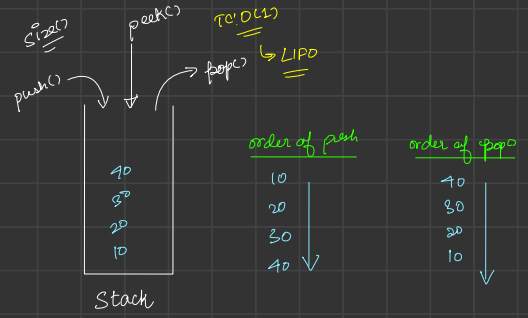
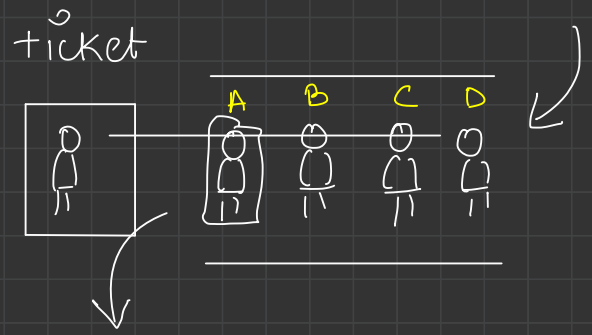
Queues.

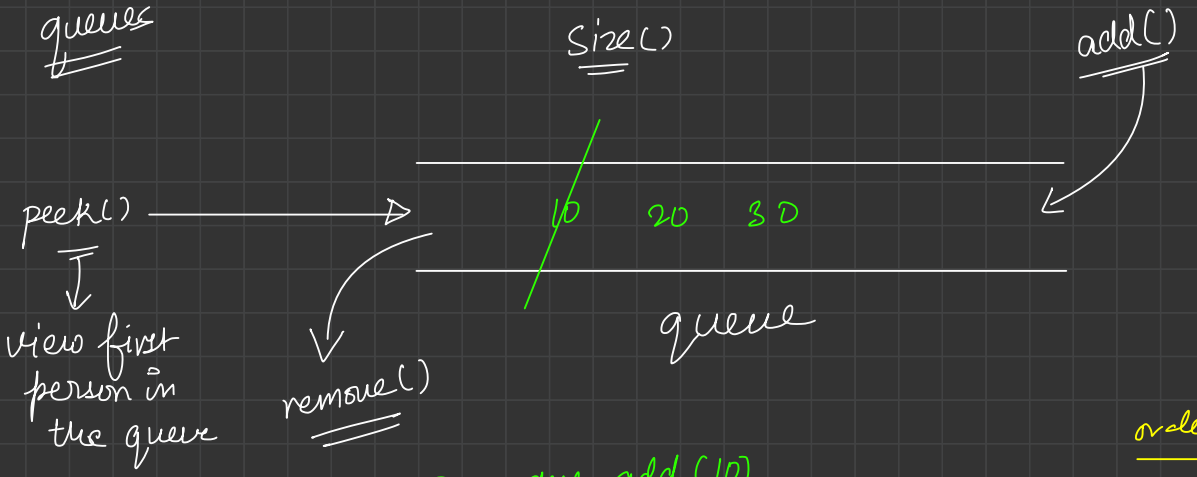
→ linear data structure

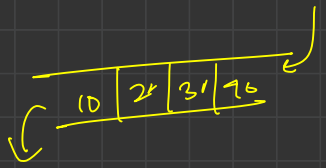
→ FIFO (first in, first out)





queues






```
que.add(10)
que.add(20)
que.add(30)
que.peek()    → 10
que.remove()  → 10
que.peek()    → 20
que.size()    → 2
```

order of add

10
20
30
40

order of removal

10
20
30
40

Hence follow fifo

enqueue → enter + queue

TC : O(1)

→ adding a element to a queue.

add()

dequeue → delete + queue

TC : O(1)

