# 题目

给你一个下标从0开始的整数数组nums。nums的一个子数组如果满足以下条件，那么它是不间断的：

i，i + 1 ，...，j表示子数组中的下标。对于所有满足 i <= i1, i2 <= j 的下标对，都有0 <= |nums[i1] - nums[i2]| <= 2。

请你返回不间断子数组的总数目。

子数组是一个数组中一段连续非空的元素序列。

示例 1：

输入：nums = [5,4,2,4]

输出：8

解释：

大小为 1 的不间断子数组：[5], [4], [2], [4] 。

大小为 2 的不间断子数组：[5,4], [4,2], [2,4] 。

大小为 3 的不间断子数组：[4,2,4] 。

没有大小为 4 的不间断子数组。

不间断子数组的总数目为 4 + 3 + 1 = 8 。

除了这些以外，没有别的不间断子数组。

示例 2：

输入：nums = [1,2,3]

输出：6

解释：

大小为 1 的不间断子数组：[1], [2], [3] 。

大小为 2 的不间断子数组：[1,2], [2,3] 。

大小为 3 的不间断子数组：[1,2,3] 。

不间断子数组的总数目为 3 + 2 + 1 = 6 。

提示：

1 <= nums.length <= 105

1 <= nums[i] <= 109

# 分析

## 方法一：滑动窗口/双指针+哈希表

分析：

在遍历数组的同时，维护窗口内的数字。

由于绝对差至多为2，所以用平衡树或者哈希表维护都行，反正至多维护3个数，添加删除可以视作是O(1)的。（如果用哈希表，还需记录数字的出现次数。）

如果窗口内的最大值与最小值的差大于2，就不断移动左端点left，减少窗口内的数字。

最后

[left,right],[left+1,right],⋯,[right,right]

这一共right−left+1个子数组都是合法的，加入答案。

**代码：**

class Solution {

public:

long long continuousSubarrays(vector<int>& nums) {

long long ans = 0;

multiset<int> s;

int left = 0, right = 0, n = nums.size();

// 逐一遍历所有可能的子串

for (right = 0; right < n; right++) {

s.insert(nums[right]);

// 判断当前遍历的位置是否存在符合要求的子串，不符合的在set中删除

while (\*s.rbegin() - \*s.begin() > 2) {

auto pos = s.find(nums[left++]);

s.erase(pos);

}

// 计算当前的符合要求的子串的长度

long long len = right - left + 1;

ans += len;

}

return ans;

}

};

复杂度分析

时间复杂度：O(n)，其中n为nums的长度。

空间复杂度：O(1)。注意至多维护3个数，仅用到常量额外空间。