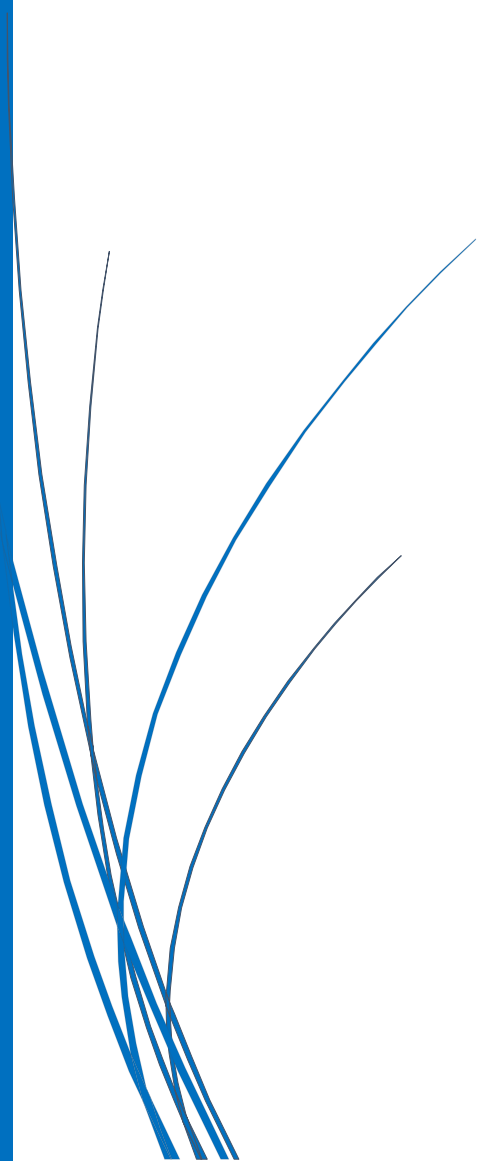
Made By:
Kenan Gazwan

Stacks & Queues

ICS202-Summary

King Fahd University of
Petroleum and Minerals



Telegram: @KenanGazwan

(Stacks & Queues)

✓ Stacks

A linear data structure that can be accessed only at one of its ends for storing and retrieving data.

It is called a **LIFO** data structure; Last In/First Out

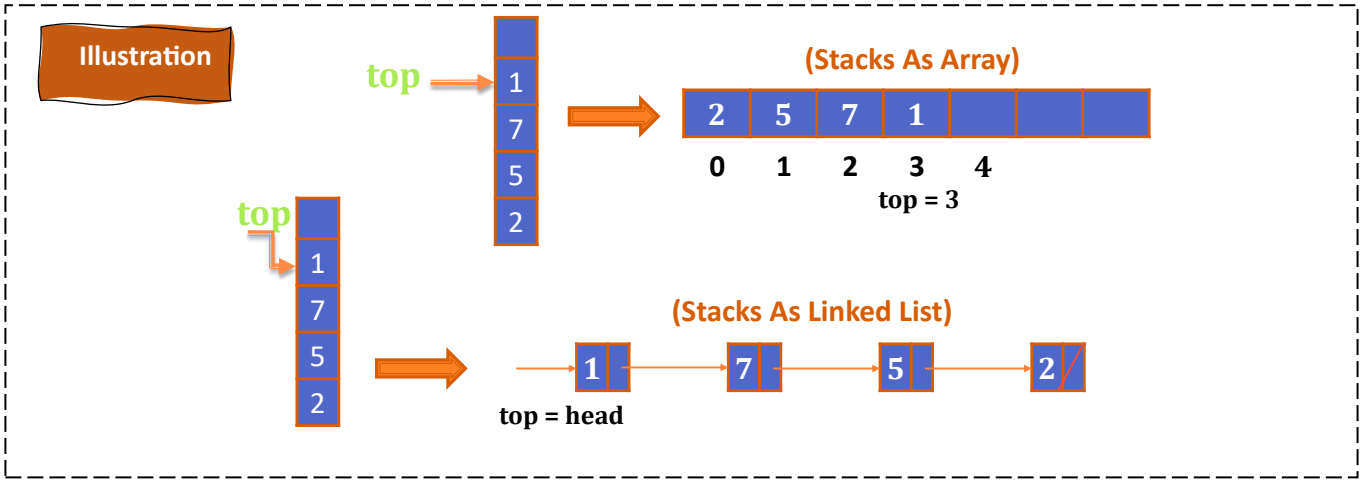
Stacks' Main Operations:

- **clear()**
- **isEmpty()**
- **push(el) –** put the element el on the top of the stack.
- **pop() –** delete and return the topmost element in the stack.
- **topEl() -** return the topmost element in the stack without removing it.

