。

请你在删掉元素的结果数组中，返回最长的且只包含 1 的非空子数组的长度。

如果不存在这样的子数组，请返回 0 。

提示 1：

输入：nums = [1,1,0,1]

输出：3

解释：删掉位置 2 的数后，[1,1,1] 包含 3 个 1 。

示例 2：

输入：nums = [0,1,1,1,0,1,1,0,1]

输出：5

解释：删掉位置 4 的数字后，[0,1,1,1,1,1,0,1] 的最长全 1 子数组为 [1,1,1,1,1] 。

示例 3：

输入：nums = [1,1,1]

输出：2

解释：你必须要删除一个元素。

提示：

1 <= nums.length <= 10^5

nums[i] 要么是 0 要么是 1 。

# 分析

## 方法一：滑动窗口

思路：

这道题可以使用滑动窗口来解决。我们维护一个窗口，使窗口内最多只有一个 0。具体步骤如下：

1、使用两个指针 left 和 right 分别表示窗口的左右边界，初始时都指向数组的第一个元素。

2、移动右指针 right，并更新窗口内的 0 的个数 count。

3、如果窗口内 0 的个数超过了 1 个，则开始移动左指针 left，直到窗口内 0 的个数不超过 1 个。

4、在每次移动左右指针时，更新最长子数组的长度 maxLength。

5、最终 maxLength 就是要求的结果。

代码：

class Solution {

public:

int longestSubarray(vector<int>& nums) {

int n = nums.size();

int left = 0, right = 0; // 定义窗口的左右边界

int count = 0; // 统计窗口内 1 的个数

int maxLength = 0; // 记录最长子数组的长度

while (right < n) {

if (nums[right] == 1) {

count++; // 右边界是 1，count 加 1

}

while (right - left + 1 - count > 1) {

if (nums[left] == 1) {

count--; // 如果窗口内 0 的个数超过 1 个，左边界是 1，count 减 1

}

left++; // 移动左边界

}

maxLength = max(maxLength, right - left); // 更新最长子数组的长度

right++; // 移动右边界

}

return maxLength;

}

};