# 题目

在数组 A 和数组 B 中，对于第一个满足 A[i] != B[i] 的索引 i ，当 A[i] > B[i] 时，数组 A 大于数组 B。

例如，对于索引从 0 开始的数组：

[1,3,2,4] > [1,2,2,4] ，因为在索引 1 上， 3 > 2。

[1,4,4,4] < [2,1,1,1] ，因为在索引 0 上， 1 < 2。

一个数组的子数组是原数组上的一个连续子序列。

给定一个包含不同整数的整数类型数组 nums ，返回 nums 中长度为 k 的最大子数组。

示例 1:

输入: nums = [1,4,5,2,3], k = 3

输出: [5,2,3]

解释: 长度为 3 的子数组有： [1,4,5]、 [4,5,2] 和 [5,2,3]。

在这些数组中， [5,2,3] 是最大的。

Example 2:

输入: nums = [1,4,5,2,3], k = 4

输出: [4,5,2,3]

解释: 长度为 4 的子数组有： [1,4,5,2] 和 [4,5,2,3]。

在这些数组中， [4,5,2,3] 是最大的。

示例 3:

输入: nums = [1,4,5,2,3], k = 1

输出: [5]

提示：

1 <= k <= nums.length <= 105

1 <= nums[i] <= 109

nums 中的所有整数都是不同的。

进阶：如果允许 nums 中存在相同元素，你该如何解决该问题？

# 分析

## 方法一

找出前n−k+1 个元素中最大的元素作为数组的首元素即可。

class Solution {

public:

vector<int> largestSubarray(vector<int>& nums, int k) {

auto pos = max\_element(nums.begin(), nums.begin() + nums.size() - k + 1);

return {pos, pos + k};

}

};

## 方法二

思路和心得：

1、最右只能是倒数第k个，否则就不满足长度为k了

2、找前面的最大值，作为起点数

3、设计好变量，做好统计

**代码：**

class Solution

{

public:

vector<int> largestSubarray(vector<int>& nums, int k)

{

int n = nums.size();

int max\_num = nums[n-k], max\_idx = n-k; //最大的那个数

for(int i = n-k-1; i > -1; i--) //最右只能是倒数第k个。从右往前找

{

if (nums[i] > max\_num)

{

max\_num = nums[i];

max\_idx = i;

}

}

return vector<int>(nums.begin() + max\_idx, nums.begin() + max\_idx + k); //切片

}

};