**难度: 中等**

请定义一个队列并实现函数 max\_value 得到队列里的最大值，要求函数max\_value、push\_back 和 pop\_front 的均摊时间复杂度都是O(1)。

若队列为空，pop\_front 和 max\_value 需要返回 -1

示例 1：

输入:

["MaxQueue","push\_back","push\_back","max\_value","pop\_front","max\_value"]

[[],[1],[2],[],[],[]]

输出: [null,null,null,2,1,2]

示例 2：

输入:

["MaxQueue","pop\_front","max\_value"]

[[],[],[]]

输出: [null,-1,-1]

限制：

1 <= push\_back,pop\_front,max\_value的总操作数 <= 10000

1 <= value <= 10^5

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/dui-lie-de-zui-da-zhi-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). 由于限制我们知道了总操作是10000,所有直接创建一个数组为10000的数组就行了

2). 可以用结束-开始是否等于0,如果等于0则表示为空,直接返回-1,不为空就遍历当前的数组获取最大值,因为是数组,查询速度快!!!

3). 添加的是否,直接让数组添加值,然后下标加一即可

4). 移除的时候,同样要进行判断是否为空,空返回-1,不为空,返回数组的第一个下标然后加一

5). 这个的缺点就是不能循环使用!

|  |
| --- |
| public class MaxQueue {  private int[] queue; //模拟栈  private int start = 0; //开始  private int end = 0; //结束  public MaxQueue() {  queue = new int[10000];  }  public int max\_value() {  if (end - start == 0) {  return -1;  }  int max = Integer.MIN\_VALUE;  for (int i = start; i < end; i++) {  max = Math.max(max,queue[i]);  }  return max;  }  public void push\_back(int value) {  queue[end++] = value;  }  public int pop\_front() {  if (end - start == 0) {  return -1;  }  return queue[start++];  }  } |