# 题目

设计一个栈，支持基本的栈操作，这个栈的内部存储数据的结构为队列，队列的方法只能包括push、peek(front)、pop、size、empty等标准的队列方法。

使用队列实现栈的下列操作：

push(x) -- 元素 x 入栈

pop() -- 移除栈顶元素

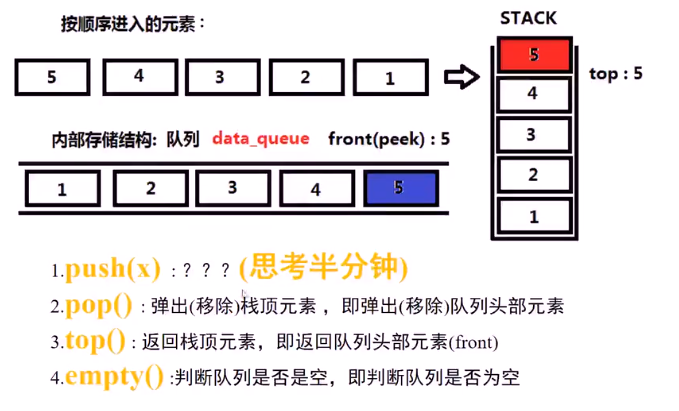
top() -- 获取栈顶元素

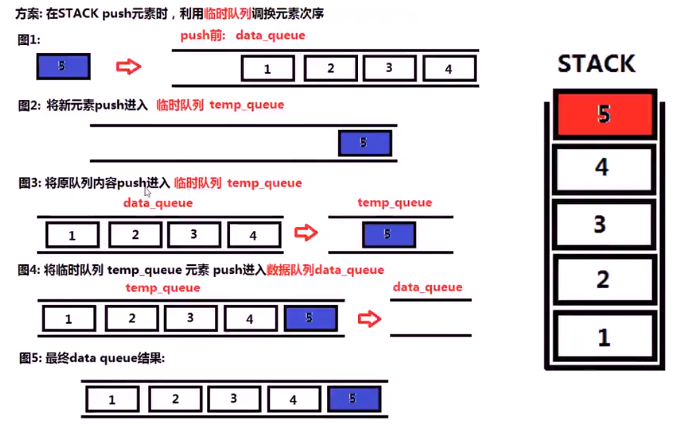
empty() -- 返回栈是否为空

注：Leetcode 225

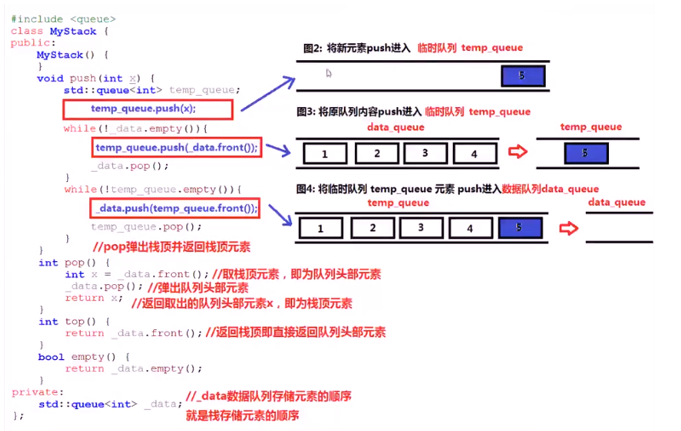
# **分析**

关键在于怎么实现栈的push操作，此时需要借助临时队列实现。





**代码：**



注：用队列实现栈，其实就是实现栈的top，push等操作。

class MyQueue {

private:

    std::queue<int> \_data;

public:

    /\*\* Initialize your data structure here. \*/

    MyQueue() {

    }

    /\*\* Push element x to the back of queue. \*/

    void push(int x) {

        std::queue<int> temp\_queue;

        temp\_queue.push(x);

*while(!\_data.empty()){ //note:采用empty()循环,不要采用迭代器*

            temp\_queue.push(\_data.front());

             \_data.pop();

        }

*//note:不能采用在这里\_data.clear()清空队列，因为循环中会使用\_data*

        while(!temp\_queue.empty()){

            \_data.push(temp\_queue.front());

            temp\_queue.pop();

        }

    }

    /\*\* Removes the element from in front of queue and returns that element. \*/

    int pop() {

        //return \_data.front();

//note:pop()需要修改栈的元素，不能采用这样（错误答案）

*//note:正确的pop()操作*

*int top = \_data.front();*

*\_data.pop();*

*return top;*

    }

    /\*\* Get the front element. \*/

    int top() {

        return \_data.front();

    }

    /\*\* Returns whether the queue is empty. \*/

    bool empty() {

        return \_data.empty();

    }

};

**测试：**

