链滴

包含 min 函数的栈, 栈的压入、弹出序列

作者：ReyRen
原文链接：https://ld246.com/article/1569736776537
来源网站：链滴

题目一:

定义栈的数据结构, 请在该类型中实现一个能够得到栈中所含最小元素的min函数(时间复杂度应为O(1)).

时间限制:

1秒

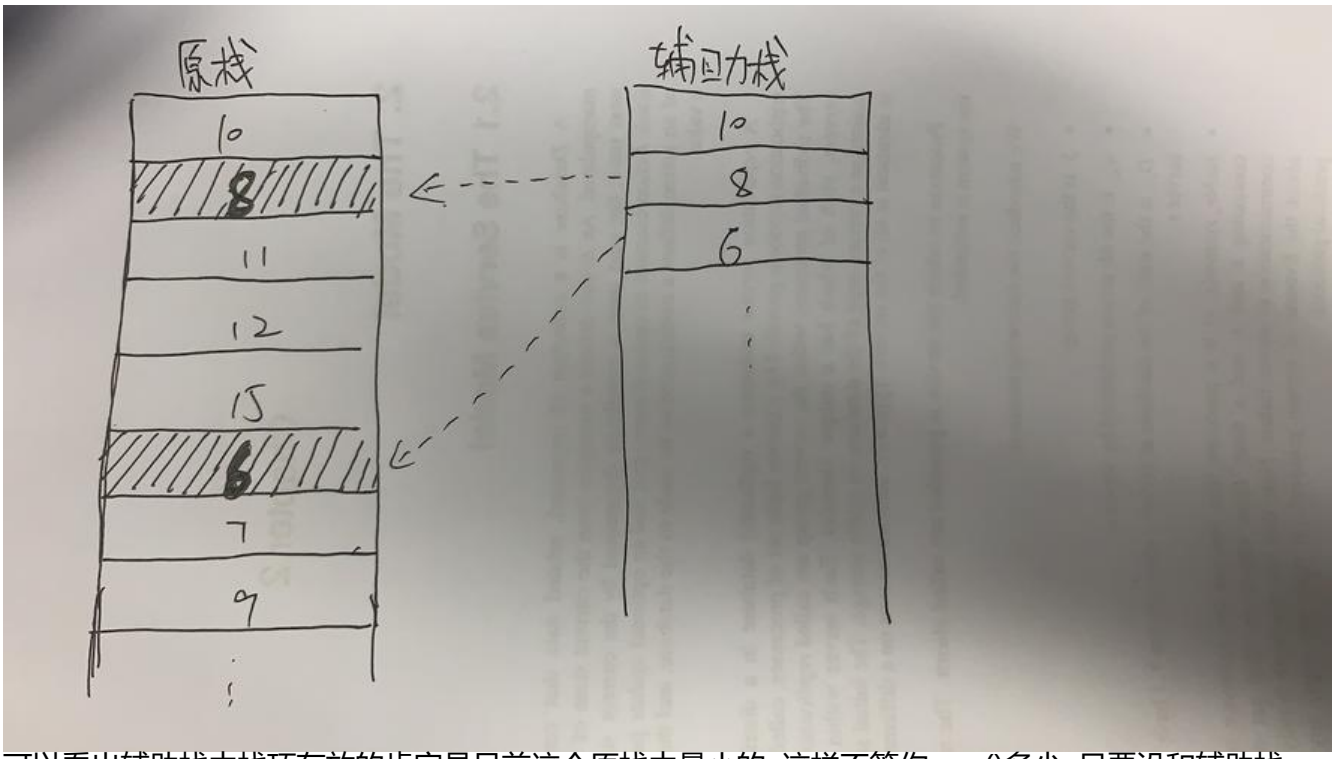
空间限制:

32768K

解题思路:

首先理解题意, 题目的意思是说要stack.min()能获取到这个栈的最小元素. 这样的话, 就需要有东西来录这个最小元素. 但是也不能只记录一个最小值, 因为虽然取完当前的最小值了, 紧接着来了几波pop(), 要保证下个min()是当前剩余的里面最小的.

所以最好的方式就是使用一个辅助栈, 第一个栈就是正常押入, 但是辅助栈中存储的就是压栈中最小的, 一旦再次压入发现比之前的还小, 那么就同时压入辅助栈中. 上个图吧:



可以看出辅助栈中栈顶存放的肯定是目前这个原栈中最小的, 这样不管你pop()多少, 只要没和辅助栈的top()相等, 那么最小值仍然是辅助栈中的top(). 但是一旦pop()的相等了，那就辅助栈中也将栈顶pop)出. 这样就能保证目前的原栈中的最小值还是在辅助栈中.

```
class Solution {
    stack<int> stack1, stack2;
    void push(int value) {
        stack1.push(value);
        if (stack2.empty()) {
            stack2.push(value);
        }
        else if (value <= stack2.top()) {
            stack2.push(value);
        }
    }
    int top() {
        return stack1.top();
    }
    void pop() {
        /*注意达到条件也得删2中的值*/
        if (stack1.top() == stack2.top()) {
            stack2.pop();
        }
        stack1.pop();
    }
    int min() {
        return stack2.top();
    }
}
```

题目二:

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)

时间限制:

1秒

空间限制:

32768K

解题思路:

先搞懂题, 起初我的理解是直接将A数组全部压入栈, 然后top()和pop()结合着看是不是和B数组完全一对应就行了.

只能说思维完全错误, 大脑紊乱. 题目中的例子就是活生生的大脸. 简单来说题目的意思就是有可能在栈过程中我就出栈, 然后继续压栈. 这样的话, 我们就需要一个辅助容器, vector和stack都行, 我这里使vector(在小知识点中分享了几个本题相关的vector好用的方法.). 当从A数组中进行压入时, 如果发现B数组的下标所对应的值相等, 那么对辅助栈进行循环pop, B数组下标不断向后移动. 旁白: 匹配上了? 让你出一波栈我在压. 当发现不相等的时候, A中的元素继续向辅助栈中压入, 再与B数组当前位置的值行比较....

```
class Solution {
public:
    bool IsPopOrder(vector<int> pushV,vector<int> popV) {
        if (pushVi.empty() || popV.empty()) {
            return false;
        }
        vector<int> tmpStack;
        for (int i = 0, j = 0; i < pushV.size(); i++) {
            tmpStack.push_back(pushV[i]);
            while (tmpStack.back() == popV[j] && j < popV.size()) {
                tmpStack.pop_back();
                j++;
            }
        }
        return tmpStack.empty();
    }
}
```

小知识点 □thumbsup :

vector中的常用方法:

- push_back 在数组的最后添加一个数据
- pop_back 去掉数组的最后一个数据
- back 得到数组的最后一个单元的引用