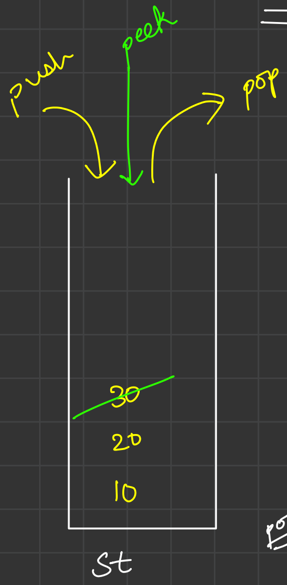
Queues

stack

↳ LIFO

TC : O(1)

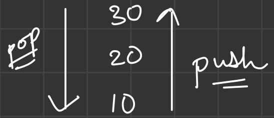


st push(10)

st . push (20|

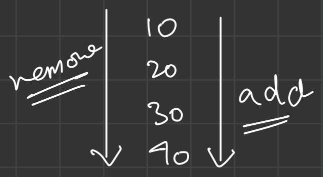
st . push(30)

st . pop ()

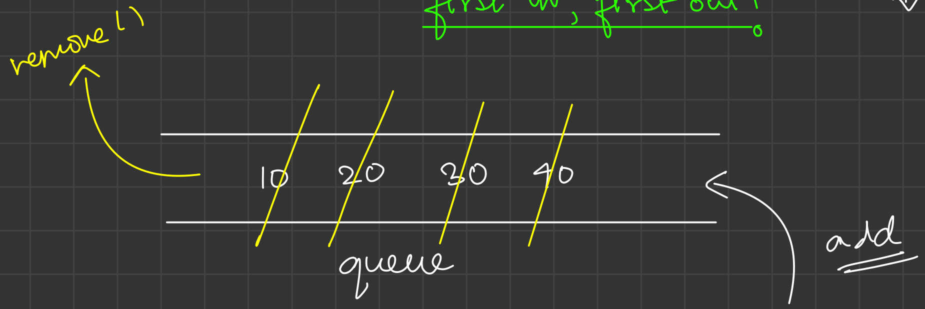


Queues

↳ linear data structure



first in, first out!



que.add(10) 1

que add(20)

que add(30)

que. add(40)

que.remove() → 10 1

que.remove() → 20

que.remove() → 30

que. remove() → 40

enqueue

→ enter + queue

→ Add an element to queue

dequeue

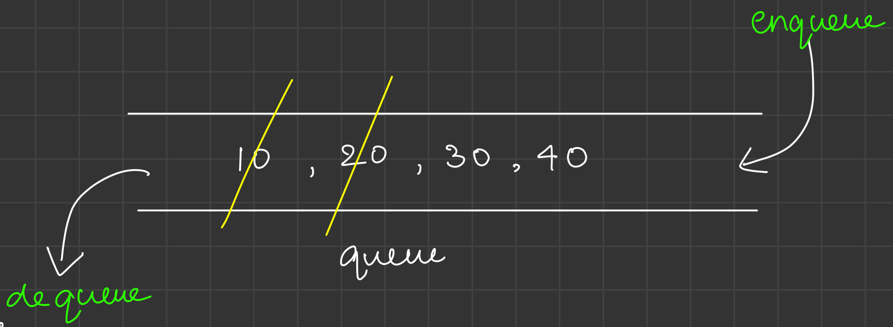
→ delete + queue

→ removal of an element from a queue

Front()

↓

to get/view the first person in the que.



Queue < G > que_name = new ArrayDeque<>();

= new LinkedList<>();

Queues

↳ linear data structure

↳ it follows FIFO (First in, First out)

Methods

enqueue , dequeue , ③ front() , ④ size()

enqueue ↓ ① add()

offer()

dequeue ↓ ② remove()/poll()

