**难度: 简单**

定义栈的数据结构，请在该类型中实现一个能够得到栈的最小元素的 min 函数在该栈中，调用 min、push 及 pop 的时间复杂度都是 O(1)。

示例:

MinStack minStack = new MinStack();

minStack.push(-2);

minStack.push(0);

minStack.push(-3);

minStack.min(); --> 返回 -3.

minStack.pop();

minStack.top(); --> 返回 0.

minStack.min(); --> 返回 -2.

提示：

各函数的调用总次数不超过 20000 次

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/bao-han-minhan-shu-de-zhan-lcof

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**思路:**

1). 使用LinekdList 来模拟栈,再创建一个min变量

2). 每次调用push的时候先判断一下,最小值是否大于x,

1. 如果大于就先把一开始的最小值push进去,然后把x赋值给min

2). 最后把x的值push进行栈

3). 每次调用pop的时候,在pop的基础上,判断是否最小值,如果是在pop一次给min

4). 每次调用top,直接返回栈顶的值

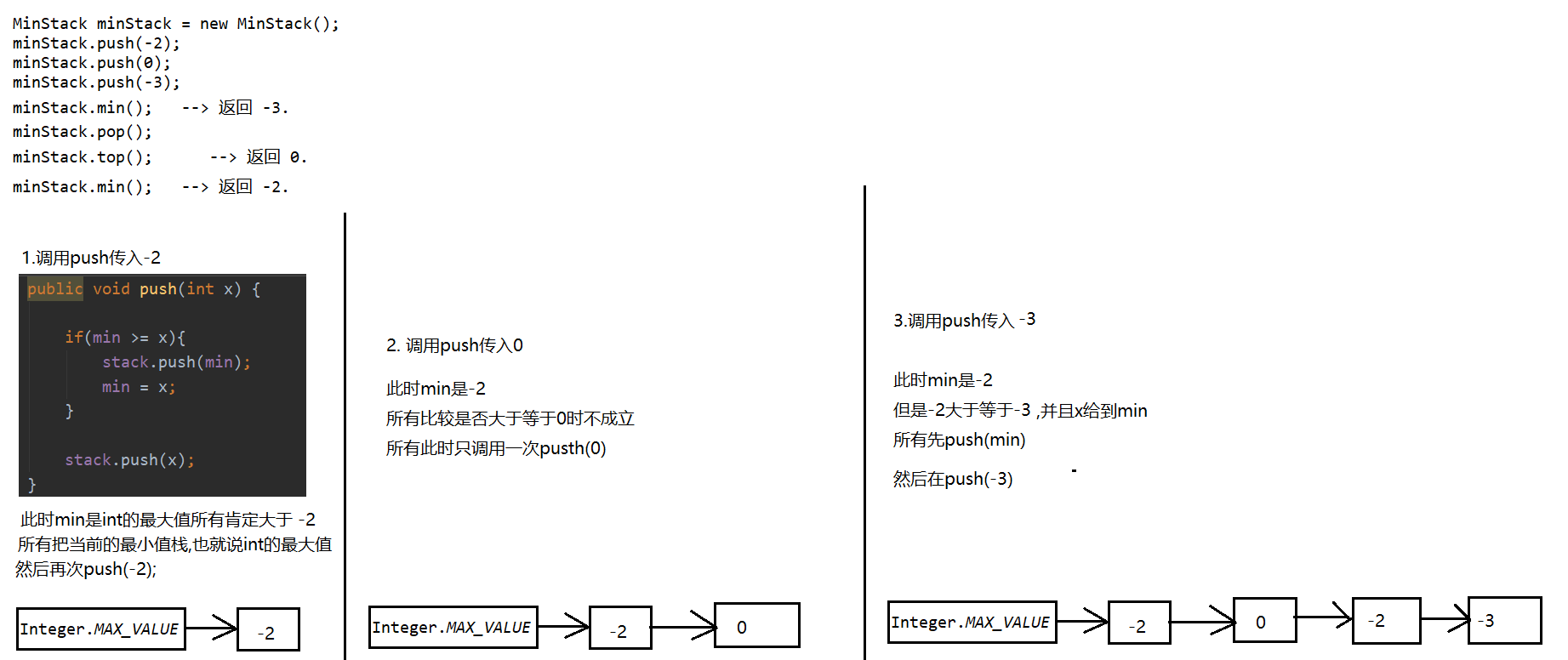
5). 每次调用min,直接返回min

**走一遍代码:**

minStack.push(-2);

minStack.push(0);

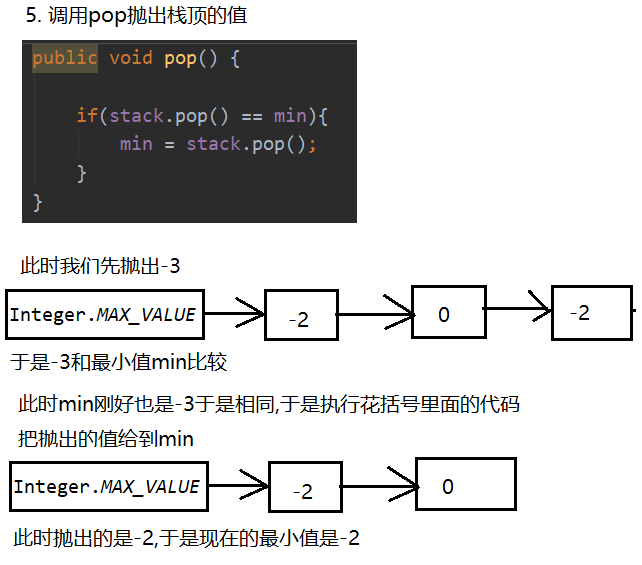
minStack.push(-3);

****

minStack.min(); --> 返回 -3.

此时min就是-3, 所有直接返回-3就好了

minStack.pop();



minStack.top(); --> 返回 0.

此时直接查看一下栈顶就行了

minStack.min(); --> 返回 -2.

再次看一下最小值

**代码:**

|  |
| --- |
| public class MinStack {  private LinkedList<Integer> stack;  private int min = Integer.MAX\_VALUE;  public MinStack() {  stack = new LinkedList<>();  }  public void push(int x) {  //先压先前最小值  //再压一个当前最小值，保证最小值一直存在  if(min >= x){  stack.push(min);  min = x;  }  stack.push(x);  }  public void pop() {  if(stack.pop() == min){  min = stack.pop();  }  }  public int top() {  return stack.peek();  }  public int min() {  return min;  }  } |