## 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：

 借用一个辅助的栈，遍历压栈顺序，先讲第一个放入栈中，这里是1，然后判断栈顶元素是不是出栈顺序的第一个元素，这里是4，很显然1≠4，所以我们继续压栈，直到相等以后开始出栈，出栈一个元素，则将出栈顺序向后移动一位，直到不相等，这样循环等压栈顺序遍历完成，如果辅助栈还不为空，说明弹出序列不是该栈的弹出顺序。

## 算法描述：

class Solution {

public:

bool IsPopOrder(vector<int> pushV,vector<int> popV) {

if (pushV.size() == 0)

return false;

stack<int> stack;

int i=0,j=0;

for (i=0; i<pushV.size(); i++)

{

stack.push(pushV[i]);

while (j<popV.size() && popV[j] == stack.top())

{

stack.pop();

j++;

}

}

return stack.empty();

}

};