画解剑指 Offer - LeetBook - 力扣（LeetCode）全球极客挚爱的技术成长平台

剑指 Offer 31. 栈的压入、弹出序列 - 解题思路

题目描述

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。
```

限制：

1. `0 <= pushed.length == popped.length <= 1000`
2. `0 <= pushed[i], popped[i] < 1000`
3. `pushed` 是 `popped` 的排列。

解题方案

思路

- 标签：模拟
- 整体思路：

借用一个辅助栈 `stackstack` ，模拟 压入 / 弹出操作的排列。根据是否模拟成功，即可得到结果。

复杂度：

- 时间复杂度：O(n)O(n)。 nn 为入栈序列的长度
- 空间复杂度：O(n)O(n)： 辅助栈最多存储 nn 个元素

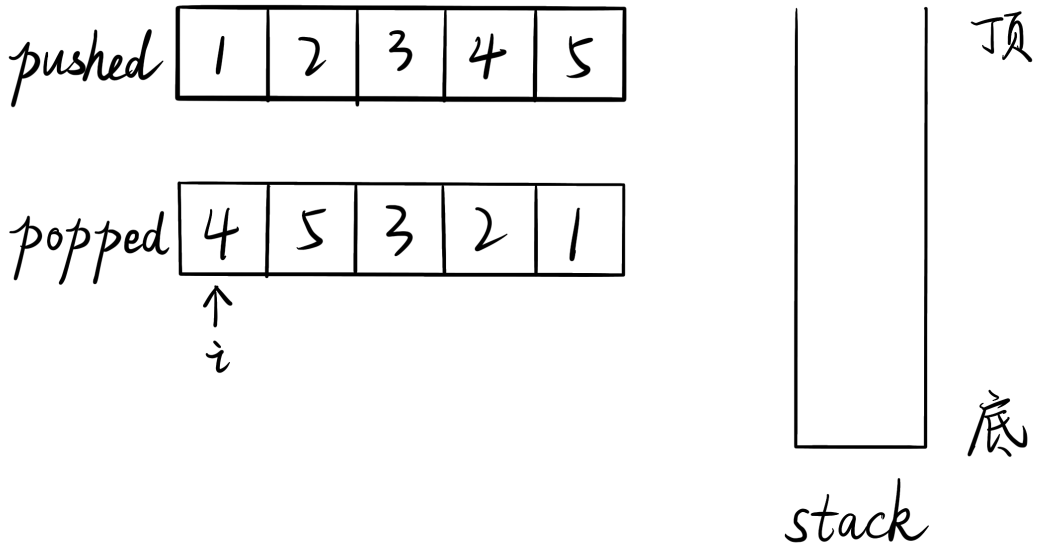
算法流程

1. 建立一个辅助栈
2. 遍历入栈序列
    - 元素入栈
    - 若辅助栈栈顶元素等于弹出序列元素，则执行出栈操作
3. 返回结果

代码

```
class Solution {
    public boolean validateStackSequences(int[] pushed, int[] popped) {
        Stack<Integer> stack = new Stack<>();
        int i = 0;
        for(int element : pushed) {
            stack.push(element); // num 入栈
            while(!stack.isEmpty() && stack.peek() == popped[i]) { // 循环判断与出栈
                stack.pop();
                i++;
            }
        }
        return stack.isEmpty();
    }
}
```

画解



1 / 13

花絮

讨论区

共 3 个讨论

最热

笨熊 L5

来自广东2021-07-04

这方法比较多的，但是最好的应该是对于每个栈维护一个指针，因为都是数组，然后push指向入栈，pop指向出栈，一次遍历即可，出栈入栈只需要对指针就是指向元素的地址++--就好，这样时间复杂度O(n), 空间复杂度O(1)，是直接100%击败的

```
public boolean validateStackSequences(int[] pushed,int[] popped){
    int push = 0, pop = 0;
    for(int i = 0; i < pushed.length; i++){
        pushed[push] = pushed[i];
        while(push >= 0 && pushed[push] == popped[pop]){
            push--; pop++;
        }
        push++;
    }
    return push == 0;
}
```

8

MGFangel L3
来自上海2022-05-29

这方法比较多的，但是最好的应该是对于每个栈维护一个指针，因为都是数组，然后push指向入栈，pop指向出栈，一次遍历即可，出栈入栈只需要对指针就是指向元素的地址++--就好，这样时间复杂度O(n), 空间复杂度O(1)，是直接100%击败的 「代码块」

秒的不能在秒了^_^

woitaylor L1
来自广东2021-03-22

灵魂画家