# 题目

给你一个整数数组arr，请你删除一个子数组（可以为空），使得arr中剩下的元素是非递减的。

一个子数组指的是原数组中连续的一个子序列。

请你返回满足题目要求的最短子数组的长度。

示例 1：

输入：arr = [1,2,3,10,4,2,3,5]

输出：3

解释：我们需要删除的最短子数组是 [10,4,2] ，长度为 3 。剩余元素形成非递减数组 [1,2,3,3,5] 。

另一个正确的解为删除子数组 [3,10,4] 。

示例 2：

输入：arr = [5,4,3,2,1]

输出：4

解释：由于数组是严格递减的，我们只能保留一个元素。所以我们需要删除长度为 4 的子数组，要么删除 [5,4,3,2]，要么删除 [4,3,2,1]。

示例 3：

输入：arr = [1,2,3]

输出：0

解释：数组已经是非递减的了，我们不需要删除任何元素。

示例 4：

输入：arr = [1]

输出：0

提示：

1 <= arr.length <= 10^5

0 <= arr[i] <= 10^9

# 分析

思路：

这段代码的解题思路如下：

1、首先找到数组中从左到右第一个不满足非递减的位置left，以及从右到左第一个不满足非递减的位置right。

2、如果整个数组都是非递减的，返回0。

3、初始化最短非递减子数组长度为right。

4、接下来的步骤是在left到n-1的范围内，对于每个位置i，找到在[left, right]区间内的最大非递减子数组长度，并更新最短长度。

5、最后返回最短非递减子数组长度。

这个算法通过一次从左到右的遍历和一次从右到左的遍历，以及一次在指定区间内的遍历，实现了在O(n)的时间复杂度内找到最短的非递减子数组长度。

代码：

class Solution {

public:

int findLengthOfShortestSubarray(vector<int>& arr) {

int n = arr.size();

int left = 0, right = n - 1;

while (left + 1 < n && arr[left] <= arr[left + 1]) {

left++;

}

if (left == n - 1) {

return 0; // Whole array is non-decreasing.

}

while (right > left && arr[right - 1] <= arr[right]) {

right--;

}

int result = min(n - left - 1, right);

int i = 0, j = right;

while (i <= left && j < n) {

if (arr[i] <= arr[j]) {

result = min(result, j - i - 1);

i++;

} else {

j++;

}

}

return result;

}

};