239. Sliding Window Maximum

Hard 1223 73 Favorite Share

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.

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
```

Note:
You may assume *k* is always valid, 1 ≤ k ≤ input array's size for non-empty array.

Follow up:
Could you solve it in linear time?

```java
package Algorithm;

import java.util.Collections;
import java.util.PriorityQueue;
/*
 * 此道题目可以维护一个优先队列作为一个大顶堆，堆顶就是最大的数
 */
public class L239 {

    public int[] maxSlidingWindow(int[] nums, int k) {
        if(nums.length == 0 || nums == null) {
            return new int[0];
        }
        int [] result = new int [nums.length + 1 - k];
        PriorityQueue<Integer> pq = new PriorityQueue<Integer>(Collections.reverseOrder());
        for(int i = 0; i < nums.length; i++) {
            //去掉最左边的数
            if (i >= k)
                pq.remove(nums[i - k]);
            //将新的数加入到窗口中
            pq.offer(nums[i]);
            //堆顶就是最大值
            if(i + 1 >= k) {
                result[i - k + 1] = pq.peek();
            }
        }
        return result;
    }

}

```