# 题目

给定一个数组 nums ，将其划分为两个连续子数组 left 和 right， 使得：

* left 中的每个元素都小于或等于 right 中的每个元素。
* left 和 right 都是非空的。
* left 的长度要尽可能小。

在完成这样的分组后返回 left 的 长度 。

用例可以保证存在这样的划分方法。

示例 1：

输入：nums = [5,0,3,8,6]

输出：3

解释：left = [5,0,3]，right = [8,6]

示例 2：

输入：nums = [1,1,1,0,6,12]

输出：4

解释：left = [1,1,1,0]，right = [6,12]

提示：

2 <= nums.length <= 10^5

0 <= nums[i] <= 10^6

可以保证至少有一种方法能够按题目所描述的那样对 nums 进行划分。

# 分析

## 方法一：一次遍历

思路：

算法思路是遍历数组nums，同时维护两个变量leftmax和curmax，以及一个用于记录划分点位置的变量loc。leftmax表示从左侧到当前位置为止的最大值，而curmax表示从数组开始到当前位置为止的最大值。

在遍历过程中，我们不断更新curmax，同时检查当前元素nums[i]是否小于leftmax。如果小于，说明我们找到了一个可能的划分点，因为左侧子数组的最大值leftmax不会影响到右侧子数组的最小值。此时，我们更新leftmax为curmax，并更新loc为当前位置i。

最后，返回loc + 1作为左侧子数组的长度（包括划分点本身）。

代码：

class Solution {

public:

int partitionDisjoint(vector<int>& nums) {

int n = nums.size(); // 获取数组长度

int loc = 0; // 初始化划分点位置为0

int leftmax = nums[0]; // 初始化左侧最大值为数组第一个元素

int curmax = nums[0]; // 初始化当前最大值为数组第一个元素

for (int i = 1; i < n; i++) { // 从第二个元素开始遍历数组

curmax = max(curmax, nums[i]); // 更新当前最大值

// 如果当前元素小于左侧最大值，说明找到了一个划分点

if (nums[i] < leftmax) {

leftmax = curmax; // 更新左侧最大值为当前最大值

loc = i; // 更新划分点位置为当前位置

}

}

// 返回左侧子数组的长度（包括划分点）

return loc + 1;

}

};

需要注意的是，这个算法假设题目描述的条件总是成立，即总能找到一个划分点使得左侧子数组的最大值小于等于右侧子数组的最小值。如果输入不满足这个条件，那么这个算法可能不会返回正确的结果。在实际应用中，可能需要增加一些检查来确保输入的有效性。

时间复杂度：O(n)

空间复杂度：O(1)