Using stack Implement queue

232. Implement Queue using Stacks

Easy 6250 362 Add to List Share

Implement a first in first out (FIFO) queue using only two stacks. The implemented queue should support all the functions of a normal queue (`push` , `peek` , `pop` , and `empty`).

Implement the `MyQueue` class:

- `void push(int x)` Pushes element x to the back of the queue.
- `int pop()` Removes the element from the front of the queue and returns it.
- `int peek()` Returns the element at the front of the queue.
- `boolean empty()` Returns `true` if the queue is empty, `false` otherwise.

Notes:

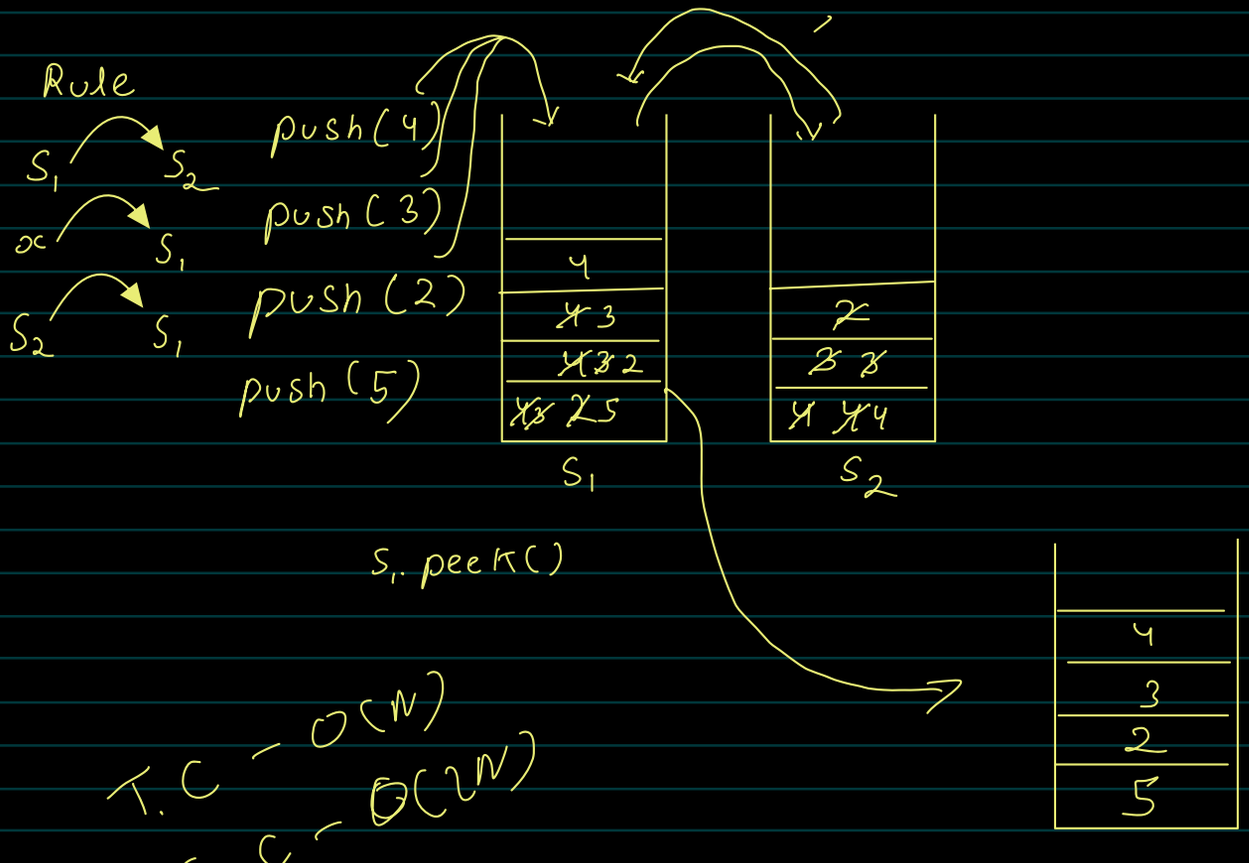
- You must use **only** standard operations of a stack, which means only `push to top` , `peek/pop from top` , `size` , and `is empty` operations are valid.
- Depending on your language, the stack may not be supported natively. You may simulate a stack using a list or deque (double-ended queue) as long as you use only a stack's standard operations.

To design a queue we will need 2 stack

→ for insertion

→ for deletion

example push(4, 3, 2, 5), then top must be 4, pop = 4

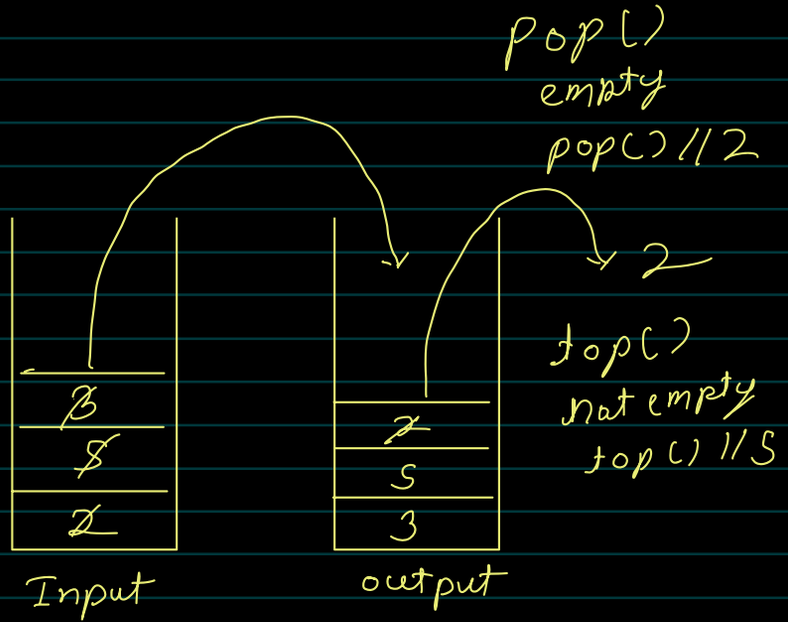


T.C - O(N)

S.C - O(2N)

Approch : 2

push(2)
push(5)
push(3)



Rules

push(x) ⇒ add x → input

pop() if (output is not empy)
output.pop()

else
input → output
output.pop()

top() if (output not empty)
return output.top()

else
input → output
output.top()

Code

```
class MyQueue {
public:
    void push(int x) {
        push_stk.push(x);
    }

    int pop() {
        peek();
        int val = pop_stk.top();
        pop_stk.pop();
        return val;

    }

    int peek() {
        if (pop_stk.empty())
        while (!push_stk.empty())
            pop_stk.push(push_stk.top()), push_stk.pop();
        return pop_stk.top();
    }

    bool empty() {
        return push_stk.empty() && pop_stk.empty();
    }
private:
    stack<int> push_stk;
    stack<int> pop_stk;
};

/**
```