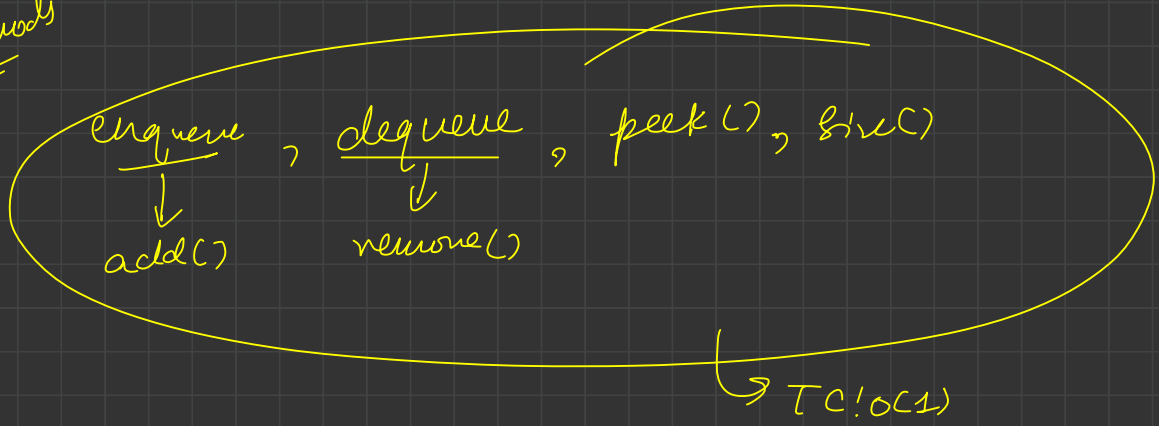
→ remove first element from the queue.

remove()

queue

- A linear data structure
- follows FIFO (first in, first out)

Methods



Method1

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

↳ add(), remove(), peek(), size().

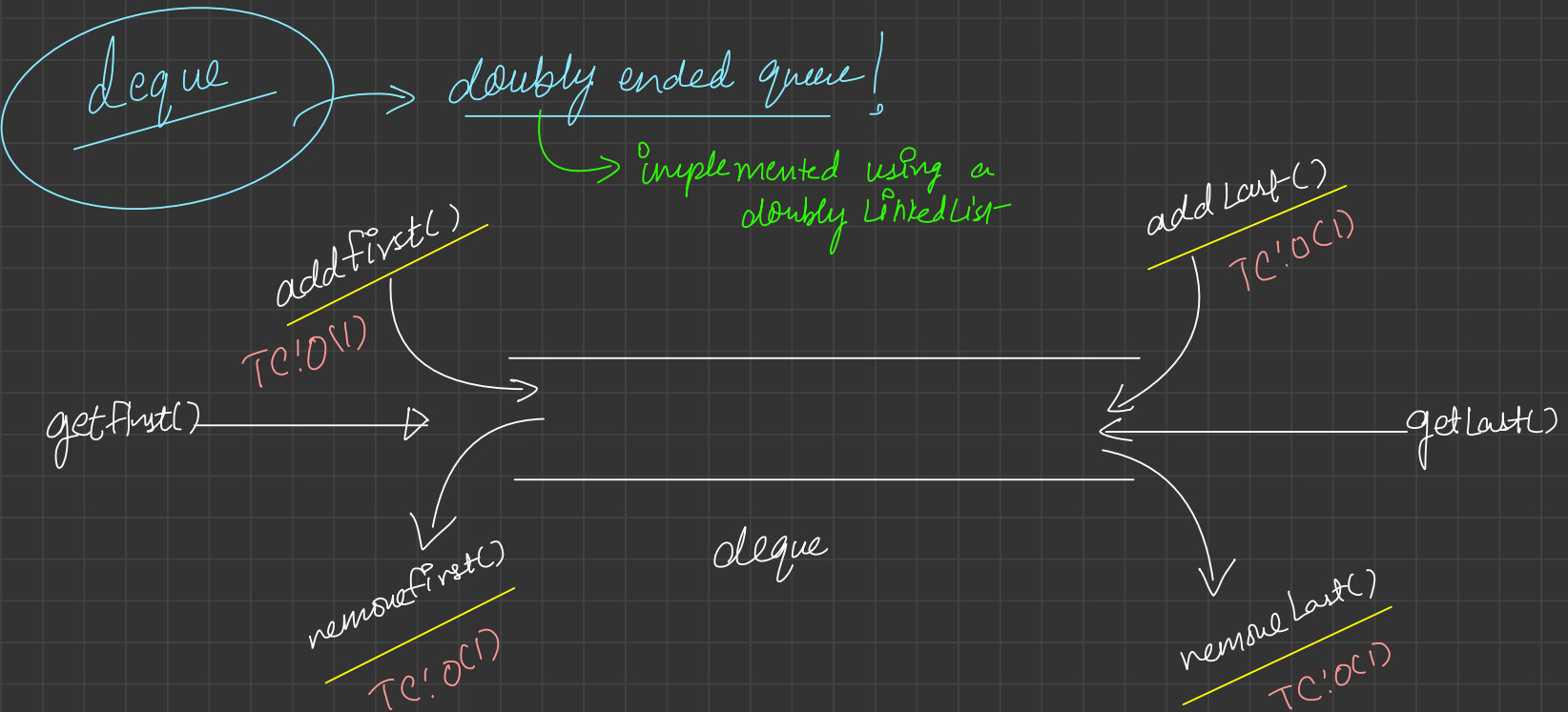
Method2

```
Queue<G> que_name = new LinkedList();
```

