Min Stack

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

- push(x) -- Push element x onto stack.
- pop() -- Removes the element on top of the stack.
- top() -- Get the top element.
- getMin() -- Retrieve the minimum element in the stack.

Solution 1

The question is ask to construct One stack. So I am using one stack.

The idea is to store the gap between the min value and the current value;

The problem for my solution is the cast. I have no idea to avoid the cast. Since the possible gap between the current value and the min value could be Integer.MAX*VALUE-Integer.MIN*VALUE;

```
public class MinStack {
    long min;
    Stack<Long> stack;

    public MinStack(){
        stack=new Stack<>();
    }

    public void push(int x) {
        if (stack.isEmpty()){
            stack.push(0L);
            min=x;
        }else{
            stack.push(x-min);//Could be negative if min value needs to change
            if (x<min) min=x;
        }
    }

    public void pop() {
        if (stack.isEmpty()) return;

        long pop=stack.pop();

        if (pop<0)  min=min-pop;//If negative, increase the min value

    }

    public int top() {
        long top=stack.peek();
        if (top>0){
            return (int)(top+min);
        }else{
           return (int)(min);
        }
    }

    public int getMin() {
        return (int)min;
    }
}
```

written by reeclapple original link here

Solution 2

```
class MinStack {
    int min=Integer.MAX_VALUE;
    Stack<Integer> stack = new Stack<Integer>();
    public void push(int x) {
       // only push the old minimum value when the current
       // minimum value changes after pushing the new value x
        if(x <= min){
            stack.push(min);
            min=x;
        }
        stack.push(x);
    }

    public void pop() {
       // if pop operation could result in the changing of the current minimum va
lue,
       // pop twice and change the current minimum value to the last minimum valu
e.
        if(stack.peek()==min) {
            stack.pop();
            min=stack.peek();
            stack.pop();
        }else{
            stack.pop();
        }
        if(stack.empty()){
            min=Integer.MAX_VALUE;
        }
    }

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

    public int getMin() {
        return min;
    }
}
```

written by sometimescrazy original link here

Solution 3

```
class MinStack {
private:
    stack<int> s1;
    stack<int> s2;
public:
    void push(int x) {
        s1.push(x);
        if (s2.empty() || x <= getMin())  s2.push(x);
    }
    void pop() {
        if (s1.top() == getMin())  s2.pop();
        s1.pop();
    }
    int top() {
        return s1.top();
    }
    int getMin() {
        return s2.top();
    }
};
```

written by zjchenRice original link here