```
import Foundation
/// **剑指 Offer 59 - I. 滑动窗口的最大值**
/// 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
/// ---
                                   3`
/// `[1
       3 -11 -3
                  5 3 6 7
   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`
class Solution {
   func maxSlidingWindow(_ nums: [Int], _ k: Int) -> [Int] {
       if (k \le 0 \mid k > nums.count \mid nums.isEmpty) { return [] }
       /// 辅助队列**存放的是索引**, 队列的第一个数永远是当前窗口内数组的最大值
       var queue: [Int] = []
       /// 存放结果
       var res: [Int] = []
       /// **此处使用 `nums.enumerated()` 将导致超时**
       for id in 0..<nums.count {
           /// 移除所有小干 value 的元素
           while !queue.isEmpty && nums[queue.last!] <= nums[id] {</pre>
            queue.removeLast() }
           /// 添加新元素的索引
           queue.append(id)
           /// 判断是否形成滑动窗口
           if (id >= k-1) {
               res.append(nums[queue.first!])
           }
           /// 判断当前索引是否已经超过最大值的索引,如果已经超过则删除前一次的最大值
           if (id - queue.first! >= k - 1) {
               queue.removeFirst()
           }
       }
       return res
   }
}
let solution = Solution()
print(solution.maxSlidingWindow([1,3,-1,-3,5,3,6,7], 3))
print(solution.maxSlidingWindow([], 3))
print(solution.maxSlidingWindow([7,2,4], 2))
```