Q K ⇒ Integer. Given

Print

Generate first (K) numbers using only digits 1, 2 & 3.

K = 10

Ans = 1, 2, 3, 11, 12, 13, 21, 22, 23, 31

Solⁿ

1 1 ⇒ 12, 13

1 2

1/2/3 1/2/3

1 2 K

2 1

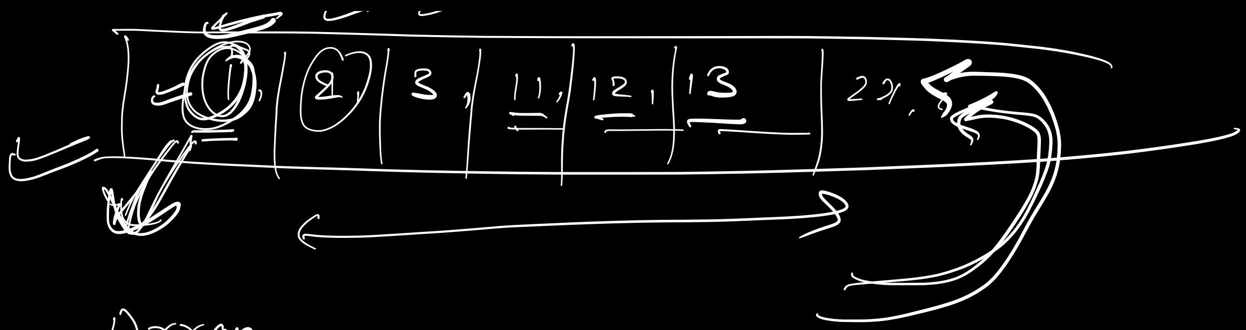
1, 2, 3

1, 2, 3, 11, 12, 13, 21, 22, 23, 31, 32, 33

1
2
3

111, 112, 113

←



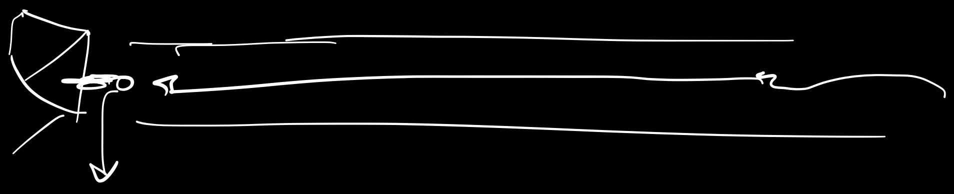
Array

Data Structure required

Queues

FIFO ⇒ first In first Out

⇒



Operations on Queue

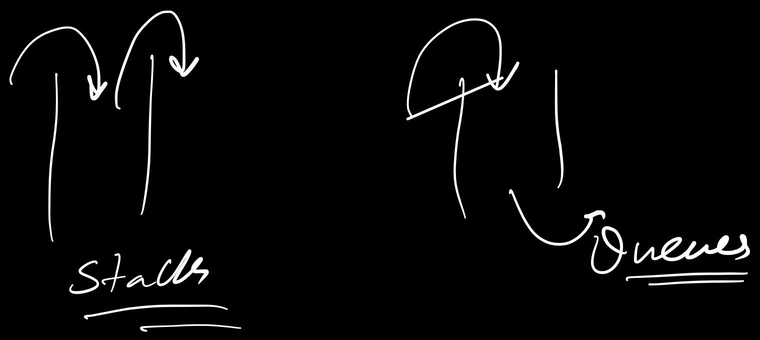
1) Enque() ⇒ Adding an element at the back of the queue ⇒ O(1)

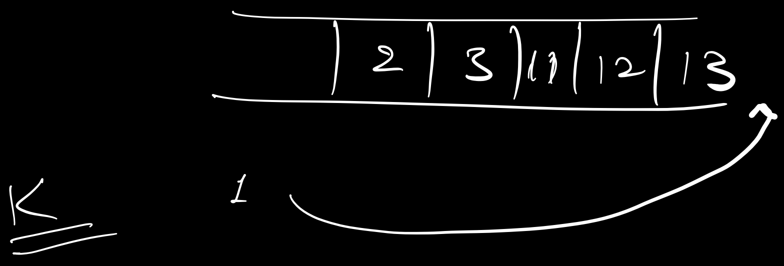
2) Dequeue() ⇒ Remove element from the front of the queue
⇒ O(1)

3) front() ⇒ Tell the first element

4) IsEmpty() ⇒ True if queue is Empty / else false.

