**难度: 简单**

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:

输入：

["MinStack","push","push","push","getMin","pop","top","getMin"]

[[],[-2],[0],[-3],[],[],[],[]]

输出：

[null,null,null,null,-3,null,0,-2]

解释：

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.

提示：

pop、top 和 getMin 操作总是在 非空栈 上调用。

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/min-stack

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:

1). 使用双栈(用Deque来创建)

2). 第一个栈用来放push的内容,第二个栈用来放每次第一个栈push之后,最小的值

3). 刚开始我们第二个栈先放一个最大的值(Integer,MAX),后面当我们第一个栈进行push的时候,就比较一下是否比栈顶的第一值小,如果小就放哪个小的值,如果大就放栈顶的值

|  |
| --- |
| class MinStack {  Deque<Integer> stack; //栈  Deque<Integer> stackMin; //用来存放最小栈  public MinStack() {  stack = new ArrayDeque<>();  stackMin = new ArrayDeque<>();  stackMin.push(Integer.MAX\_VALUE);  }  public void push(int x) {  stack.push(x);  if (stackMin.isEmpty() || x < stackMin.peek()) {  stackMin.push(x);  }else {  stackMin.push(stackMin.peek());  }  }  public void pop() {  stack.pop();  stackMin.pop();  }  public int top() {  return stack.isEmpty() ? -1 :stack.peek();  }  public int getMin() {  return stackMin.isEmpty() ? -1 : stackMin.peek();  }  } |

思路2(自己创建一个类):

1). 自己定义一个存储value以及min还有指向下一个节点的next

2). 每次除了存储value,还有比较一个当前的值是否小于上个的最小值,如果小于就存储当前的值,如果大于就存储上一个的最小值,然后指向下个节点

|  |
| --- |
| class MinStack2 {  private ListNode head;  public void push(int x) {  if (isEmpty()) {  head = new ListNode(x,x,null);  }else {  head = new ListNode(x,Math.min(x,head.min),head);  }  }  public void pop() {  head = head.next;  }  public int top() {  return head.val;  }  public int getMin() {  return head.min;  }  private boolean isEmpty() {  return head == null;  }  }  class ListNode {  public int val;  public int min;//最小值  public ListNode next;  public ListNode(int val, int min, ListNode next) {  this.val = val;  this.min = min;  this.next = next;  }  } |