队列的最大值

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]

```
class MaxQueue {
    queue<int> q;
    deque<int> d;//单调递减队列 维护最大值，始终保持队列头部的值最大  单调递减
public:
    MaxQueue() {
    }

    int max_value() {
        if (d.empty())
            return -1;
        return d.front();//单调递减队列的头部即为最大值
    }

    void push_back(int value) {
        while (!d.empty() && d.back() < value) {
            d.pop_back();
        }//单调递减队列的插入需要进行维护最大值 新插入队列的值需要与之前的比较，直到遇到的值比自己大  否则小值都需要从尾部弹出，然后最大值插入
        d.push_back(value);//有可能前面值都小  新值可以直接插入，即为当前最大
        q.push(value);//普通队列的尾部插入
    }

    int pop_front() {
        if (q.empty())
            return -1;
        int ans = q.front();
        if (ans == d.front()) {//当 q 队列头部元素与单调队列头部元素相同时 一块弹出  否则只将 q 队列头部弹出
        //这样做的目的是以免当前最大值丢失
            d.pop_front();
```

```
        }
        q.pop();
        return ans;
    }
};

/**
 * Your MaxQueue object will be instantiated and called as such:
 * MaxQueue* obj = new MaxQueue();
 * int param_1 = obj->max_value();
 * obj->push_back(value);
 * int param_3 = obj->pop_front();
 */
```