栈与队列

栈：后进先出

队列：先进先出

栈和队列的常见写法

（1）栈和队列的声明：

栈：Deque<Integer> inStack = new ArrayDeque<Integer>(); 或 Deque<Integer> inStack = new LinkedList<Integer>();

队列：把Deque换成queue即可

（2）栈和队列加入/移出元素：

栈：inStack.push(x); / 栈顶元素弹出（同时也能访问栈顶值）outStack.pop();

队列：queue.offer(x); / 队列头元素移出（同时也能访问栈顶值） queue.poll();

（3）访问栈和队列的栈顶/队列首部元素：outStack.peek(); / queue.peek();

（4）判断栈和队列是否为空：outStack.isEmpty(); / queue.isEmpty();

用栈实现队列 & 用队列实现栈

属于栈和队列基本功能的熟悉和掌握，题目细节看leetcode对应题目

题目：

232. 用栈实现队列（简单）

235. 用队列实现栈（简单）

有效的括号

字符串中的**对称匹配问题**，应当想到用栈

本题关键点：

1、讨论无效括号的三种情况：

（1）左括号多了：([{ }] ( )，遍历完后栈不为空

（2）左右括号不匹配：[{ (} }]

（3）右括号多了：[{ ( ) }])))，遍历到右括号时存在栈为空的现象

2、遍历到左括号时，是将它对应的右括号压入栈（方便后续匹配）

（也可以用哈希表存储键值对，但要注意写法）

3、遍历到右括号不匹配时，注意（2）和（3）情况的合并，关系为“或”且对栈是否为空的判断应该写在前；匹配则把栈顶元素弹出即可

题目：

20. 有效的括号（简单）

删除字符串中的所有相邻重复项

和上题类似，同样属于字符串匹配问题，应当想到栈

（这里的匹配是相等与不相等层面的，有点像“**开心消消乐**”）

当入栈后没有重复项后，由于栈是后进先出，要得到最终结果，需要转一下顺序，此时联系前面“用栈实现队列”一题，可以再用第二个栈，来实现最后的输出

注意：要用到StringBuilder来编辑字符串，最后还要转回String

题目：

1047. 删除字符串中的所有相邻重复项（简单）

逆波兰表达式求值

当遇到字符串是符号时，由于“后”遍历的数字“先”进行操作，因此要想到栈

具体操作：

如果字符串是整数，则压入栈；如果字符串是表达符号，则将栈顶元素弹出两次，对这两个数进行操作（注意“-”和“/”的特殊处理），将操作结果又压入栈

由于栈只存储数字，则可以设置栈的类型为Integer

注意：

（1）将字符串s的类型转为整型：Integer.valueOf(s)

（2）字符串的相等判断不能只用==，要用s.equals(”+”)

题目：

150. 逆波兰表达式求值（中等）

滑动窗口最大值

本题暗含了“先进先出”的过程，因此要想到队列

单调队列：使队列元素单调递减，队列储存可能成为滑动窗口最大值的元素

本题有三大步骤：

（1）pop被移出窗口的元素（如果队列头元素不是被移出窗口的元素，则不进行该操作）

相关操作：deque.peekFirst() == nums[i - 1]；deque.removeFirst();

（2）push新进入滑动窗口的元素（如果这个元素比队列末尾元素大，则需将末尾元素弹出再加入，因为它一定不会成为滑动窗口的最大值）

相关操作：deque.peekLast() < nums[j]；deque.removeLast()；
deque.addLast(nums[j]);

（3）返回每个滑动窗口的最大值

相关操作：res[i] = deque.peekFirst();

题目细节看leetcode对应题目，注意一开始滑动窗口的末端是在数组的开头元素（索引-2到0）

题目：

239. 滑动窗口最大值（困难）

前K个高频元素

本题的三大步骤：

（1）储存数组元素值以及对应出现频率——用map哈希表

（2）对频率进行排序

（3）找出前k个高频元素

针对步骤（2）和（3），本题采用**优先级队列**

优先级队列本质是一个堆，它能自动按从大到小/从小到大的顺序排列元素

算法流程：

本题采用小顶堆储存元素值-频率的pair，且队列元素的频率从小到大排列；

堆中只维护k个元素，如果k个元素填满了堆，访问到哈希表的value比堆头元素对应value大，则将头元素的pair移出队列，再将自身加入队列；

最后按倒序索引输出元素值

相关操作的用法：

（1）堆的声明：

小顶堆：PriorityQueue<int[]> pq = new PriorityQueue<>((pair1, pair2) → pair1[1] - pair2[1]);

大顶堆：PriorityQueue<int[]> pq = new PriorityQueue<>((pair1, pair2) → pair2[1] - pair1[1]);

（2）访问哈希表的键值对：Map.Entry<Integer, Integer> entry : map.entrySet()

（3）往堆添加键值对：pq.add(new int[]{entry.getKey(), entry.getValue()});

（4）判断是否比堆头元素的value大：entry.getValue() > pq.peek()[1];

（5）移除堆的元素，即整个pair：pq.poll();

（6）输出pair的键，即数组元素值：ans[i] = pq.poll()[0];

题目细节看leetcode对应题目

题目：

347. 前K个高频元素（中等）