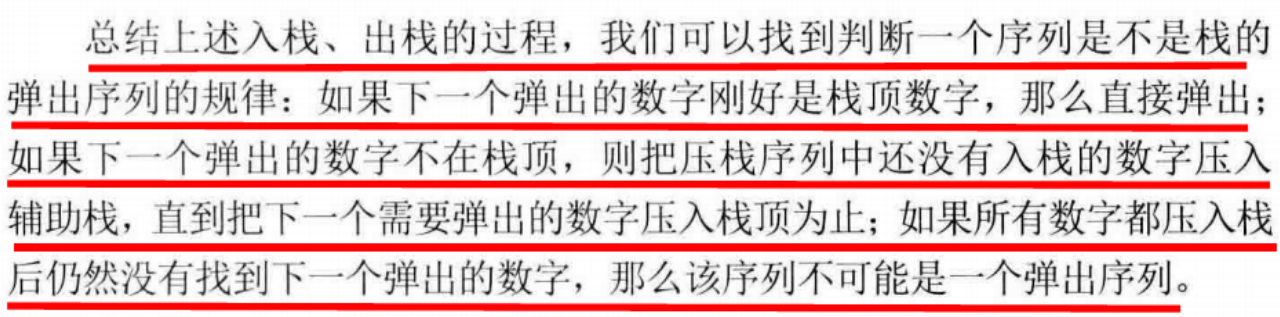
面试题31：栈的压入、弹出序列

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



C++

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

```
            return false;
        vector<int> stack;
        for(int j = 0,i = 0;i < pushV.size();i++){
            stack.push_back(pushV[i]);
            while(j < popV.size() && stack.back() == popV[j]){
                stack.pop_back();
                j++;
            }
        }
        return stack.empty();
    }
};
```

Java

```
import java.util.ArrayList;

public class Solution {
    public boolean IsPopOrder(int [] pushA,int [] popA) {
        if(pushA.length == 0||popA.length == 0)
            return false;
        ArrayList<Integer> list  = new ArrayList<Integer>();
        //用于标识弹出序列的位置
```

```
        int j = 0;
        for(int i = 0;i < pushA.length;i++){
            if(pushA[i] != popA[j])
                list.add(pushA[i]);
            else
                j++;
        }
        for(int i = list.size() - 1;i >= 0;i--){
            if(list.get(i) != popA[j])
                return false;
            j++;
        }
        return true;
    }
}
```

Java

```
import java.util.ArrayList;
import java.util.Stack;
public class Solution {
    public boolean IsPopOrder(int [] pushA,int [] popA) {
      if(pushA.length == 0 || popA.length == 0)
          return false;
        Stack<Integer> s = new Stack<Integer>();
```

```
        //用于标识弹出序列的位置
        int popIndex = 0;
        for(int i = 0;i < pushA.length;i++){
            s.push(pushA[i]);
            //如果栈不为空，且栈顶元素等于弹出序列
            while(!s.empty() && s.peek() ==
popA[popIndex]){
                //出栈
                s.pop();
                //弹出序列向后一位
                popIndex++;
            }
        }
        return s.empty();
    }
}
```

Python 2.7.3

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

```
                popV.pop(0)
                stack.pop()
        for pp in popV:
            if pp == stack[-1]:
                stack.pop()
        return stack == []
```