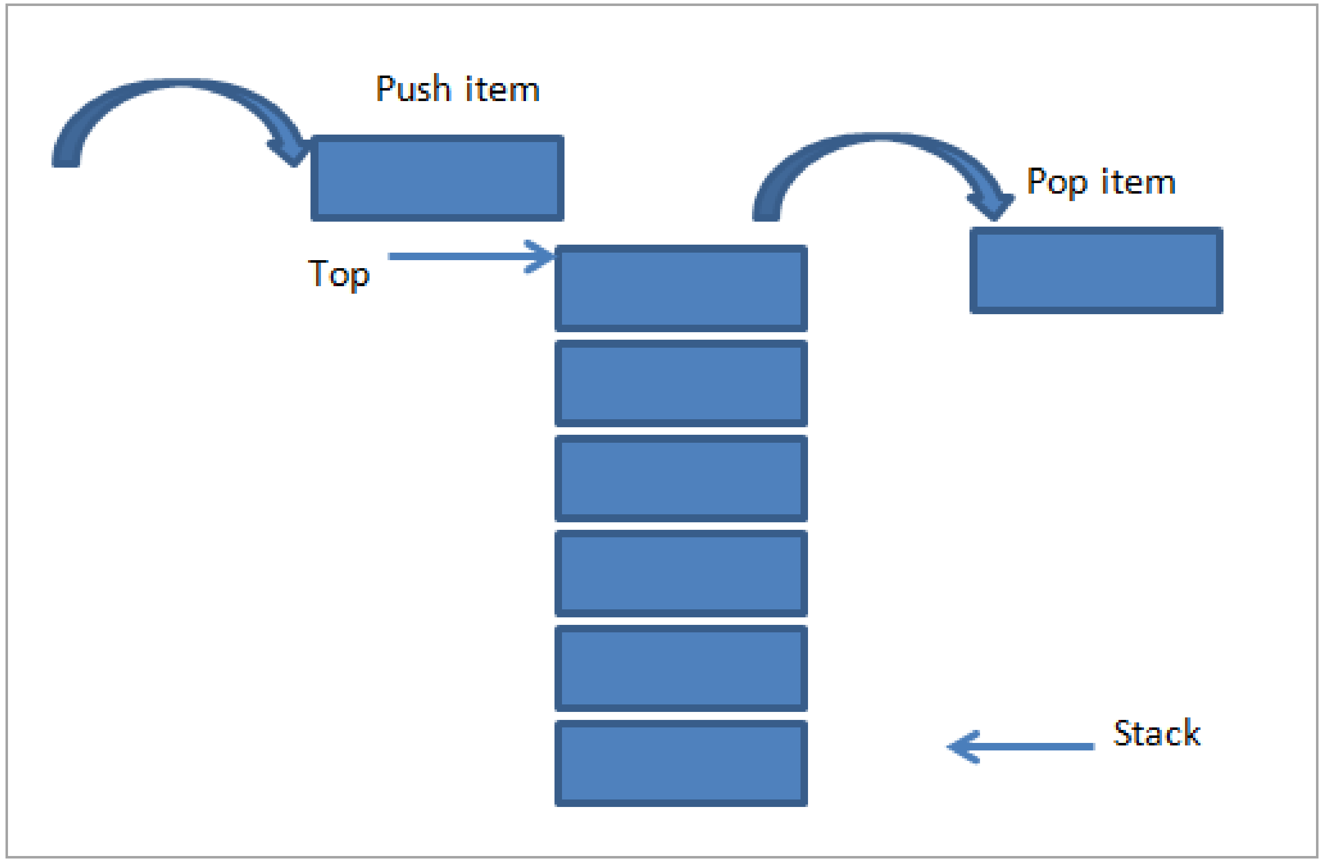
Stack & Queue

Shoaib Rauf

Stack

- Stack is a fundamental data structure which is used to store elements in a linear fashion.

- Stack follows LIFO (last in, first out) order or approach in which the operations are performed. This means that the element which was added last to the stack will be the first element to be removed from the stack.