$\boxed{TC: O(1)}$

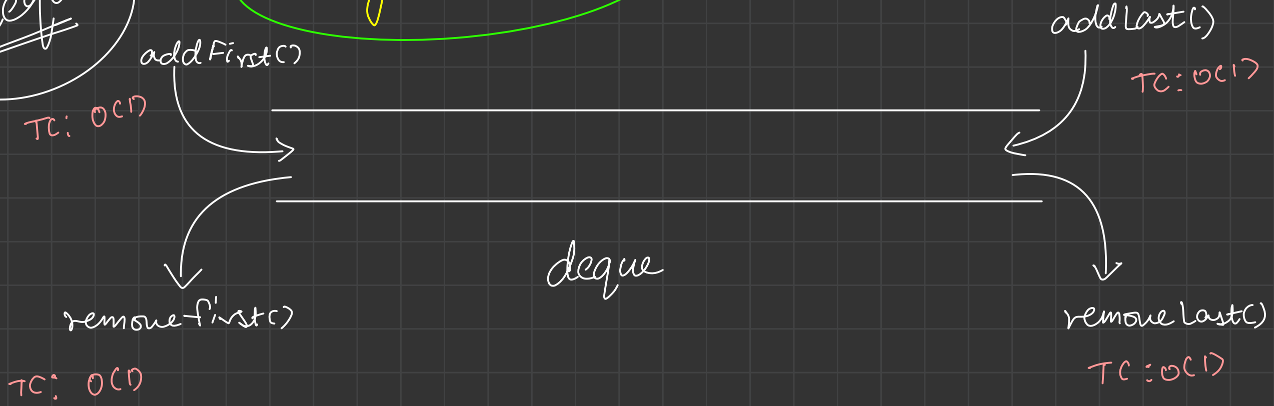
BFS

↓

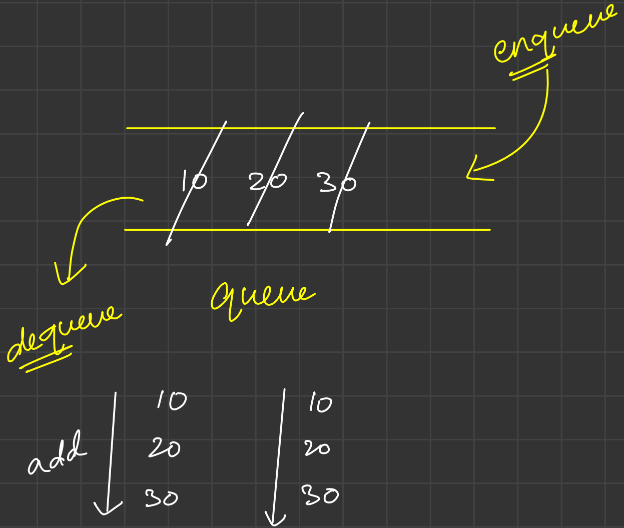
Breath first Search Algo

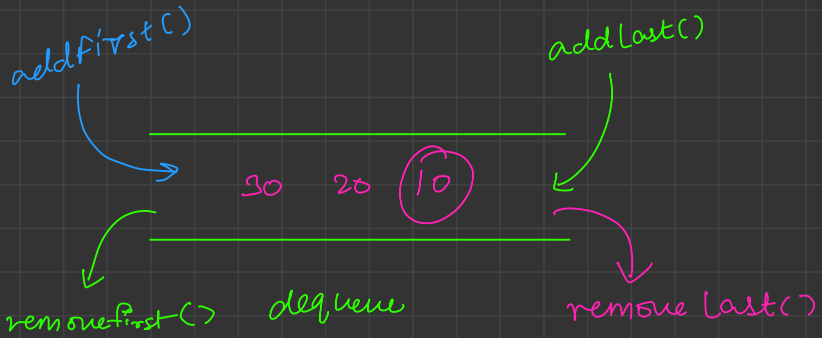
linear data structures

deque → doubly ended queue → implemented using a doubly linked list!



Q Can you implement a queue using a deque?

