LeetCode-155

最小栈

要求

- 设计一个支持 push，pop，top 操作，并能在常数时间内检索到最小元素的栈。
  - push(x) -- 将元素 x 推入栈中。
  - pop() -- 删除栈顶的元素。
  - top() -- 获取栈顶元素。
  - getMin() -- 检索栈中的最小元素。

示例

MinStack minStack = new MinStack();
minStack.push(-2);
minStack.push(0);
minStack.push(-3);
minStack.getMin(); --> 返回 -3.
minStack.pop();
minStack.top(); --> 返回 0.
minStack.getMin(); --> 返回 -2.

思路

- 建立两个栈，一个用来存放push的元素，另一个放最小值，当有最小值pop出去时，最小值栈也应该pop出最上面的元素，注意小于等于当前最小值的数都应该进栈！

Python代码：

```
1
2  class MinStack:
3
4      def __init__(self):
5          """
6          initialize your data structure here.
7          """
8          self.stack = []
9          self.min_stack = []
10         self.stack_index = -1
11         self.min_stack_index = -1
12
13     def push(self, x):
14         """
15         :type x: int
16         :rtype: void
```

```
        """
        self.stack_index+=1
        self.stack.append(x)
        if self.min_stack_index == -1:
            self.min_stack.append(x)
            self.min_stack_index += 1
        else:
            if x <= self.min_stack[self.min_stack_index]:
                self.min_stack.append(x)
                self.min_stack_index += 1


    def pop(self):
        """
        :rtype: void
        """
        if self.stack_index > -1:
            a = self.stack.pop()
            self.stack_index -= 1
            if a == self.min_stack[self.min_stack_index]:
                self.min_stack.pop()
                self.min_stack_index -= 1
        else:
            pass


    def top(self):
        """
        :rtype: int
        """
        if self.stack_index > -1:
            return self.stack[self.stack_index]
        else:
            return None


    def getMin(self):
        """
        :rtype: int
        """
        return self.min_stack[self.min_stack_index]


# Your MinStack object will be instantiated and called as such:
# obj = MinStack()
# obj.push(x)
# obj.pop()
# param_3 = obj.top()
# param_4 = obj.getMin()
```

C++代码：

```
class MinStack {
public:
```

```
    /** initialize your data structure here. */
    stack<int> s;
    stack<int> min;

    void push(int x) {
        s.push(x);
        if(min.empty() || min.top()>= x){
            min.push(x);
        }
    }

    void pop() {
        if(s.top() == min.top()){
            s.pop();
            min.pop();
        }
        else{
            s.pop();
        }
    }

    int top() {
        return s.top();
    }

    int getMin() {
        return min.top();
    }
};

/**
 * Your MinStack object will be instantiated and called as such:
 * MinStack obj = new MinStack();
 * obj.push(x);
 * obj.pop();
 * int param_3 = obj.top();
 * int param_4 = obj.getMin();
 */
```