Basic Operations

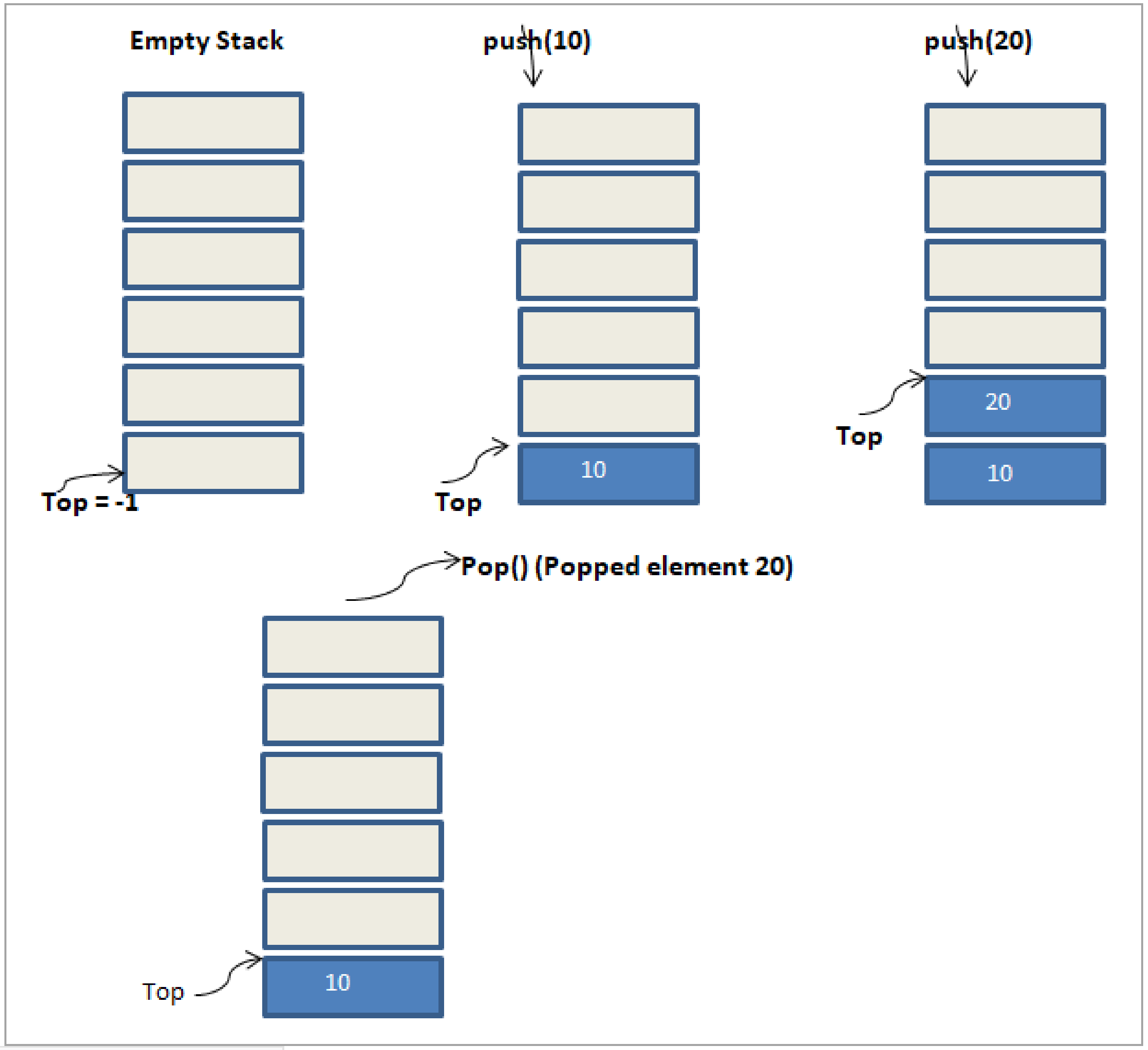
Following are the basic operations that are supported by the stack.

•**push –** Adds or pushes an element into the stack.

•**pop –** Removes or pops an element out of the stack.

•**isFull –** Tests if the stack is full.

•**isEmpty –** Tests if the stack is empty.