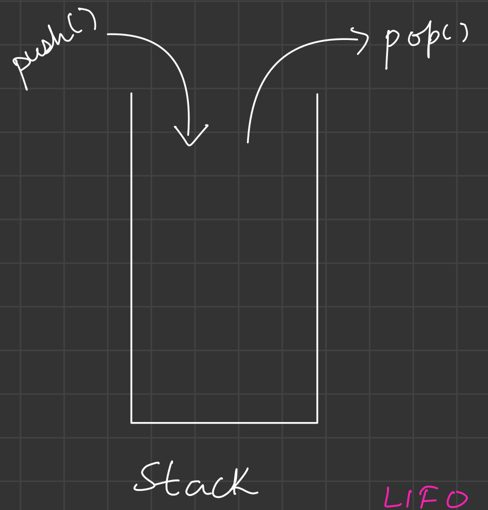




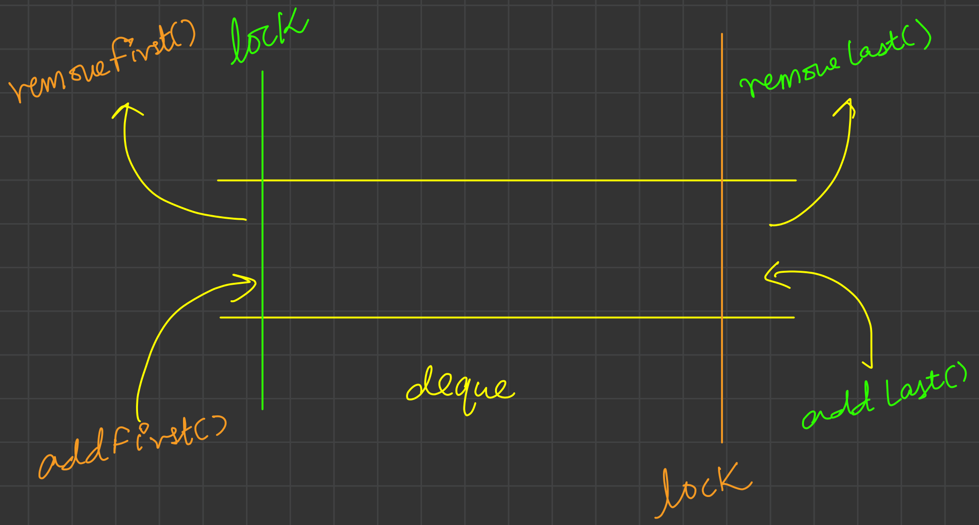
addLast()
removeFirst() → queue

addFirst()
remove Last()

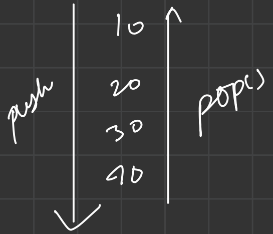
Q Can you implement a stack using deque?

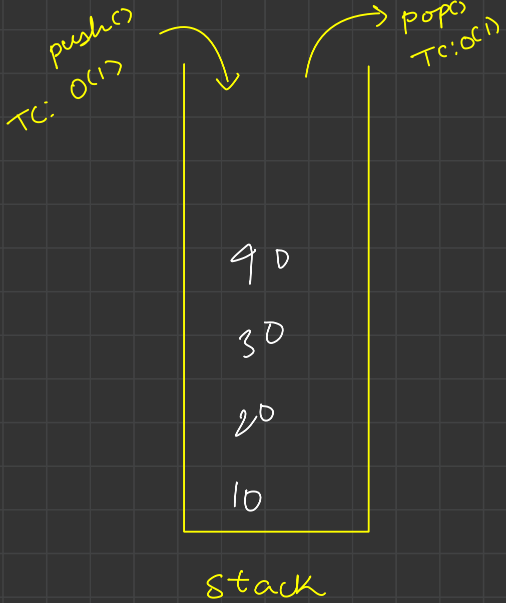


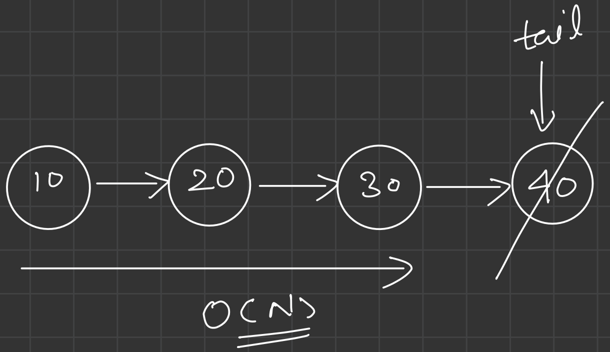
LIFO



Q design a stack using Linkedlist.....

