1. Sliding Window Maximum

Given an array *nums*, there is a sliding window of size *k* which is moving from the very left of the array to the very right. You can only see the *k* numbers in the window. Each time the sliding window moves right by one position. Return the max sliding window.

**Follow up:** Could you solve it in linear time?

**Example:**

Input: nums = [1,3,-1,-3,5,3,6,7], and k = 3  
Output: [3,3,5,5,6,7]   
Explanation:   
  
Window position Max  
--------------- -----  
[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

**解** 很奇怪很多答案为什么又是堆又是哈希表的。。。

用一个tmp\_max变量记录窗口里的最大值，向右移动时，如果最左边出去的是最大值tmp\_max，就在窗口里重新线性搜索或者调用max\_element()或者其他什么方法找到最大值，右端点进来的新的值和tmp\_max比较一下取大的作为新的tmp\_max

class Solution {  
public:  
 vector<int> maxSlidingWindow(vector<int>& nums, int k) {  
 int l = 0, r = k;  
 vector<int>ans;  
 int tmp = INT\_MIN;  
 for(int i = 0; i < r; ++i)tmp = max(nums[i], tmp);  
 ans.push\_back(tmp);  
 r++;  
 l++;  
 while(r <= nums.size()){  
 if(tmp == nums[l-1]){  
 tmp = INT\_MIN;  
 for(int i = l; i < r; ++i)tmp = max(tmp, nums[i]);  
 }else{  
 tmp = max(tmp, nums[r-1]);  
 }  
 ans.push\_back(tmp);  
 l++;  
 r++;  
 }  
 return ans;  
 }  
};