LeetCode #155 最小栈

作者：matthewhan
原文链接：https://ld246.com/article/1587616996604
来源网站：链滴

Problem Description

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。

e.g.

1. MinStack minStack = new MinStack();
2. minStack.push(-2);
3. minStack.push(0);
4. minStack.push(-3);
5. minStack.getMin(); ==> 返回 -3.
6. minStack.pop();
7. minStack.top(); ==> 返回 0.
8. minStack.getMin(); ==>返回 -2.

Solution

该题很简单，但是有意思的是，我的解法也算通过了，虽然题目说

在常数时间内检索到最小元素的栈。

但是这里应该指的是 getMin()这个方法，所以即使我的 push(Object obj)方法复杂度比较高也没关。。

因为有删除元素（pop方法）的存在，所以最小值在push完之后还是会随时变。所以单纯使用一个变来保存最小值是不可能的。

我使用了双栈来实现，实际不用双栈用其他的数据结构也可以。大概原理就是在push的过程中，一个准stack正常push，另一个辅助栈从栈底到栈顶从小到大排序的插入，利用了Stack的父类（Vector方法 insertElementAt(Object obj, int i)，这里注意index是第二个参数。pop时，标准stack正常po，辅助stack只需要remove标准stack刚刚pop返回的对象即可 minStack.remove(stack.pop());。这确保两栈移除的元素是相同的。

代码如下：

```
public class MinStack {

    private Stack<Integer> stack;
    private Stack<Integer> minStack;
    /**
     * 执行用时 : 6 ms , 在所有 Java 提交中击败了 97.91% 的用户
     * 内存消耗 : 41.7 MB , 在所有 Java 提交中击败了 14.46% 的用户
     */
    public MinStack() {
        stack = new Stack<>();
        minStack = new Stack<>();
    }

    public void push(int x) {
        stack.push(x);
        if (minStack.empty()) {
            minStack.push(x);
        } else if (x >= minStack.peek()) {
            minStack.push(x);
        } else {
            for (int i = 0; i <= minStack.size(); i++) {
                if (x < minStack.get(i)) {
                    minStack.insertElementAt(x, i);
                    break;
                }
            }
        }

    }

    public void pop() {
        minStack.remove(stack.pop());
    }

    public int top() {
        return stack.peek();
    }

    public int getMin() {
        return minStack.firstElement();
```

```
    }

    @Override
    public String toString() {
        return "MinStack{" +
                "stack=" + stack +
                ", minStack=" + minStack +
                '}';
    }

    public static void main(String[] args) {

        MinStack minStack = new MinStack();
        minStack.push(-2);
        minStack.push(0);
        minStack.push(-3);
        minStack.push(-11);
        minStack.push(-1);
        System.out.println(minStack.toString());
        System.out.println(minStack.getMin());
        minStack.pop();
        minStack.top();
        System.out.println(minStack.getMin());

    }

}
```

Feature

虽然我这样做是通过的，但其实我想的不够多，看了高赞的题解，即使是利用辅助栈push方法时间复度也可以不到线性级 O(n)。

因为题目是说了设计的该栈对于移除元素只有pop方法，就是从栈顶移除，也就是说next元素比当前部元素大的话，就没必要保留了，因为在只有pop方法的情况下，该元素一定是先被移除的。所以在件下辅助栈可以不需要把所有元素保存。

除非题目加一个删除中间某个元素的方法的条件。

```
入栈 3
|   |     |   |
|   |     |   |
|_3_|     |_3_|
stack   minStack

入栈 5 ，  5 大于 minStack 栈顶，不处理
|   |     |   |
| 5 |     |   |
|_3_|     |_3_|
stack   minStack

入栈 2 ，此时右边的 minStack 栈顶就保存了当前最小值 2
| 2 |     |   |
| 5 |     | 2 |
|_3_|     |_3_|
stack   minStack

出栈 2 ，此时右边的 minStack 栈顶就保存了当前最小值 3
|   |     |   |
| 5 |     |   |
|_3_|     |_3_|
stack   minStack

出栈 5 ，右边 minStack 不处理
|   |     |   |
|   |     |   |
|_3_|     |_3_|
stack   minStack

出栈 3
|   |     |   |
|   |     |   |
|_ _|     |_ _|
stack   minStack
```

当然还有不用辅助栈的做法，利用压栈。