Empty Stack
push(10)
push(20)
Top = -1
10
Top
20
Top
10
Pop() (Popped element 20)
Top
10

Queue

- The queue is a basic data structure just like a stack. In contrast to stack that uses the LIFO approach, queue uses the FIFO (first in, first out) approach. With this approach, the first item that is added to the queue is the first item to be removed from the queue. Just like Stack, the queue is also a linear data structure.
- In a real-world analogy, we can imagine a bus queue where the passengers wait for the bus in a queue or a line. The first passenger in the line enters the bus first as that passenger happens to be the one who had come first.

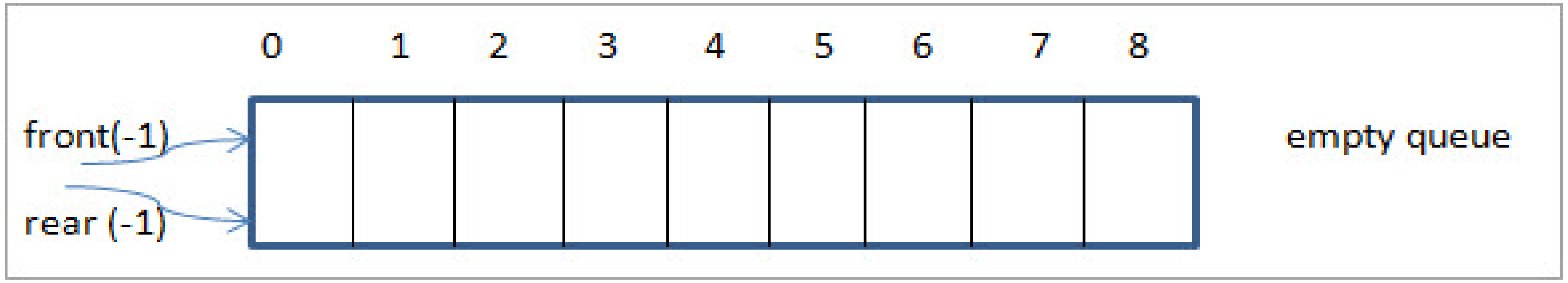
Basic Operations

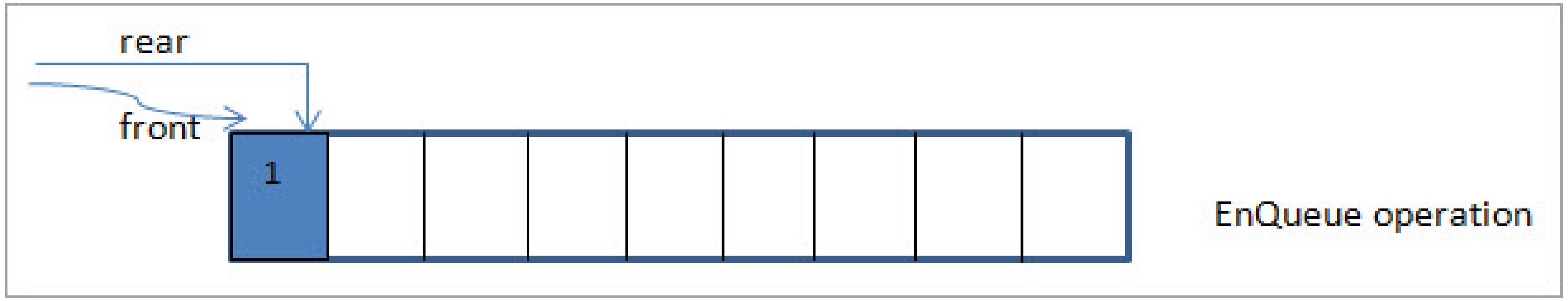
- **The queue data structure includes the following operations:**
- **EnQueue:** Adds an item to the queue. Addition of an item to the queue is always done at the rear of the queue.
- **DeQueue:** Removes an item from the queue. An item is removed or de-queued always from the front of the queue.
- **isEmpty:** Checks if the queue is empty.
- **isFull:** Checks if the queue is full.