⇒

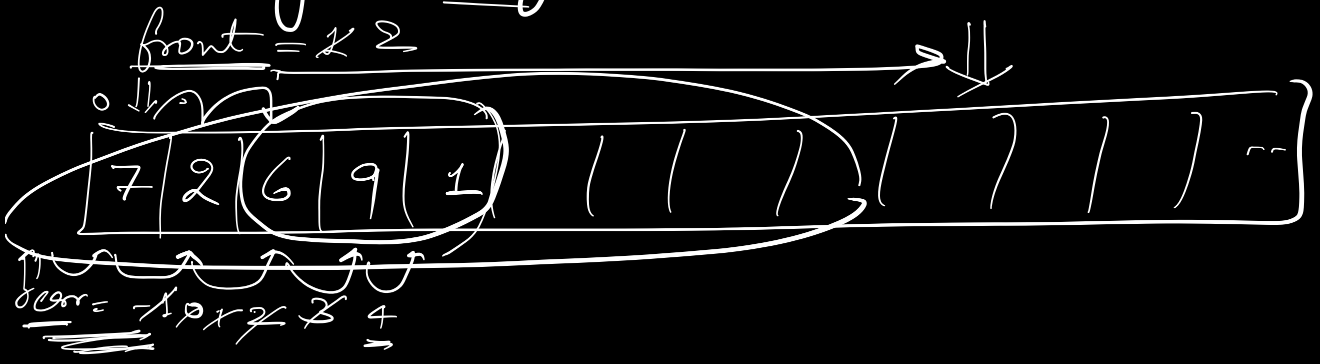




T.C. = O(K)

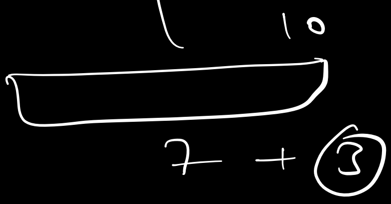
Implementation of Queue

1) Using Array



```
Insert (Enqueue) {

    rear++
    arr[rear] = value;
}
```

