剑指 Offer 30 包含min函数的栈

Label：栈

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

- 添加 list

```
class MaxQueue {

    Deque<Integer> deque;
    List<Integer> list;

    public MaxQueue() {
        deque = new ArrayDeque<>();
        list = new ArrayList<>();
    }

    public int max_value() {
        if (deque.size() == 0)
            return -1;

        Collections.sort(list);
        return list.get(list.size()-1);
    }

    public void push_back(int value) {
        list.add(value);
        deque.offerLast(value);
    }

    public int pop_front() {
        if (deque.size() == 0)
            return -1;

        int a = deque.pollFirst();
        list.remove(list.indexOf(a));

        return a;
    }
}
```

- 辅助单调栈

```
class MaxQueue {
    private Deque<Integer> queue;
    private Deque<Integer> help;

    public MaxQueue() {
        queue = new ArrayDeque<>();  // 保存队列
        help = new ArrayDeque<>();   // 保存max，如果queue中它的后面有比他更大的数，
help就不用存储他
    }

    public int max_value() {
        return queue.isEmpty() ? -1 : help.peek();  // 因为有可能出现queue为空，max
还有一个值的情况
    }

    public void push_back(int value) {
        queue.offer(value);
        while(!help.isEmpty() && value > help.peekLast()) {
            help.pollLast();
        }
        help.offer(value);
    }

    public int pop_front() {
        if(queue.isEmpty())
            return -1;

        int val = queue.pop();
        if(help.peek() == val) {  // 无所谓help中的数目
            help.pop();
        }
        return val;
    }
}
```