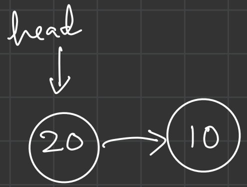






addLast() → O(1)

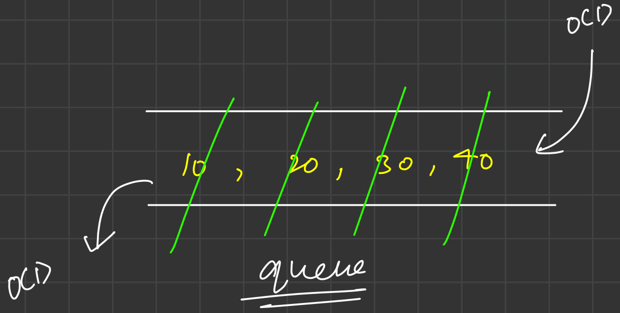
remove last() → O(N) ✗

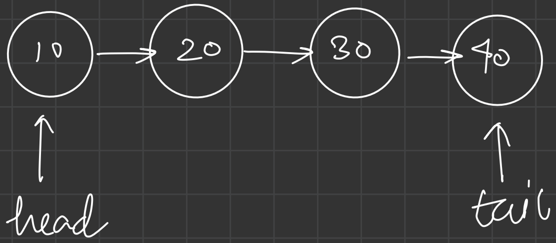


addFirst() → O(1)

remove First() → O(1) ✓

Q Design a queue using linked list

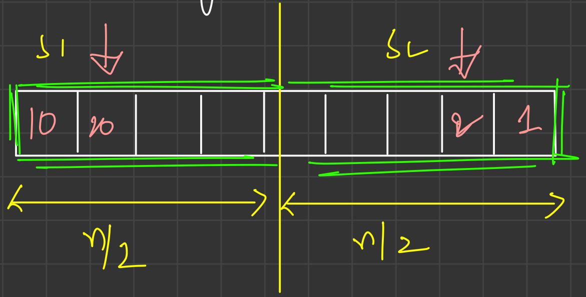


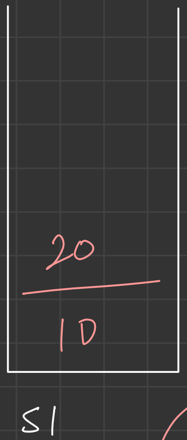


add()	remove()
10	10
20	20
30	30
40	40

addLast() O(1)
removeFirst() O(1)

implement 2 stacks using an array! → Amazon!





S1

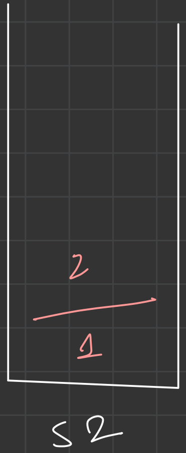
add(10)

add(20)

S2

add(1)

add(2)



