946. Validate Stack Sequences

16 March 2022 03:25 PM

Given two integer arrays `pushed` and `popped` each with distinct values, return `true` *if this could have been the result of a sequence of push and pop operations on an initially empty stack, or* `false` *otherwise.*

Example 1:

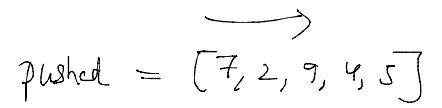
```
Input: pushed = [1,2,3,4,5], popped =
[4,5,3,2,1]
Output: true
Explanation: We might do the following sequence:
push(1), push(2), push(3), push(4),
pop() -> 4,
push(5),
pop() -> 5, pop() -> 3, pop() -> 2, pop() -> 1
```

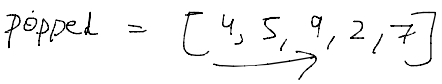
Example 2:

```
Input: pushed = [1,2,3,4,5], popped =
[4,3,5,1,2]
Output: false
Explanation: 1 cannot be popped before 2.
```

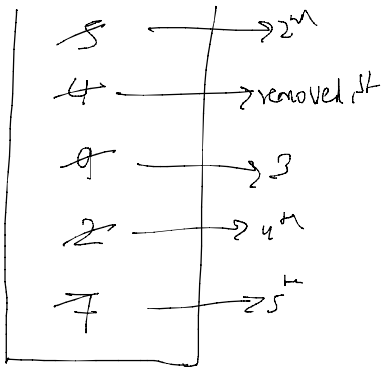
Constraints:

- `1 <= pushed.length <= 1000`
- `0 <= pushed[i] <= 1000`
- All the elements of `pushed` are **unique**.
- `popped.length == pushed.length`
- `popped` is a permutation of `pushed`.

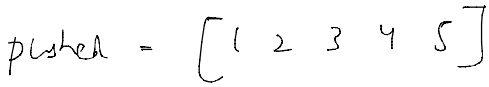


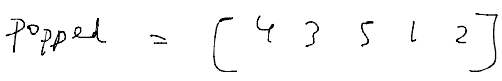


pop ⇒ 4, 5, 9, 2, 7

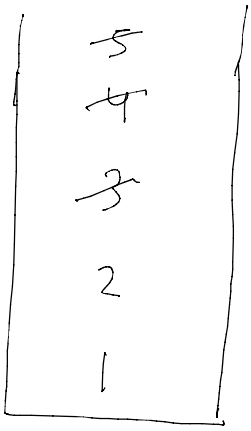


It's poped in correct order so true.





popped = [4 3 5 1 2]



4 3 5 2 1

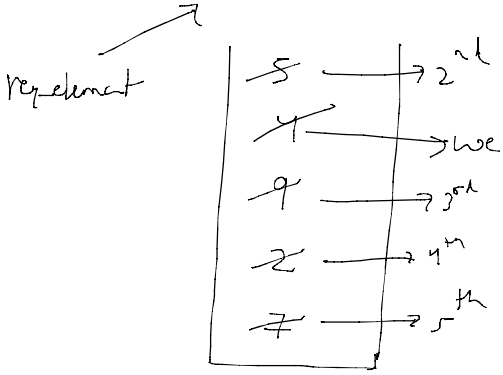
now if we start to pop from 4 we start in sequence order 4 3, now push 5 and pop 5

Now, at stack top is 2 but in pop it's 1 so it's false.

---

pushed ⇒ [7 2 9 4 5] (we can't decide this)

popped ⇒ [4 5 9 2 7] (we can decide this)



4 5 9 2 7 → pop == popped

True

and same as top() element

* When we have required element to pop(), that's the best time to start popping.

```
st.peek == popped[j] ⟹ while
    st.pop()
    j++
```

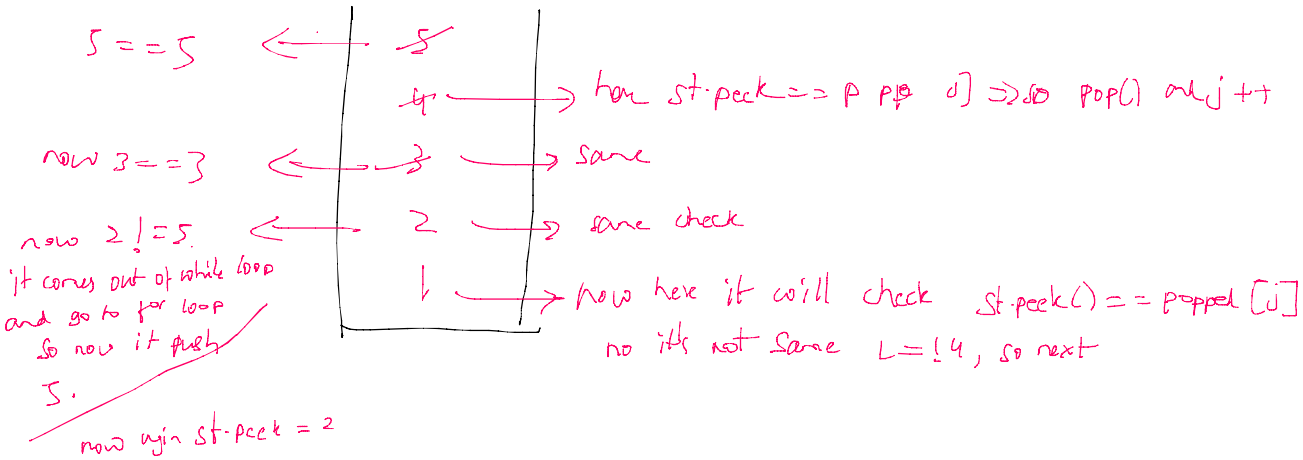
```java
class Solution {
    public boolean validateStackSequences(int[] pushed, int[] popped) {

        Stack<Integer> st = new Stack<>();
        int j = 0;
        for(int e: pushed){
            st.push(e);
            while(st.size() > 0 && st.peek() == popped[j]){
                st.pop();
                j++;
            }
        }
        return j == popped.length;
    }
}
```

Dry Run

pushed ⇒ [1 2 3 4 5]

popped ⇒ [4 3 5 1 2]
j j j j

```java
Stack<Integer> st = new Stack<>();
int j = 0;
for(int e: pushed){
    st.push(e);
    while(st.size() > 0 && st.peek() == popped[j]){
        st.pop();
        j++;
    }
}
return j == popped.length;
```



now agin st.peek = 2
and popped[j] = 1

still it not end. it come out of loop, so it check now whether j checked all the element or not (return j == popped.length) no it didn't so false.

[4 3 5 1 2]
j j j j
↳ j is still in 1 and wouldn't complete.

T.C ⇒ O(N)

S.C ⇒ O(N)