

Course Algorithms and Data Structures · Lecture 7

Stacks and Queues

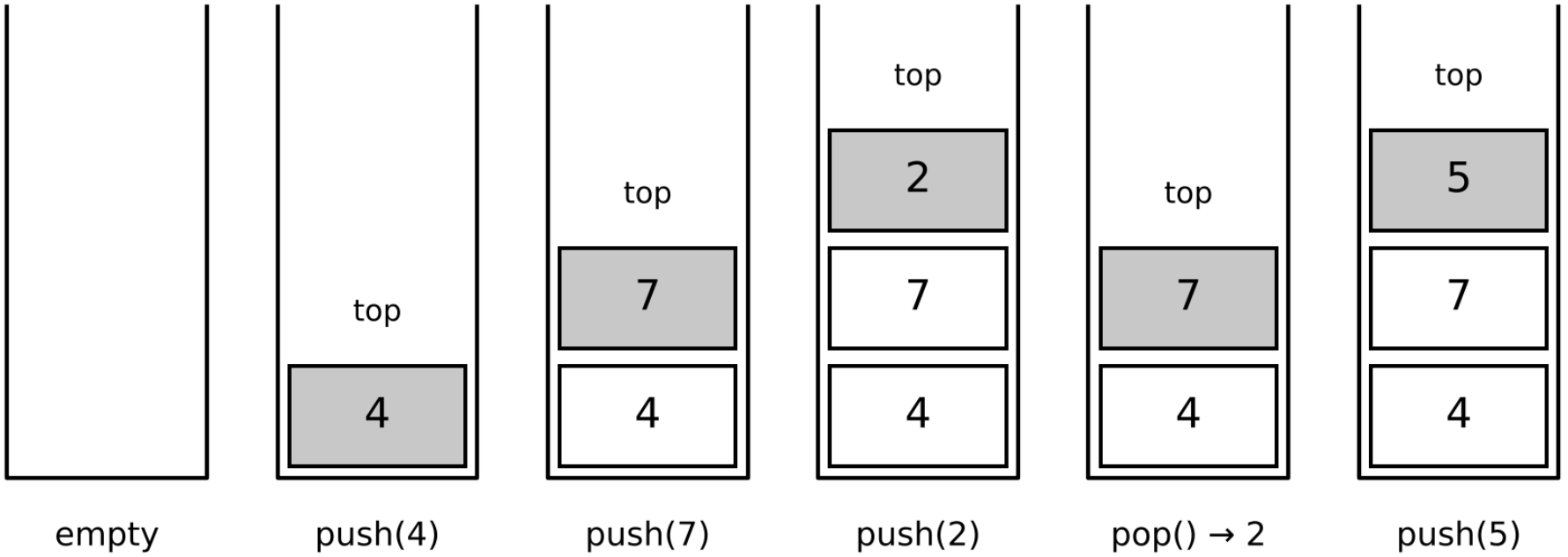
Textbook: A. Levitin, Section 1.4

Stack: last in, first out (LIFO)

Like a pile of plates: put on top, take from the top.

Operation	What it does
push(x)	put x on top
pop()	remove the top element and return it
peek()	look at the top element
isEmpty()	is the stack empty?

A stack in action



The grey element is the top.

Stack on an array

```
ALGORITHM Push(x)              ALGORITHM Pop()
    if top = N − 1                 if top = −1
        error "overflow"               error "underflow"
    top ← top + 1                  x ← S[top]
    S[top] ← x                     top ← top − 1
                                   return x
```

S[0..N − 1], index **top** of the top element; empty stack: top = −1. No loops, no shifting: **O(1)**.

Stack on a linked list

```
ALGORITHM Push(x)            ALGORITHM Pop()
    q ← new node                 if top = null
    q.data ← x                   error "underflow"
    q.next ← top                 x ← top.data
    top ← q                      top ← top.next
                                 return x
```

The top is the head of the list. Push = insert at the front, pop = delete the front: **O(1)**, no size limit.

Where stacks are used

- **Function calls**: the call stack from Lecture 5
- **Undo** in editors, the **Back** button in browsers
- **Checking brackets** in programs and formulas
- **Depth-first search** in graphs (week 14)

Example: balanced brackets

```
ALGORITHM Balanced(s[0..n − 1])
    create an empty stack
    for i ← 0 to n − 1 do
        if s[i] is an opening bracket
            push(s[i])
        else if s[i] is a closing bracket
            if isEmpty() return false
            if pop() is not the pair of s[i] return false
    return isEmpty()
```

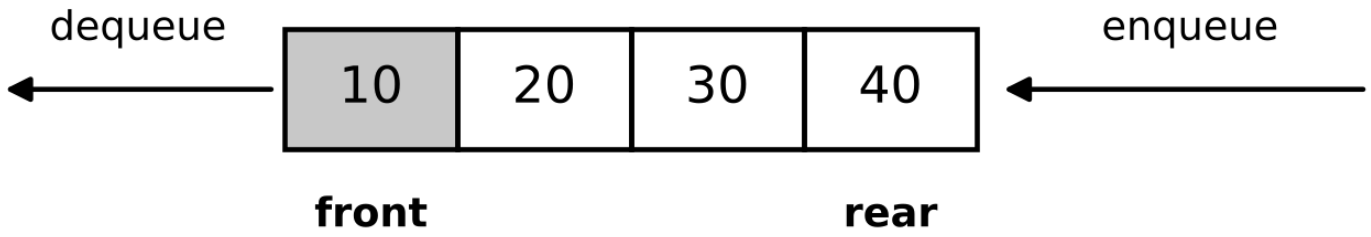
Tracing ([ ] { })

Symbol	Action	Stack (top on the right)
(	push	(
[	push	([
]	pop [, matches	(
{	push	({
}	pop {, matches	(
)	pop (, matches	empty

Empty at the end: balanced. For ( ] the pop gives (, which doesn’t match: false. O(n).