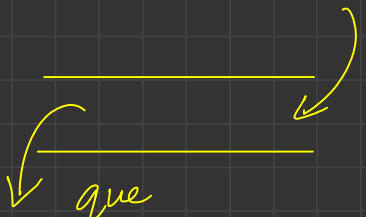
↳ poll(), offer(), peek(), size();

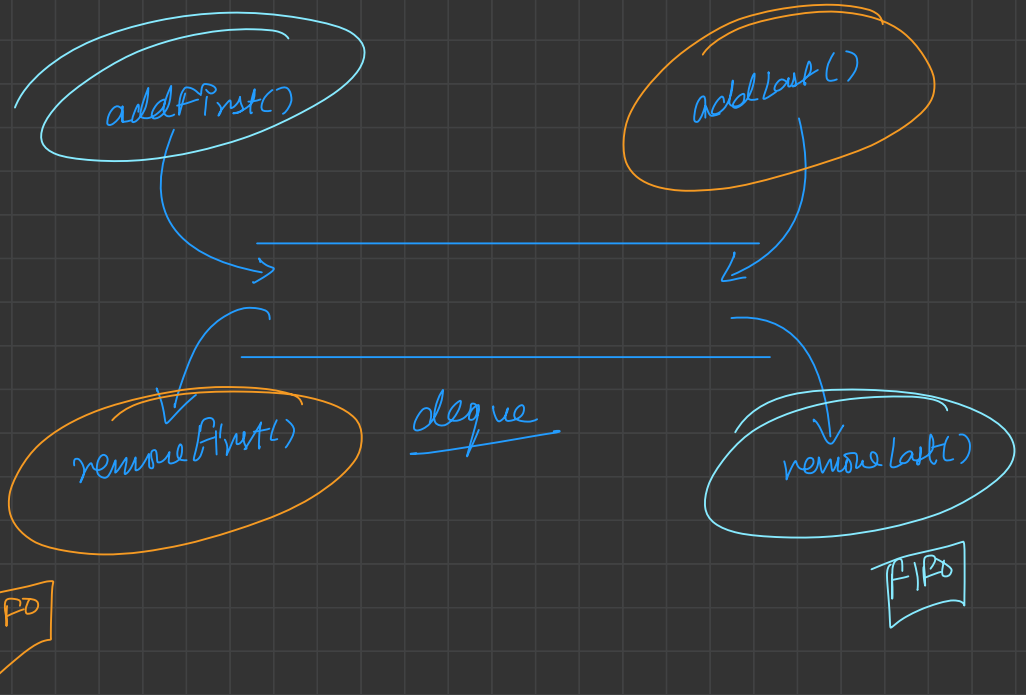
poll() → remove

offer() → add


deque
doubly ended queue!
implemented using a doubly LinkedList
addFirst()
TC: O(1)
addLast()
TC: O(1)
getFirst()
getLast()
deque
removeFirst()
TC: O(1)
removeLast()
TC: O(1)

Q Can you implement a queue using deque?





To implement a que using a deque.

