# 题目

给定一个数组 nums 和滑动窗口的大小 k，请找出所有滑动窗口里的最大值。

**示例:**

输入: nums = [1,3,-1,-3,5,3,6,7], 和 k = 3

输出: [3,3,5,5,6,7]

解释:

滑动窗口的位置 最大值

[1 3 -1] -3 5 3 6 7 3

1 [3 -1 -3] 5 3 6 7 3

1 3 [-1 -3 5] 3 6 7 5

1 3 -1 [-3 5 3] 6 7 5

1 3 -1 -3 [5 3 6] 7 6

1 3 -1 -3 5 [3 6 7] 7

**提示：**

你可以假设 k 总是有效的，在输入数组不为空的情况下，1 ≤ k ≤ 输入数组的大小。

注意：本题与主站 239 题相同：

https://leetcode-cn.com/problems/sliding-window-maximum/

# 分析

## 方法一：暴力破解

**思路：**

每次窗口滑动到一个位置就去计算一下这个窗口里的最大值，时间复杂度为 O(nk)。

**代码：**

class Solution {

public:

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

if(nums.size() == 0) return {};

vector<int> ans;

for(int i = 0; i <= nums.size() - k; i ++){

int maxNum = nums[i];

for(int j = i; j < i + k; j ++){

maxNum = max(maxNum, nums[j]);

}

ans.push\_back(maxNum);

}

return ans;

}

};

## 方法二：双端队列

**思路：**

为了降低时间复杂度，我们观察一下窗口移动的过程类似于队列出队入队的过程，每次队尾出一个元素，然后队头插入一个元素，求该队列中的最大值，这个和“包含min函数的栈”有点类似，我们可以维护一个**递减队列**，队列用来保存可能是最大值的数字的index。当前窗口最大值的index在队首，当窗口滑动时，会进入一个新值，出去一个旧值，我们需要给出当前窗口的最大值。

1、需要先检查队首（上一窗口的最大值）的index是否还在当前窗口内，如果不在的话需要淘汰。

2、然后新进入的值要和队尾元素做比较，如果比队尾元素大，那么队尾元素出队（用到双端队列特性的地方），直到队列为空或者前面的值不比他小为止。

**代码：**

class Solution {

public:

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

vector<int> ans;

deque<int> deq;

int n = nums.size();

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

//新元素入队时如果比队尾元素大的话就替代队尾元素

while(!deq.empty() && nums[i] > nums[deq.back()]){

deq.pop\_back();

}

//检查队首的index是否在窗口内，不在的话需要出队

if (!deq.empty() && deq.front() < i - k + 1){

deq.pop\_front();

}

deq.push\_back(i);//移动窗口，不断在队尾添加元素

if (i >= k -1)

ans.push\_back(nums[deq.front()]); //获取前k个最大数

}

return ans;

}

};

时间复杂度O(N)

## 方法三：单调栈