int top1

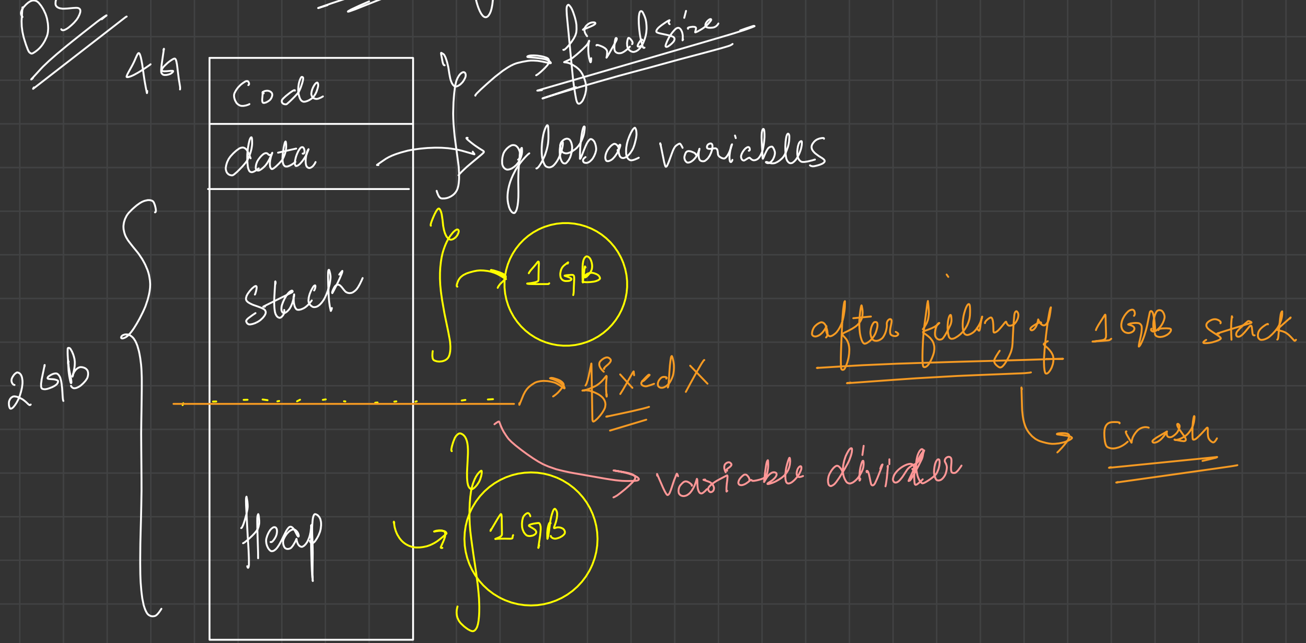
int top2

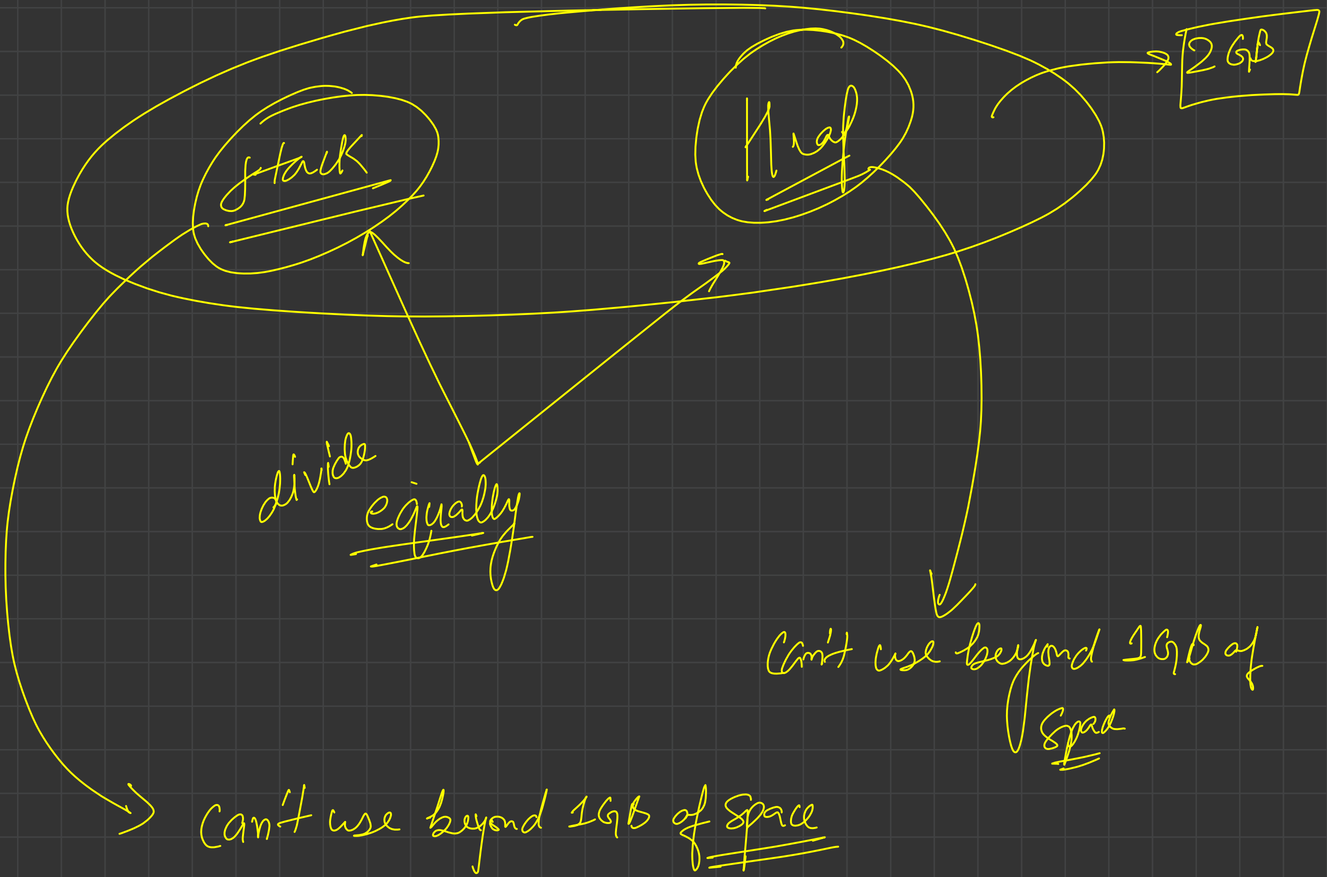