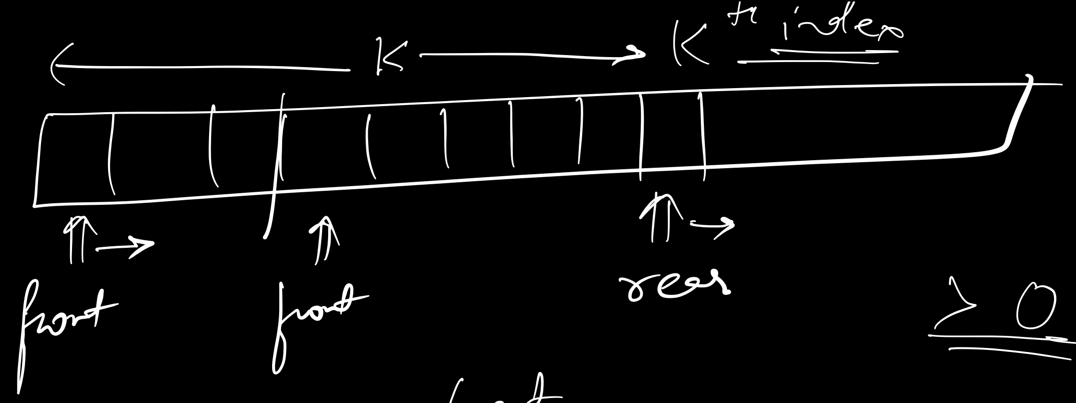
```
Dequeue (); {

    front++;
}
```



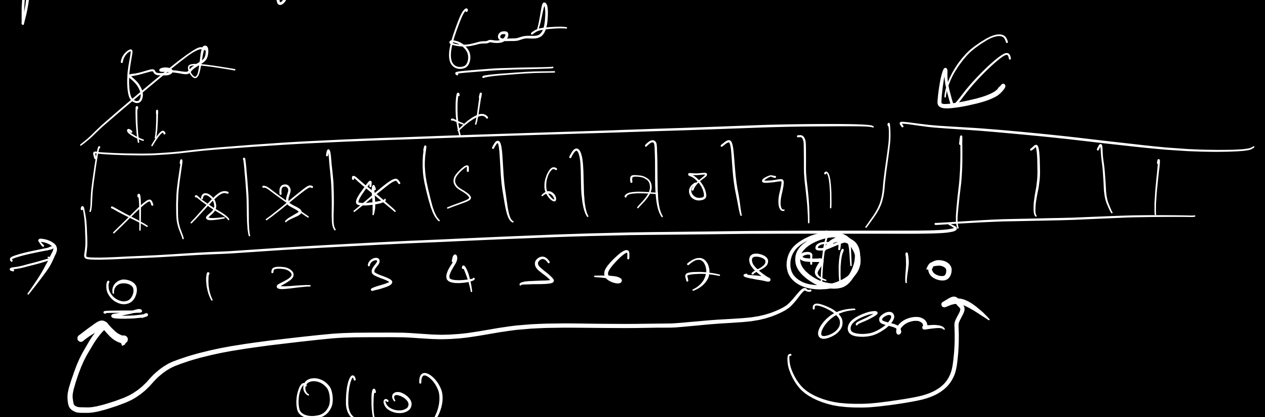
1) fixed Size

2) Memory wasted

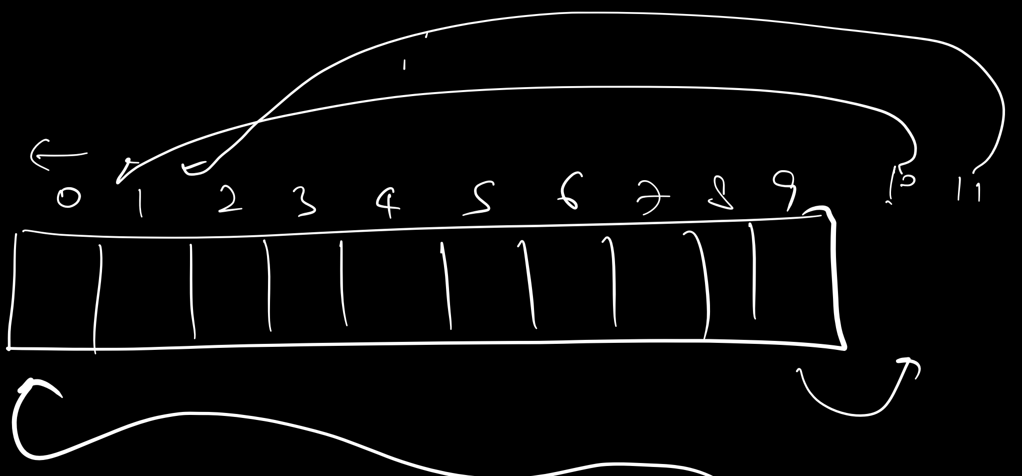
Size of the queue is fixed.