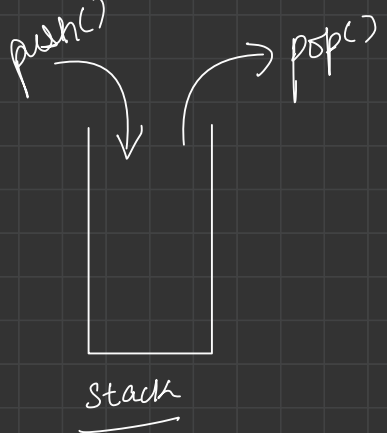
addFirst()
removeLast()

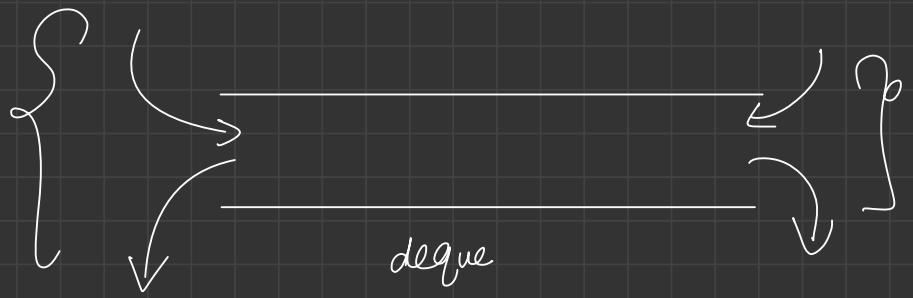
or

addLast()
removeFirst()

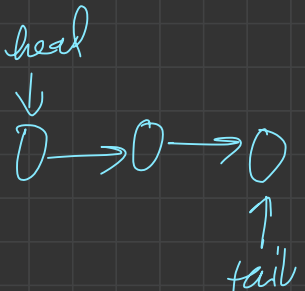
as they follow FIFO

Can you implement Stack using a deque?





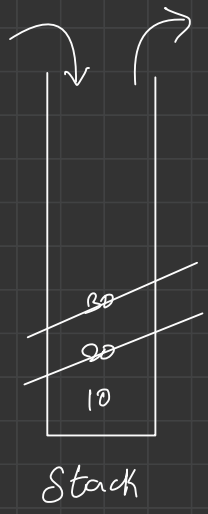
deque uses more memory than que!




```
class Node {
    int data;
    Node prev;
    Node next;
}
```

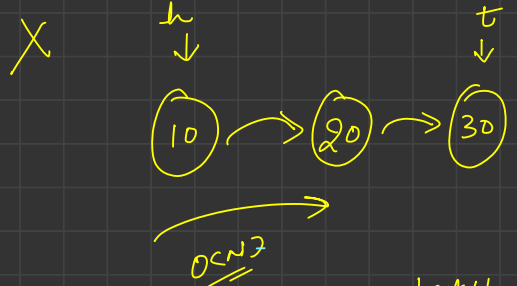
```
class Node {
    int data;
    Node next;
}
```

Q Design a stack using LinkedList?



eg:
- push(10)
- push(20)
- push(30)
- pop() → 30
- pop() → 20

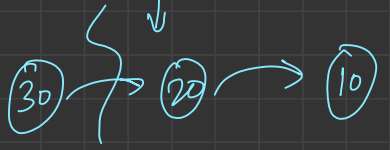
addLast() of LL for push() → TC: O(1)



O(N)

removeLast() of LL → O(N)

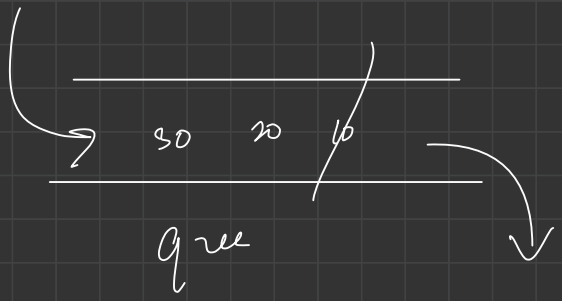
addFirst() in LL → TC: O(1)

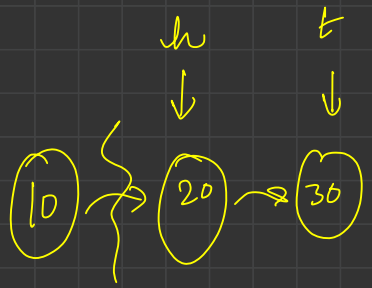


removeFirst() in LL → TC: O(1)

