**难度: 中等**

输入两个整数序列，第一个序列表示栈的压入顺序，请判断第二个序列是否为该栈的弹出顺序。假设压入栈的所有数字均不相等。例如，序列 {1,2,3,4,5} 是某栈的压栈序列，序列 {4,5,3,2,1} 是该压栈序列对应的一个弹出序列，但 {4,3,5,1,2} 就不可能是该压栈序列的弹出序列。

示例 1：

输入：pushed = [1,2,3,4,5], popped = [4,5,3,2,1]

输出：true

解释：我们可以按以下顺序执行：

push(1), push(2), push(3), push(4), pop() -> 4,

push(5), pop() -> 5, pop() -> 3, pop() -> 2, pop() -> 1

示例 2：

输入：pushed = [1,2,3,4,5], popped = [4,3,5,1,2]

输出：false

解释：1 不能在 2 之前弹出。

提示：

0 <= pushed.length == popped.length <= 1000

0 <= pushed[i], popped[i] < 1000

pushed 是 popped 的排列。

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/zhan-de-ya-ru-dan-chu-xu-lie-lcof

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**思路:**

1.创建一个栈用来模拟入栈退栈的值

2. 循环入栈,然后每次判断自己第一个入栈的值,是否等于需要测试抛出栈的顺序

3). 每次抛出计数的+1

4). 最后判断长度计数的长度是否是最后一个,也就说等于需要抛出栈的长度

|  |
| --- |
| public boolean validateStackSequences(int[] pushed, int[] popped) {  if(pushed.length != popped.length) {  return false;  }  //创建一个栈  Deque<Integer> stack = new ArrayDeque<Integer>();  int i = 0;  //循环入栈  for (int elem : pushed) {  //入栈  stack.push(elem);  //判断栈是否为空, 偷窥栈顶的元素判断是否等于需要比较的栈中的元素  while(!stack.isEmpty() && stack.peek() == popped[i]) {  //抛出  stack.pop();  //比较下一个节点  i++;  }  }  //判断是否相等  return i == popped.length;  } |