arr[top1] = 0

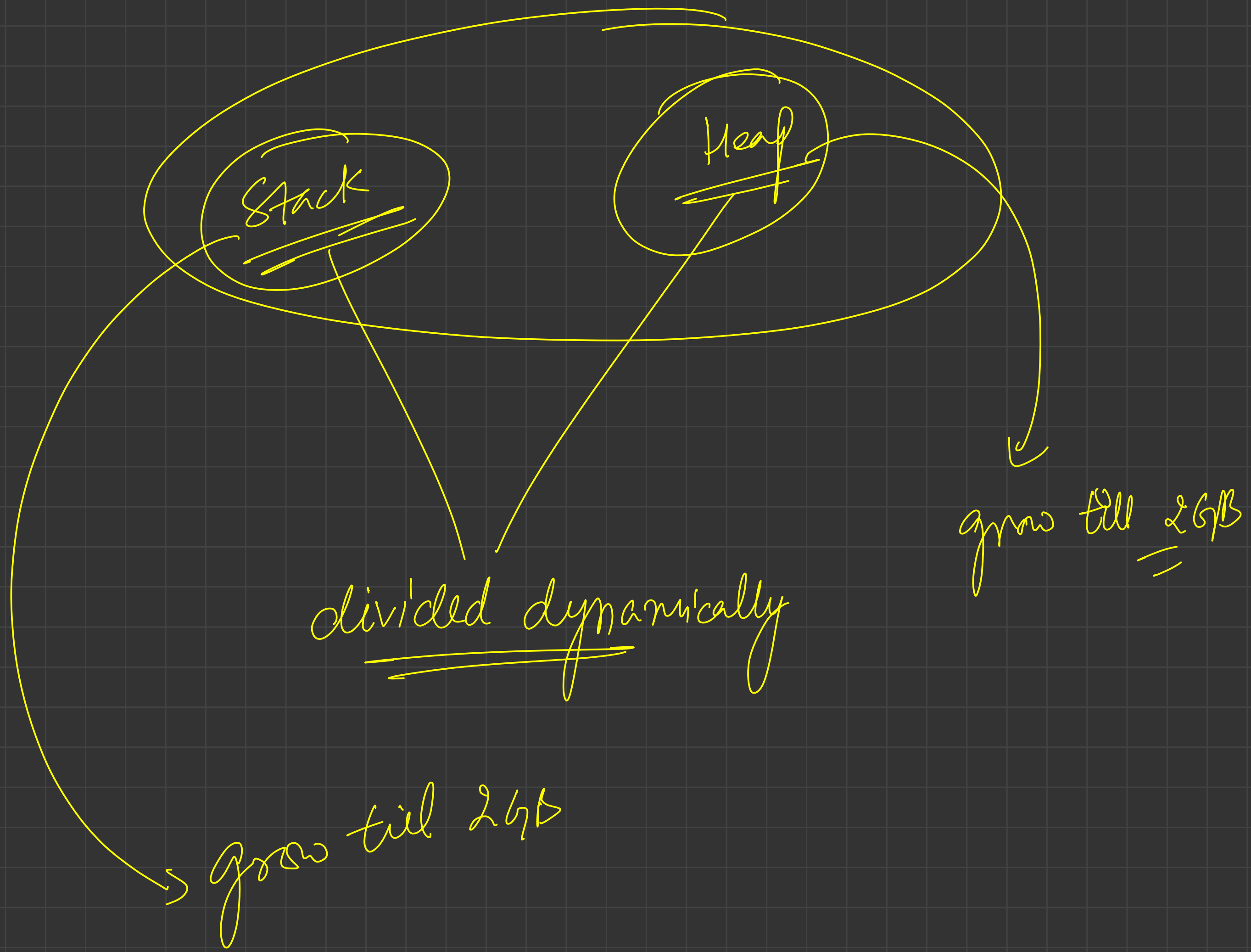
top1--;

