题目 滑动窗口的最大值

考点 栈和队列 热点指数 28352 通过率 23.84%

具体题目

给定一个数组和滑动窗口的大小，找出所有滑动窗口里数值的最大值。例如，如果输入数组{2,3,4,2,6,2,5,1}及滑动窗口的大小3，那么一共存在6个滑动窗口，他们的最大值分别为{4,4,6,6,6,5}； 针对数组{2,3,4,2,6,2,5,1}的滑动窗口有以下6个： {[2,3,4],2,6,2,5,1}， {2,[3,4,2],6,2,5,1}， {2,3,[4,2,6],2,5,1}， {2,3,4,[2,6,2],5,1}， {2,3,4,2,[6,2,5],1}， {2,3,4,2,6,[2,5,1]}。

```
import java.util.ArrayList;
import java.util.Deque;
import java.util.LinkedList;
import java.util.PriorityQueue;
/**
 * 滑动窗口的最大值
 * 给定一个数组和滑动窗口的大小，找出所有滑动窗口里数值的最大值。例如，如果输入数组{2,3,4,2,6,2,5,1}及滑动窗口的大小3，那么一共存在6个滑动窗口，他们的最大值分别为{4,4,6,6,6,5}； 针对数组{2,3,4,2,6,2,5,1}的滑动窗口有以下6个： {[2,3,4],2,6,2,5,1}， {2,[3,4,2],6,2,5,1}， {2,3,[4,2,6],2,5,1}， {2,3,4,[2,6,2],5,1}， {2,3,4,2,[6,2,5],1}， {2,3,4,2,6,[2,5,1]}。
 */
public class Solution52 {
    public static void main(String[] args) {
        Solution52 solution52 = new Solution52();
        int[] num = {2, 3, 4, 2, 6, 2, 5, 1};
        int size = 3;
        ArrayList list = solution52.maxInWindows(num, size);
        System.out.println(list);
    }
    /**
     * 最大堆方法
     * 构建一个窗口size大小的最大堆，每次从堆中取出窗口的最大值，随着窗口往右滑动，需要将堆中不属于窗口的堆顶元素删除。
     *
     * [@param num
     * @param size
     * @return](/profile/547241) */
    public ArrayList maxInWindows_2(int[] num, int size) {
        ArrayList res = new ArrayList();
        if (size > num.length || size < 1) return res;
        // 构建最大堆，即堆顶元素是堆的最大值。
        PriorityQueue heap = new PriorityQueue((o1, o2) -> o2 - o1);
        for (int i = 0; i < size; i++) heap.add(num[i]);
        res.add(heap.peek());
        for (int i = 1; i + size - 1 < num.length; i++) {
            heap.remove(num[i - 1]);
            heap.add(num[i + size - 1]);
            res.add(heap.peek());
        }
        return res;
    }
    /**
     * 双队列方法
     * 滑动窗口的最大值总是保存在队列首部，队列里面的数据总是从大到小排列。
     *
```

```
     * [@param num
     * @param size
     * @return](/profile/547241) */
    public ArrayList maxInWindows(int[] num, int size) {
        ArrayList res = new ArrayList();
        if (num == null || num.length == 0 || size == 0 || size > num.length) {
            return res;
        }
        Deque deque = new LinkedList();
        for (int i = 0; i < num.length; i++) {
            if (!deque.isEmpty()) {
                // 如果队列头元素不在滑动窗口中了，就删除头元素
                if (i >= deque.peek() + size) {
                    deque.pop();
                }
                // 如果当前数字大于队列尾，则删除队列尾，直到当前数字小于等于队列尾，或者队列空
                while (!deque.isEmpty() && num[i] >= num[deque.getLast()]) {
                    deque.removeLast();
                }
            }
            deque.offer(i); // 入队列
            // 滑动窗口经过一个滑动窗口的大小，就获取当前的最大值，也就是队列的头元素
            if (i + 1 >= size) {
                res.add(num[deque.peek()]);
            }
        }
        return res;
    }
}
```