Queue: first in, first out (FIFO)

Like a line at a bank: join at the rear, leave from the front.



Operation	What it does
enqueue(x)	add x at the rear
dequeue()	remove the front element and return it

Queue on an array: the circular buffer

- Front always at Q[0]: every dequeue shifts all elements: **O(n)**, too slow
- Let **front** move instead, and keep a **count**
- After the last cell, continue from cell 0: **wrap around**

next cell after i = (i + 1) mod N

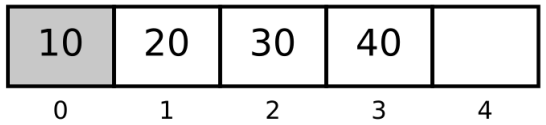
Enqueue and dequeue

```
ALGORITHM Enqueue(x)                ALGORITHM Dequeue()
    if count = N                        if count = 0
        error "full"                        error "empty"
    rear ← (front + count) mod N        x ← Q[front]
    Q[rear] ← x                         front ← (front + 1) mod N
    count ← count + 1                   count ← count − 1
                                        return x
```

No element is ever moved: both operations are O(1).

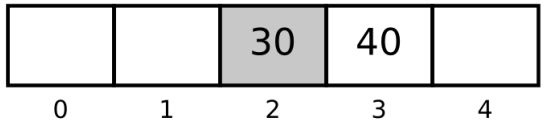
Tracing the circular queue, N = 5

after enqueue 10, 20, 30, 40



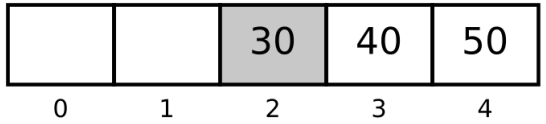
front = 0, count = 4, rear = 3

after two dequeues



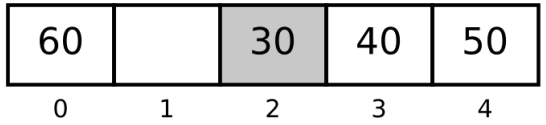
front = 2, count = 2, rear = 3

after enqueue 50



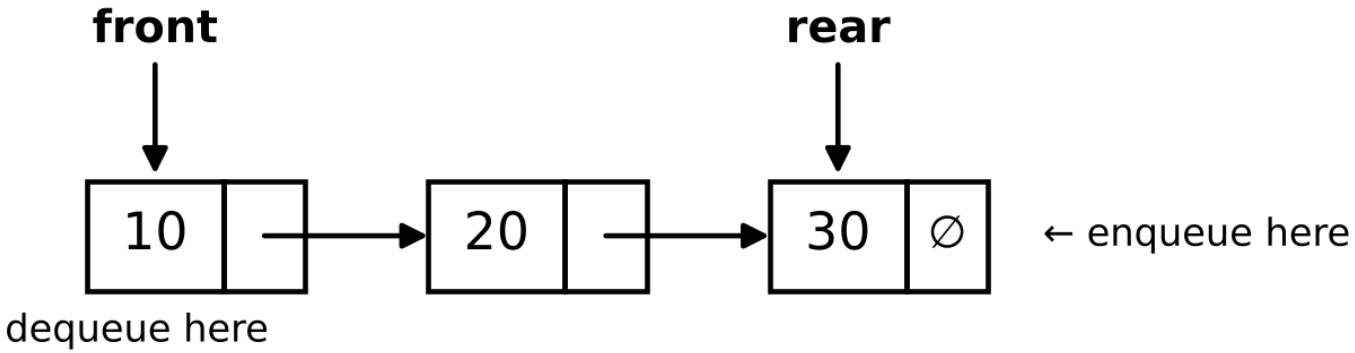
front = 2, count = 3, rear = 4

after enqueue 60: it goes to index 0



front = 2, count = 4, rear = 0

Queue on a linked list



- **Dequeue at the front**: front ← front.next
- **Enqueue at the rear**: rear.next ← q; rear ← q
- Both O(1). The other way round, removing the last node would cost O(n)
- Update **rear** when the queue becomes empty or gets its first element

Summary of costs

	Array	**Linked list**
push / pop (stack)	O(1)	O(1)
enqueue / dequeue (queue)	O(1), circular buffer	O(1), front and rear
Size limit	fixed N	only memory
Extra memory	none	a link per node

- Queues in practice: printers, servers, processes in the OS, **breadth-first search** (week 13)

Summary

- A stack is LIFO (push and pop at the top); a queue is FIFO (enqueue at the rear, dequeue at the front).
- Both can be built on an array or on a linked list, with all operations in O(1).
- A queue on an array needs a circular buffer with mod N.
- Stacks: calls, Undo, brackets. Queues: waiting lines, buffers, BFS.

Review questions

1. What do LIFO and FIFO mean? Give one real-life example of a stack and one of a queue.
2. Starting with an empty stack, do push(1), push(2), pop(), push(3), push(4), pop(). What is in the stack, and what is on top?
3. Why is a circular buffer used to implement a queue on an array?

Reading: Levitin, Section 1.4; Cormen et al., Section 10.1.