$\geq O(K)$



O(10)



9+1 ⇒ 0

9+2 ⇒ 1

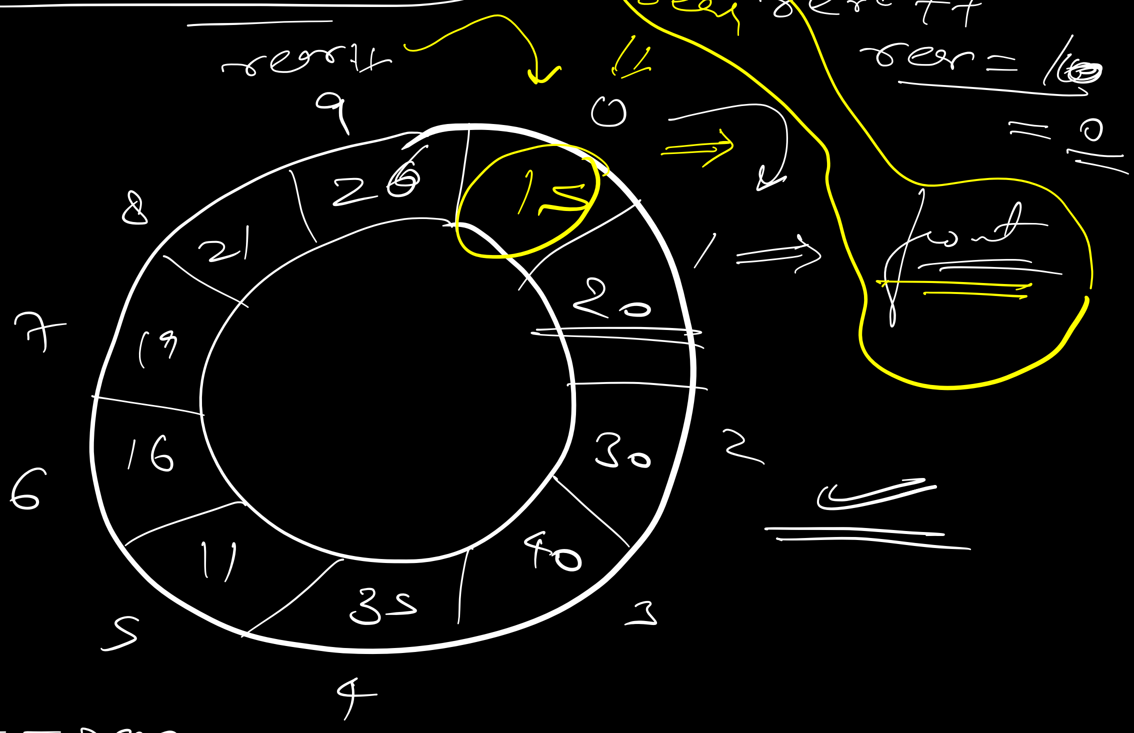
9+3 ⇒ 2

rear % 10

$\boxed{10}$

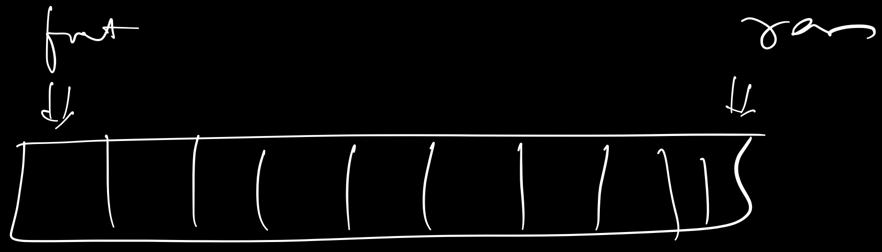
rear = rear % 10

⇒ Circular Queue



front == rear

