**难度: 简单**

用两个栈实现一个队列。队列的声明如下，请实现它的两个函数 appendTail 和 deleteHead ，分别完成在队列尾部插入整数和在队列头部删除整数的功能。(若队列中没有元素，deleteHead 操作返回 -1 )

示例 1：

输入：

["CQueue","appendTail","deleteHead","deleteHead"]

[[],[3],[],[]]

输出：[null,null,3,-1]

示例 2：

输入：

["CQueue","deleteHead","appendTail","appendTail","deleteHead","deleteHead"]

[[],[],[5],[2],[],[]]

输出：[null,-1,null,null,5,2]

提示：

1 <= values <= 10000

最多会对 appendTail、deleteHead 进行 10000 次调用

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/yong-liang-ge-zhan-shi-xian-dui-lie-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(速度慢):**

1). 创建两个栈

2). 添加的时候给栈1添加

3). 删除的时候,判断栈1是否为空

1. 为空返回-1

2. 非空的话

1. 先循环把栈1的所有全部pop给栈2

2. 抛出栈2中的一个给到value

3. 把栈2的所有给到栈1

4. return value

**说明**: 由于Stakc继承了Vector接口, Vector底层是一个Object[]数组, 那么就要考虑空间扩容和移位的问题了,所有会比较慢,并且在删除的时候,使用了两次while循环

|  |
| --- |
| public class CQueue {  private Stack<Integer> stackOne;  private Stack<Integer> stackTwo;  public CQueue() {  stackOne = new Stack<>();  stackTwo = new Stack<>();  }  public void appendTail(int value) {  //入栈  stackOne.push(value);  }  public int deleteHead() {  //判断栈2是否为空  if (!stackTwo.isEmpty()) {  //不为空  return stackTwo.pop();  }  //判断栈1是否为空  if (stackOne.isEmpty()) {  //为空  return -1;  }    //循环判断栈1是否为空  while(!stackOne.isEmpty()) {  //不为空,栈1的值抛出给栈2  stackTwo.push(stackOne.pop());  }    //抛出  return stackTwo.pop();  }  } |

**思路2(相对快):**

1). 使用了LinkedList它具有,栈有的所有功能

2). 其它基本没有什么区别

|  |
| --- |
| public class CQueue {  private LinkedList<Integer> stackOne;  private LinkedList<Integer> stackTwo;  public CQueue() {  stackOne = new LinkedList<>();  stackTwo = new LinkedList<>();  }  public void appendTail(int value) {  //入栈  stackOne.push(value);  }  public int deleteHead()  {  if (!stackTwo.isEmpty()){  return stackTwo.removeLast();  }  if (stackOne.isEmpty()) {  return -1;  }  while(!stackOne.isEmpty()) {  stackTwo.addLast(stackOne.pop());  }  return stackTwo.removeLast();  }  } |