# 题目

给定一个整数数组，编写一个函数，找出索引m和n，只要将索引区间[m,n]的元素排好序，整个数组就是有序的。注意：n-m尽量最小，也就是说，找出符合条件的最短序列。函数返回值为[m,n]，若不存在这样的m和n（例如整个数组是有序的），请返回[-1,-1]。

示例：

输入： [1,2,4,7,10,11,7,12,6,7,16,18,19]

输出： [3,9]

提示：

0 <= len(array) <= 1000000

# 分析

## 方法一：两次遍历

思路：

要找到最小的无序子数组，我们可以从左到右遍历数组，找到第一个破坏升序排序的元素，这个元素将成为子数组的最小值。然后，我们从右到左遍历数组，找到第一个破坏降序排序的元素，这个元素将成为子数组的最大值。最后，我们在找到的最小值和最大值之间寻找最小和最大的元素，它们的索引就是我们要找的 m 和 n。

代码：

class Solution {

public:

vector<int> subSort(vector<int>& array) {

int n = array.size();

if (n == 0) {

return {-1, -1};

}

int left = -1, right = -1;

int min\_seen = INT\_MAX, max\_seen = INT\_MIN;

// Find the first element out of sorting order from the left

for (int i = 0; i < n - 1; ++i) {

if (array[i] > array[i + 1]) {

left = i;

min\_seen = min(min\_seen, array[i + 1]);

}

}

// Find the first element out of sorting order from the right

for (int i = n - 1; i > 0; --i) {

if (array[i] < array[i - 1]) {

right = i;

max\_seen = max(max\_seen, array[i - 1]);

}

}

if (left == -1 && right == -1) {

return {-1, -1};

}

// Find the correct position for the left index

for (int i = 0; i < n; ++i) {

if (array[i] > min\_seen) {

left = i;

break;

}

}

// Find the correct position for the right index

for (int i = n - 1; i >= 0; --i) {

if (array[i] < max\_seen) {

right = i;

break;

}

}

return {left, right};

}

};