Implement a queue using LinkedList

LinkedList

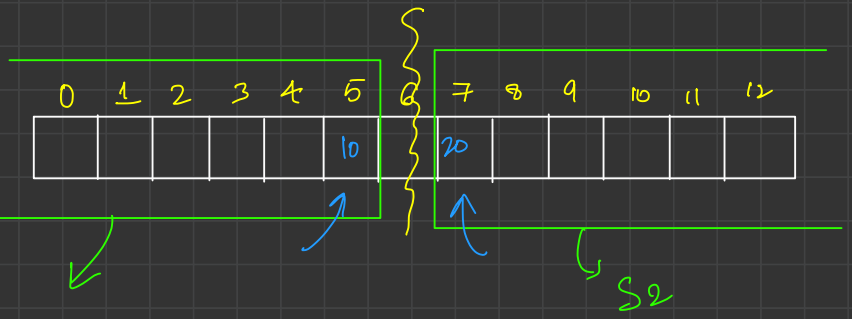




add() → addLast()
remove() → removeFirst()

- ✓ add(10)
- add(20)
- add(30)
- remove() → 10

implement 2 stack using a array.!



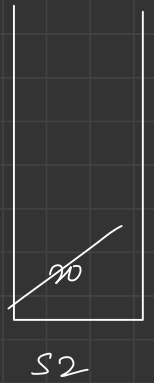
X

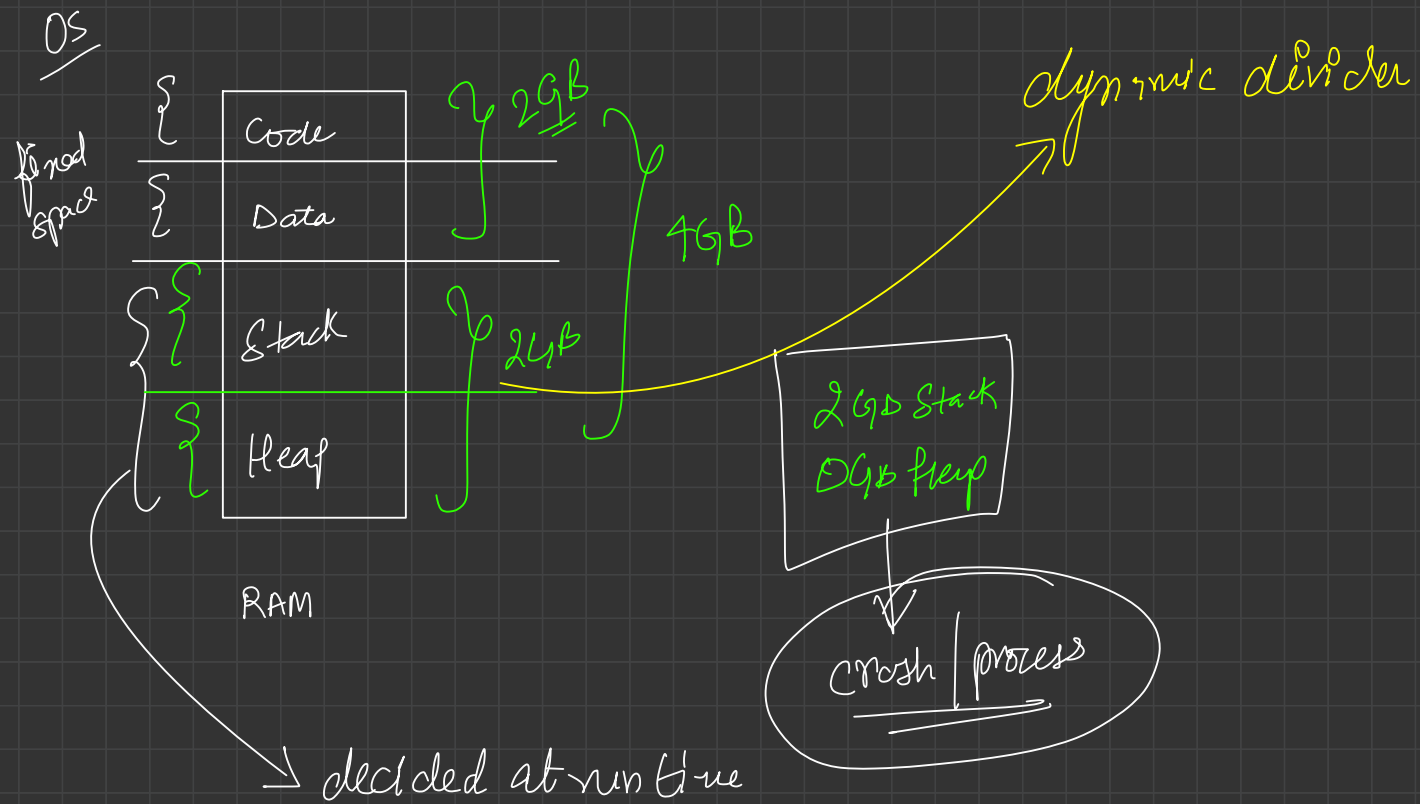
S1.push(10)

