LeetCode #225 用队列实现栈

作者：matthewhan
原文链接：https://ld246.com/article/1588129687539
来源网站：链滴

Problem Description

- 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

node

- 你只能使用队列的基本操作-- 也就是 push to back, peek/pop from front, size, 和 is empty 这操作是合法的。
- 你所使用的语言也许不支持队列。 你可以使用 list 或者 deque（双端队列）来模拟一个队列 , 只要标准的队列操作即可。
- 你可以假设所有操作都是有效的（例如, 对一个空的栈不会调用 pop 或者 top 操作）。

Solution

双向队列Deque实现

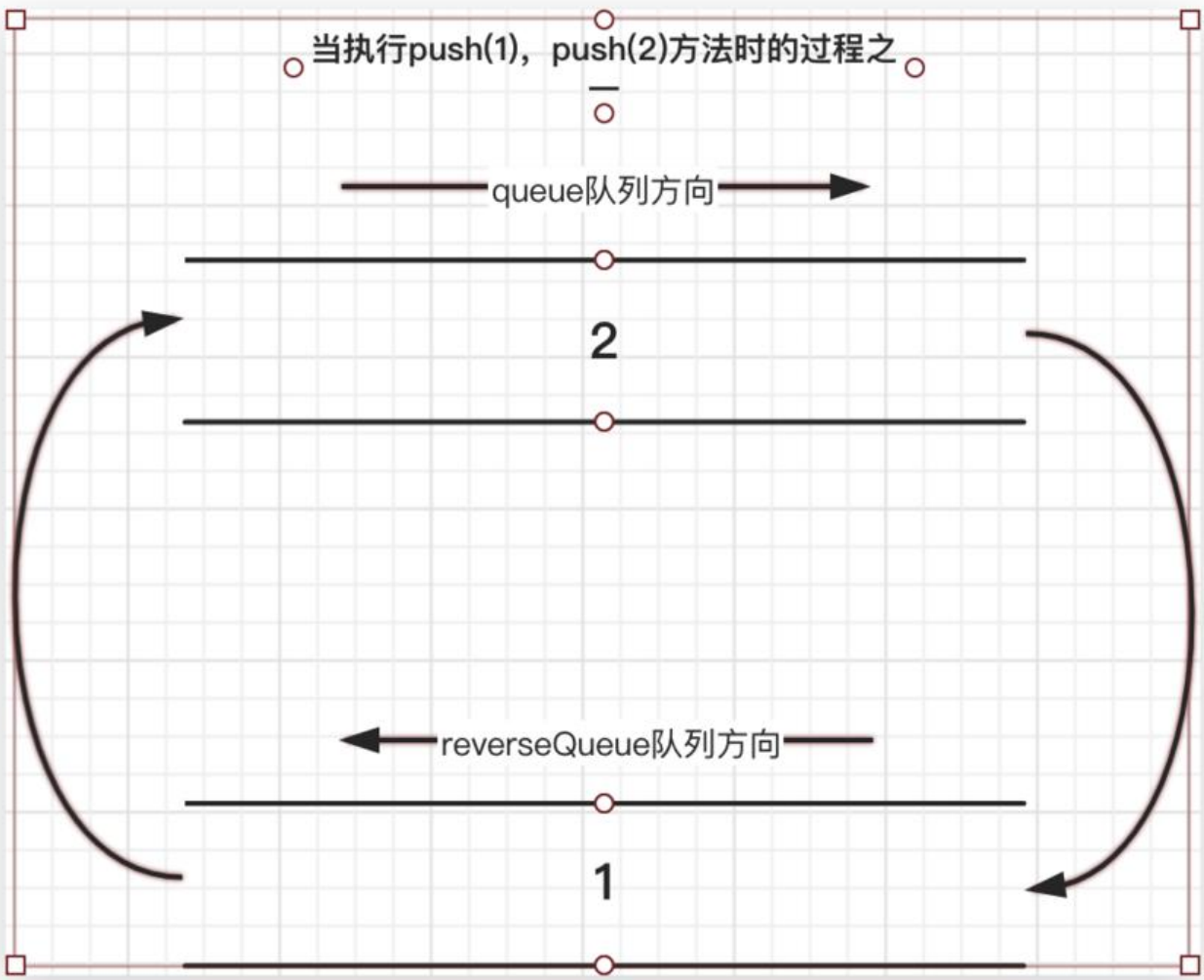
利用双端队列Deque接口的实现类LinkedList可以很简洁的完成，Deque拥有一个 boolean offerFirst E e);方法。因为底层是双向链表，prev和next指针可以很好帮你实现头尾乱♂插。

```
class MyStack {
```

```
    Deque<Integer> deque;

    public MyStack() {
        deque = new LinkedList<>();
    }

    public void push(int x) {
        deque.offerFirst(x);
    }

    public int pop() {
        return deque.remove() ;
    }

    public int top() {
        return deque.element();
    }

    public boolean empty() {
        return deque.isEmpty();
    }

}
```

双队列形成一个闭环

1. 2个单向队列queue和reverseQueue；
2. queue初次push时，判断是否为空，为空直接调用offer方法；
3. 当queue不为空时，将所有的元素从头部出队，入队到reverseQueue。queue入队新的元素；
4. 这时两个队列，queue只有新的元素在队列中，reverseQueue拥有queue刚刚出队的元素；
5. 再次将reverseQueue的元素循环出队到queue，保证每次调用push方法时，该队列是空的；
6. 两个单项队列变成闭环。




```
class MyStack {

    Queue<Integer> queue;
    Queue<Integer> reverseQueue;

    public MyStack() {
        queue = new LinkedList<>();
        reverseQueue = new LinkedList<>();
    }

    public void push(int x) {
        /**
         * 利用两个单向队列
         */
        if (queue.isEmpty()) {
            queue.offer(x);
        } else {
            reverseQueue.addAll(queue);
            queue.clear();
            queue.offer(x);
            queue.addAll(reverseQueue);
            reverseQueue.clear();
```

原文链接：LeetCode #225 用队列实现栈

```
        }
    }

    public int pop() {
        return queue.remove();
    }

    public int top() {
        return queue.element();
    }

    public boolean empty() {
        return queue.isEmpty();
    }
}
```

单个单向队列

其中注意：不要使用自身的forEach方法，根据队列的长度搞个循环即可。

push方法核心代码：

```
/*
 * 单个单向队列
 */
public void push(int x) {
    singleQueue.add(x);
    for (int i = 0; i < singleQueue.size() - 1; i++) {
        singleQueue.add(singleQueue.poll());
    }
}
```

注意是 size - 1。因为最新add的那个元素不需要执行 poll方法。