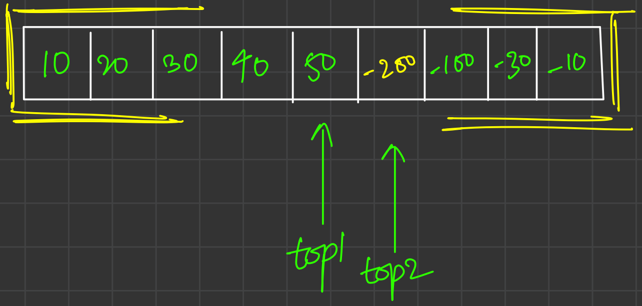
OS

Memory


4G
Code
data
Stack
Heap
2 Gb
fixed size
global variables
1 GB
1 GB
fixed X
variable divider
after filling of 1GB stack
Crash


Stack
Heap
2 GB
divide equally
Can't use beyond 1GB of space
Can't use beyond 1GB of Space


Stack
Heap
divided dynamically
grow till 2GB
grow till 2GB



1. S1.push(10)
2. S1.push(20)
3. S1.push(30)
4. S1.push(40)
5. S1.push(50)
6. S1.push(60)
7. S2.push(-10)
8. S2.push(-30)
9. S2.push(-100)
10. S1.pop()
11. S2.push(-200)

Implement Queue using 2-Stacks

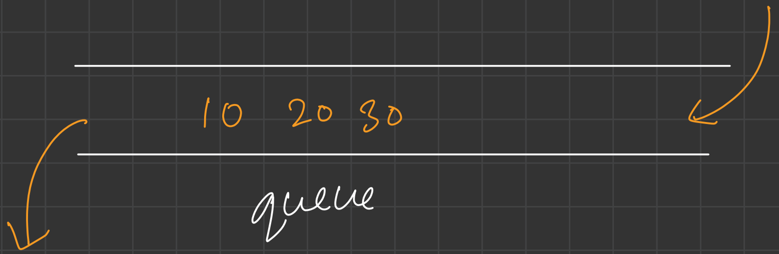
ques1

enqueue TC: O(1)

dequeue TC: O(1)

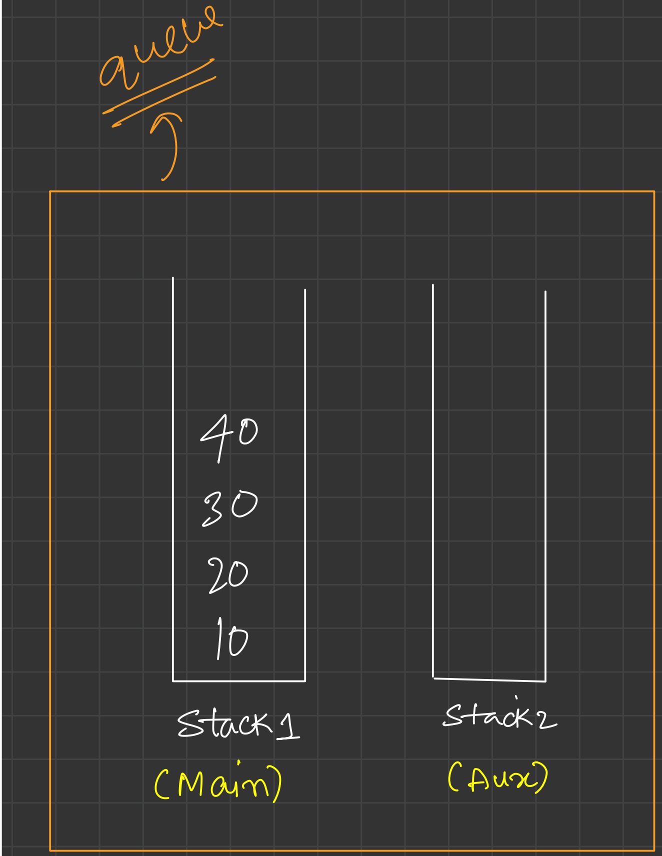
ques2

Enqueue → O(1)



add(10)
add(20)
add(30)