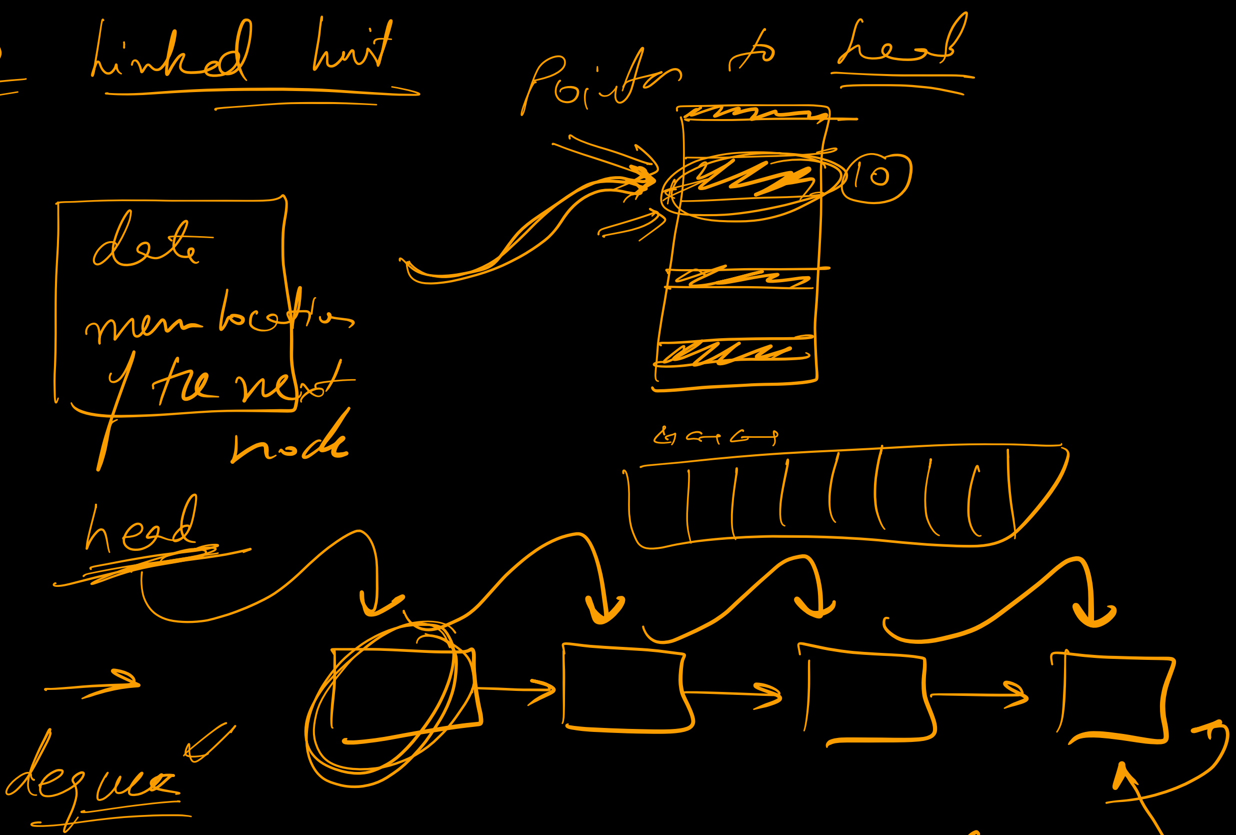
⇒ Conditions to check if the queue is full.



(i) if (front == 0 && rear == n-1)

(ii) if (rear == front - 1)

3 Linked List



temp = mem of next node

free (head)

head = temp;