Note: top == -1 if the stack is empty**.**

Stack Implementations:

- **Stacks as Array**
- **Stacks as Linked List**



Stacks as Array:

- To reach the top = array[count – 1]
- To reach the bottom = array[0]
- The fields of the stack as an array:
  - T[] array = null
  - int capacity**;**
  - int top = -1;
  - int size = 0;

Stacks as Linked List:

- The fields of the stack as a Linked List:
  - Node top = null;

Applications of Stacks:

- Delimiter Matching
- Adding Large Integers
- Evaluating postfix Expressions

✓ Queues

A linear data structure in which we add elements to one end and remove them from the other end.
It is called **FIFO**; First In/First Out

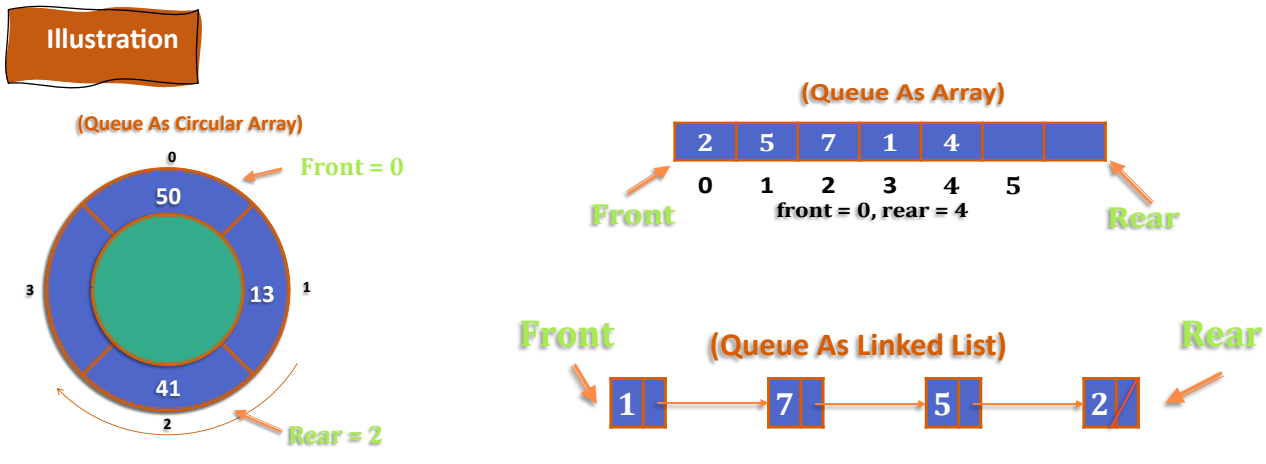
Queue Main Operations:

- **enqueue(el) –** insert an element at the rear of the queue. → $O(1)$
- **dequeue() –** return and delete an element from the front of the queue. → $O(n)/O(1)$
- **peek() -** return the element from the top without removing it. → $O(n)/O(1)$

Queue Implementations:

- **Queue as Array**
- **Queue as Circular Array**
- **Queue as Linked List**

Illustration



Queue as Array:

- Enqueue → $O(1)$
- Dequeue → $O(n)$ – due to shifting.
- Search → $O(n)$
- Peek → $O(1)$

Queue as Circular Array:

- Enqueue → $O(1)$
- Dequeue → $O(1)$
- Peek → $O(1)$
- Note: By using modulo (%) arithmetic for computing array indexes, all operations are $O(1)$

Queue as Linked List:

- Enqueue → addToTail() → $O(1)$
- Dequeue → removeFromHead() → $O(1)$
- Peek → $O(1)$

Priority Queue:

✓ A priority queue is a queue in which the dequeue operation removes an item from the front of the queue. The item removed will always have the highest priority.

✓ The items in a priority queue may or may not be sorted according to how they are inserted in the queue.

✓ Enqueue Complexity = O(n)

Operation	Push/Enqueue	Pop/Dequeue	TopEl/Peek
Stacks	$O(1)$	$O(1)$	$O(1)$
Stacks as Array	$O(1)$	$O(1)$	$O(1)$
Stacks as LinkedList	$O(1)$	$O(1)$	$O(1)$
Queue	$O(1)$	$O(1)/O(n)$	$O(1)$
Queue as Array	$O(1)$	$O(n)$	$O(1)$
Queue as Circular Array	$O(1)$	$O(1)$	$O(1)$
Queue as Linked List	$O(1)$	$O(1)$	$O(1)$
Priority Queue	$O(n)$	$O(n)$ with: Arrays $O(1)$ with: Other Data structures	$O(1)$

SLL/DLL	A dynamic data structure consisting of a sequence of nodes, forming a linear ordering.
CLL	A sequence of nodes in which every node has a link to its next node in the sequence and the last node has a link to the first node.
Stacks (LIFO)	A linear data structure that can be accessed only at one of its ends for storing and retrieving data.
Queues (FIFO)	A linear data structure in which we add elements to one end and remove them from the other end.