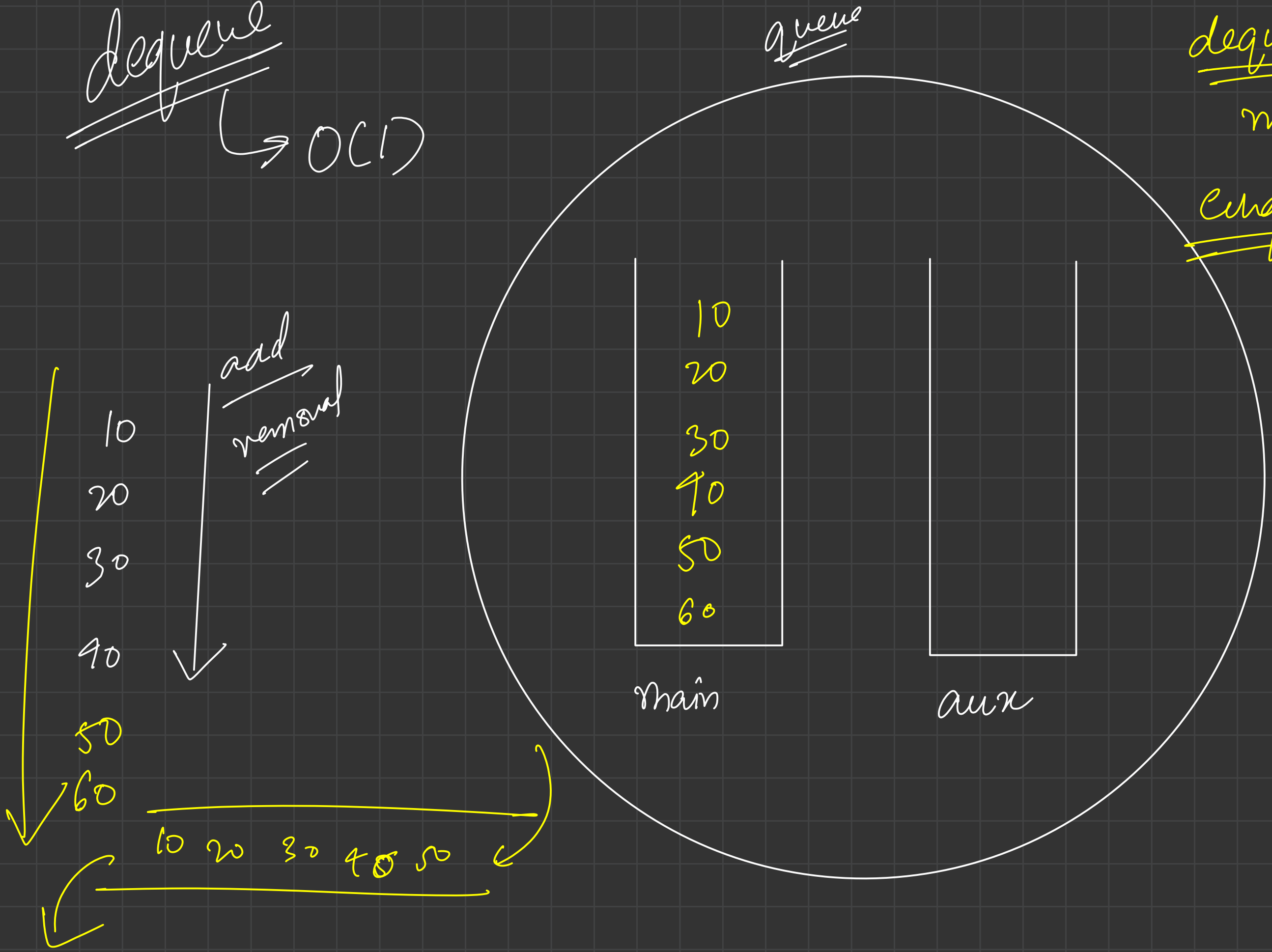
remove() → 10



main.push(x) → enqueue O(1)
dequeue O(N)

dequeue

↳ O(1)



dequeue → O(1)

mainStack.pop()

enqueue → O(N)

↳ add to the bottom of the stack