pointer = zero

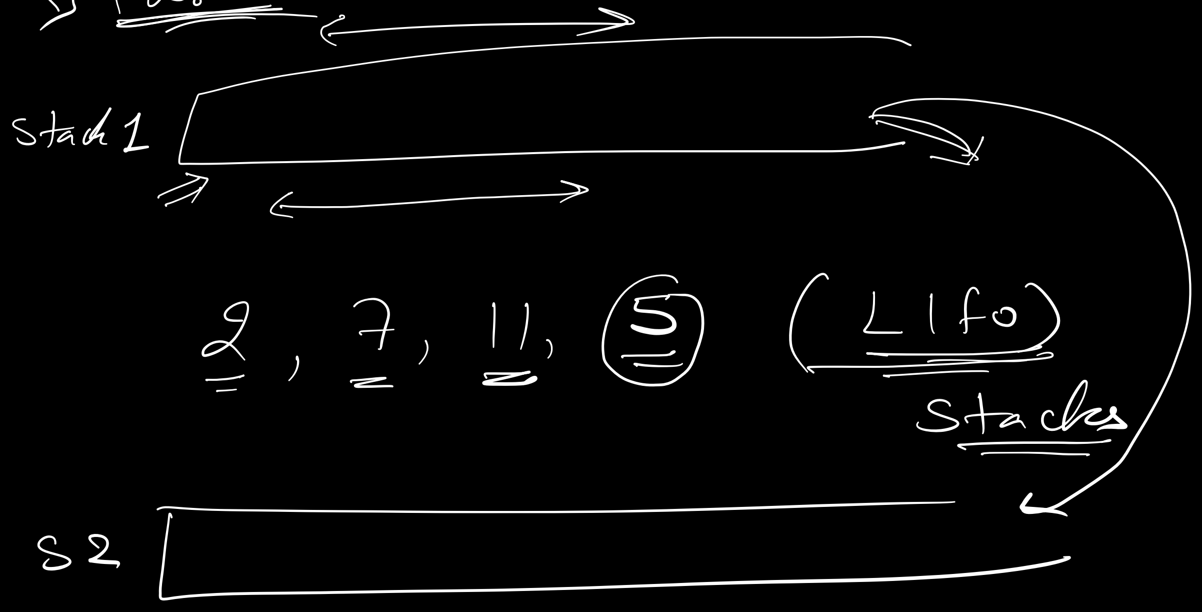
Stacks	Queues
1) LIFO	1) FIFO
2) insertion & deletion happens on the same side i.e top	2) insertion & deletion takes place from opposite ends

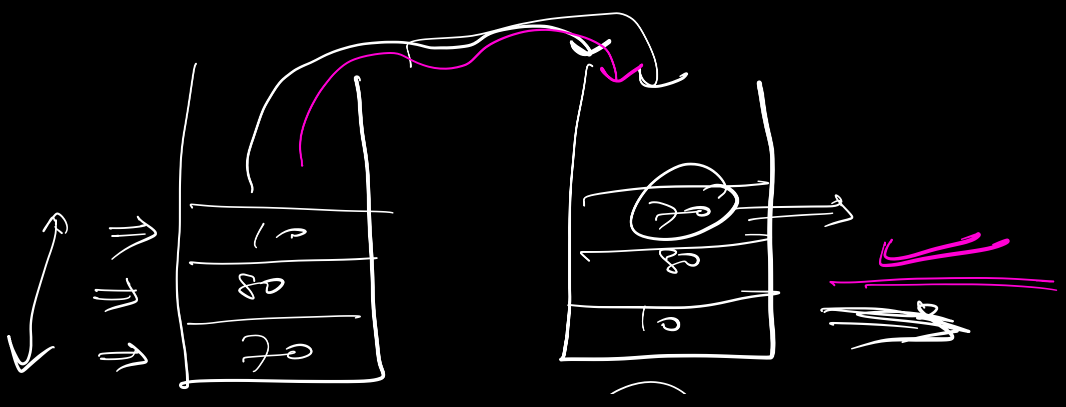
push / pop | Enqueue / dequeue

Implement Queue DS using Stacks (push / pop)

Enque / Dequeue ⇒ O(1)

Push





S1 S2 ⇒

~~20~~	30	40	70	80	10

1) E20

2) E30

3) E40

⇒ 4) Dequeue

5) Dequeue ⇒ S2.pop()

6) E 70

7) E80 , E10

1) Enqueue ⇒ S1.push_back(element)

T.C. = O(1)

2) Dequeue ⇒

```
(i) if (!S2.isEmpty()) {
        return S2.pop();
    }
    else if (!S1.isEmpty()) {
        while (!S1.isEmpty()) {
            S2.push_back(S1.pop());
        }
        return S2.pop();
    }
```

Amortized

T.C. ⇒ O(1)

Q H.W. Given all binary numbers (0 or 1)