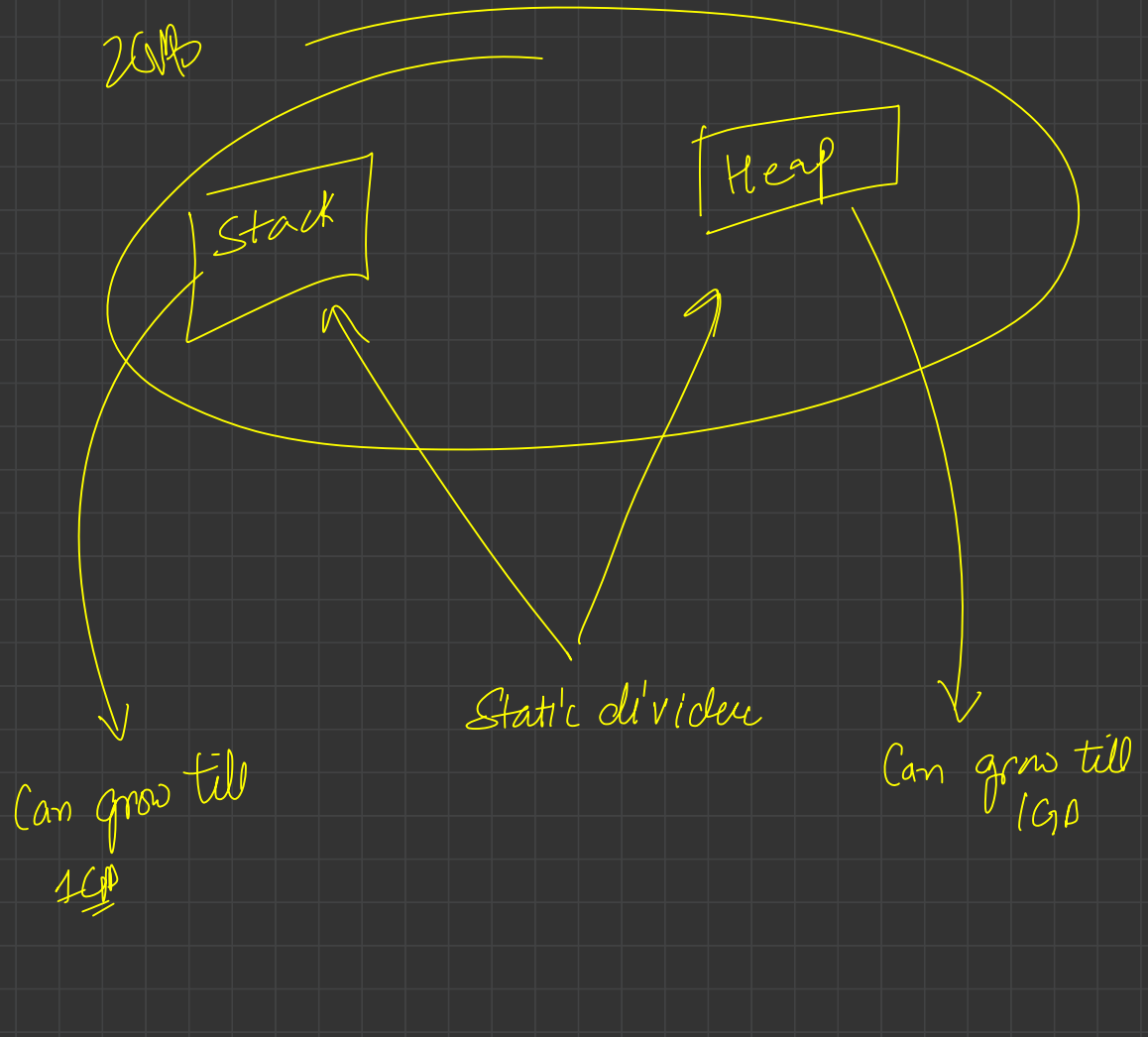
S2.push(20)