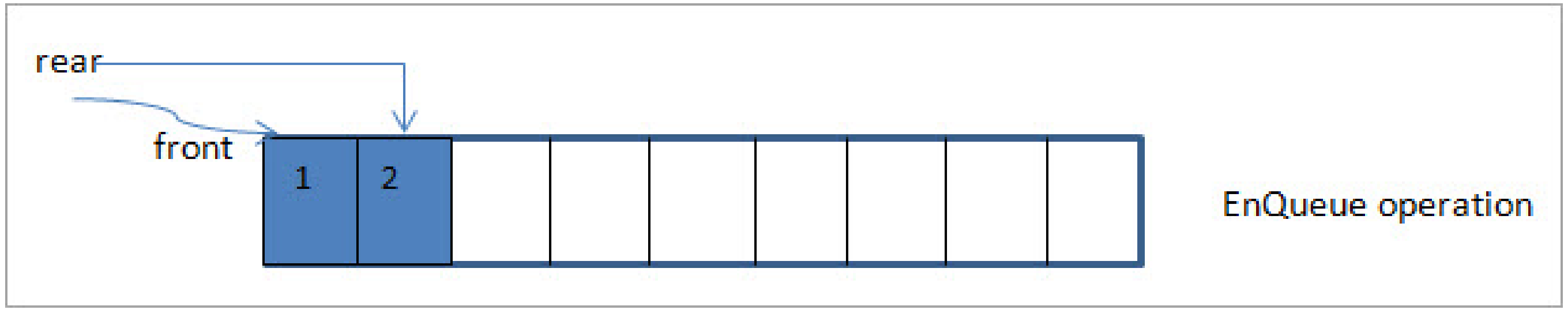
Enqueue:

- **In this process, the following steps are performed:**
- Check if the queue is full.
- If full, produce overflow error and exit.
- Else, increment ‘rear’.
- Add an element to the location pointed by ‘rear’.
- Return success.

