

面试题31：栈的压入、弹出序列

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可能为该栈的弹出顺序。假设压入栈的所有数字均不相等。例如序列1,2,3,4,5是某栈的压入顺序，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。（注意：这两个序列的长度是相等的）

解题思路

总结上述入栈、出栈的过程，我们可以找到判断一个序列是不是栈的弹出序列的规律：如果下一个弹出的数字刚好是栈顶数字，那么直接弹出；如果下一个弹出的数字不在栈顶，则把压栈序列中还没有入栈的数字压入辅助栈，直到把下一个需要弹出的数字压入栈顶为止；如果所有数字都压入栈后仍然没有找到下一个弹出的数字，那么该序列不可能是一个弹出序列。

C++

```
1 class Solution {
2 public:
3     bool IsPopOrder(vector<int>
pushV,vector<int> popV) {
4         if(pushV.empty() || popV.empty())
```

```

5         return false;
6         vector<int> stack;
7         for(int j = 0,i = 0;i <
pushV.size();i++){
8             stack.push_back(pushV[i]);
9             while(j < popV.size() &&
stack.back() == popV[j]){
10                 stack.pop_back();
11                 j++;
12             }
13         }
14         return stack.empty();
15     }
16 };

```

Java

```

1 import java.util.ArrayList;
2
3 public class Solution {
4     public boolean IsPopOrder(int [] pushA,int
[] popA) {
5         if(pushA.length == 0||popA.length == 0)
6             return false;
7         ArrayList<Integer> list = new
ArrayList<Integer>();
8         //用于标识弹出序列的位置

```

```

9         int j = 0;
10        for(int i = 0;i < pushA.length;i++){
11            if(pushA[i] != popA[j])
12                list.add(pushA[i]);
13            else
14                j++;
15        }
16        for(int i = list.size() - 1;i >= 0;i--){
17            if(list.get(i) != popA[j])
18                return false;
19            j++;
20        }
21        return true;
22    }
23 }

```

Java

```

1 import java.util.ArrayList;
2 import java.util.Stack;
3 public class Solution {
4     public boolean IsPopOrder(int [] pushA,int
5     [] popA) {
6         if(pushA.length == 0 || popA.length == 0)
7             return false;
8         Stack<Integer> s = new Stack<Integer>();

```

```

8      //用于标识弹出序列的位置
9      int popIndex = 0;
10     for(int i = 0;i < pushA.length;i++){
11         s.push(pushA[i]);
12         //如果栈不为空，且栈顶元素等于弹出序列
13         while(!s.empty() && s.peek() ==
popA[popIndex]){
14             //出栈
15             s.pop();
16             //弹出序列向后一位
17             popIndex++;
18         }
19     }
20     return s.empty();
21 }
22 }

```

Python 2.7.3

```

1  # -*- coding:utf-8 -*-
2  class Solution:
3      def IsPopOrder(self, pushV, popV):
4          # write code here
5          stack = []
6          for psh in pushV:
7              stack.append(psh)
8              if stack[-1] == popV[0]:

```

```
9         popV.pop(0)
10         stack.pop()
11     for pp in popV:
12         if pp == stack[-1]:
13             stack.pop()
14     return stack == []
```