Generate first K binary numbers

LIFO

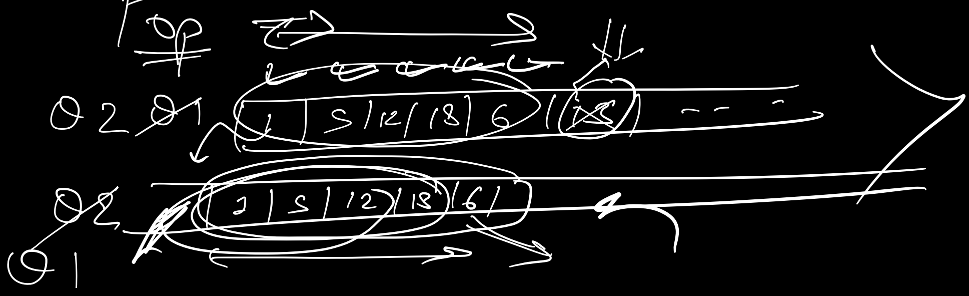
Q Queue Implementation of Stack

Queue DS given < Enqueue ⇒ O(1)
Dequeue O(1)

Implement a stack

Push ⇒ Enqueue ⇒ O(1)

Pop ⇒

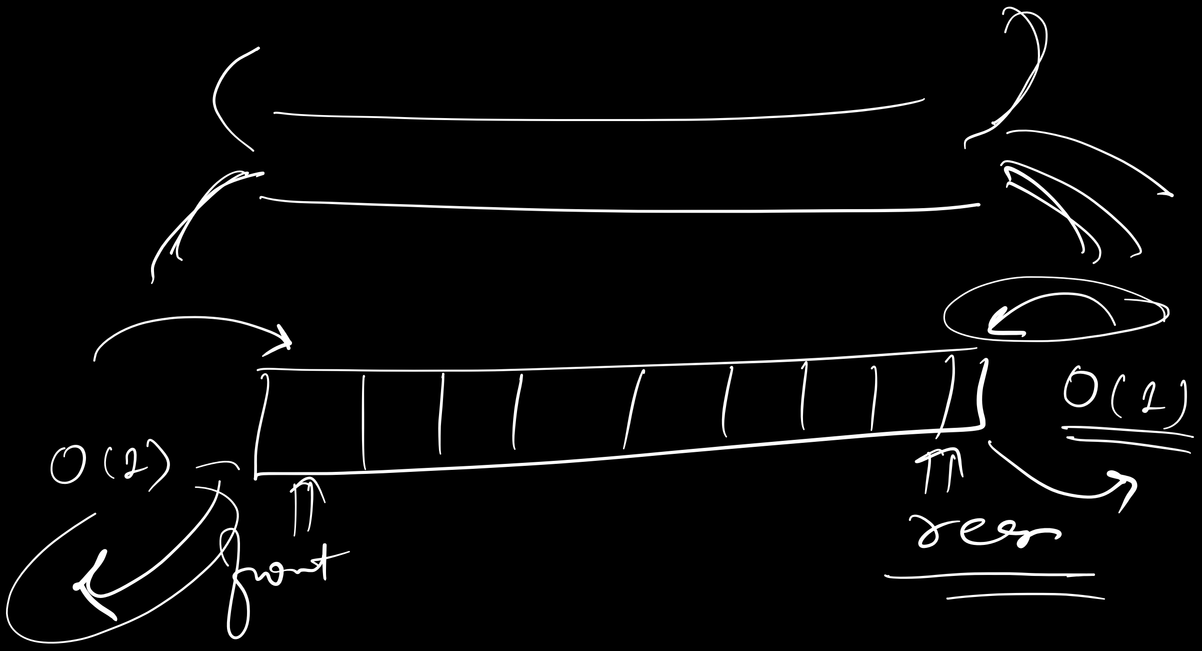


O(size of the)

Deque

⇓

Double Ended Queues



✓ Stack

Queue ✓