Enqueue


0
1
2
3
4
5
6
7
8
front(-1)
rear (-1)
empty queue

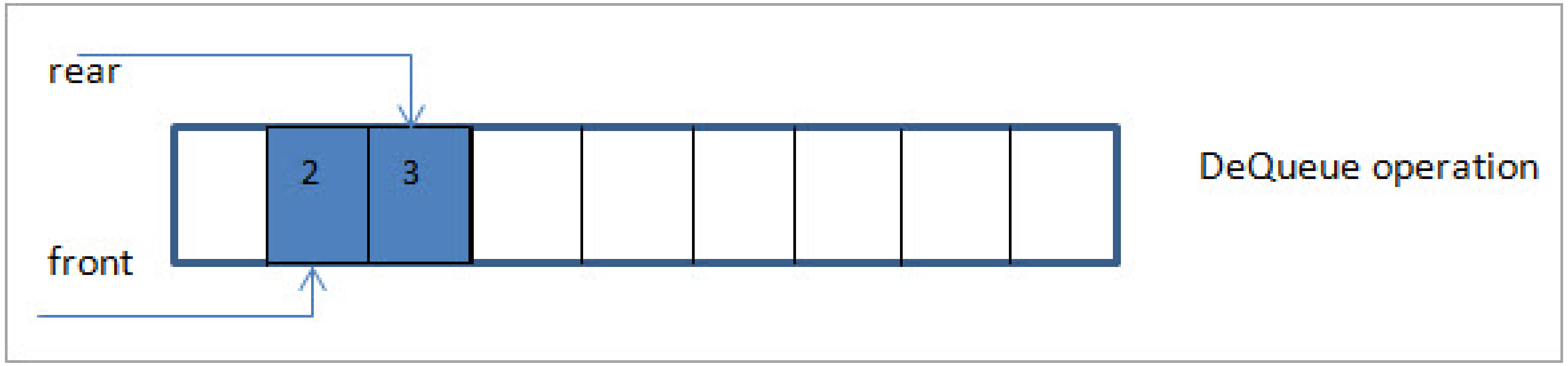

rear
front
1
EnQueue operation


rear
front
1
2
EnQueue operation

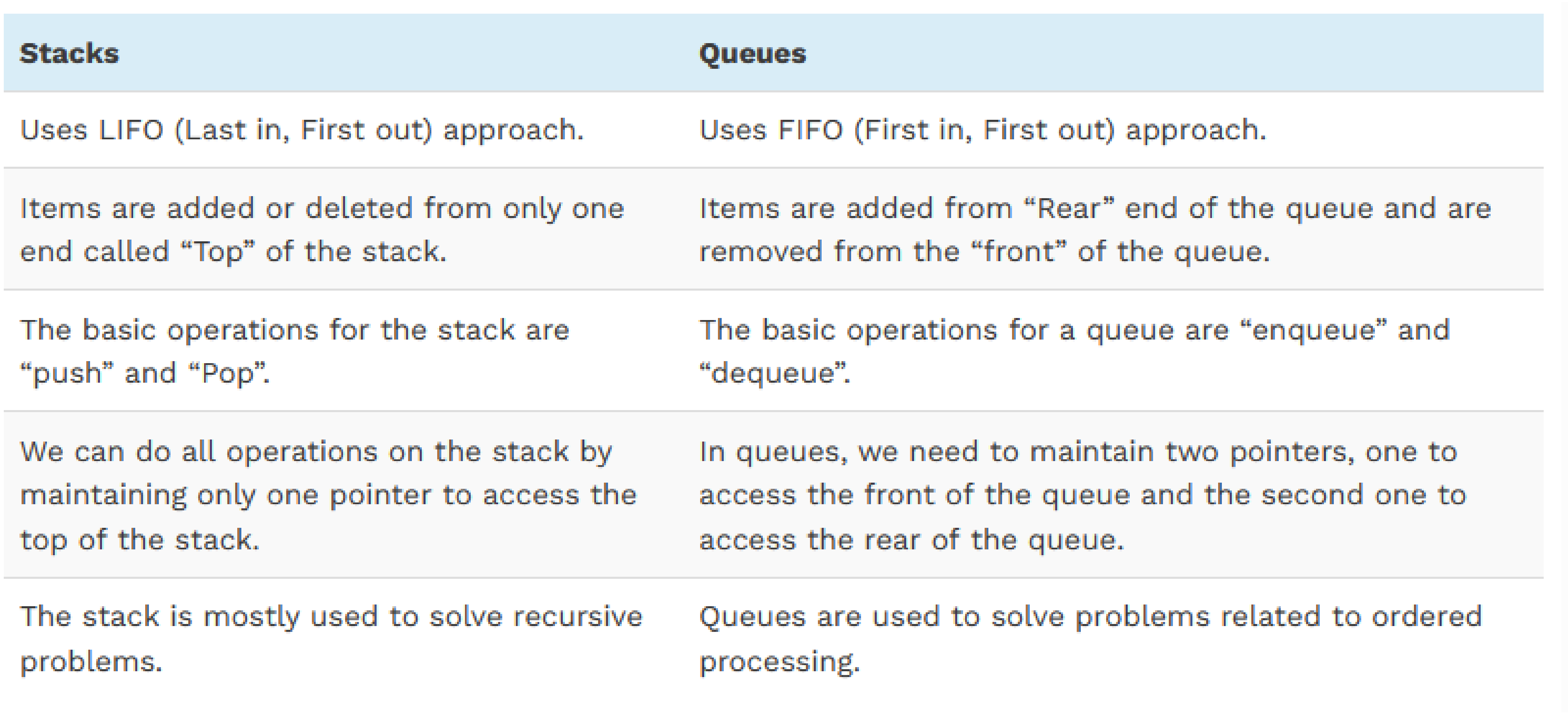
Dequeue:

- **Dequeue operation consists of the following steps:**
- Check if the queue is empty.
- If empty, display an underflow error and exit.
- Else, the access element is pointed out by ‘front’.
- Increment the ‘front’ to point to the next accessible data.
- Return success.

Dequeue



Stack Vs Queue



Stacks	Queues
Uses LIFO (Last in, First out) approach.	Uses FIFO (First in, First out) approach.
Items are added or deleted from only one end called “Top” of the stack.	Items are added from “Rear” end of the queue and are removed from the “front” of the queue.
The basic operations for the stack are “push” and “Pop”.	The basic operations for a queue are “enqueue” and “dequeue”.
We can do all operations on the stack by maintaining only one pointer to access the top of the stack.	In queues, we need to maintain two pointers, one to access the front of the queue and the second one to access the rear of the queue.
The stack is mostly used to solve recursive problems.	Queues are used to solve problems related to ordered processing.