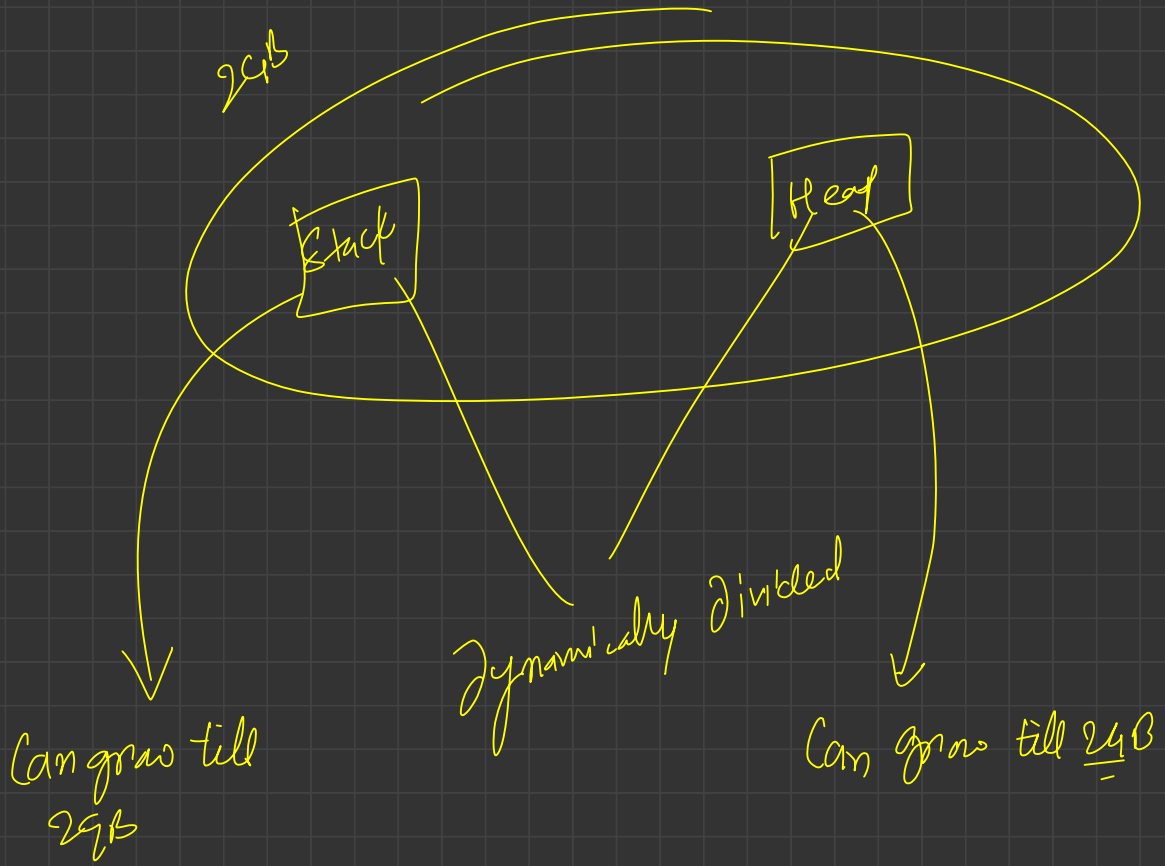
S1.pop() → 10

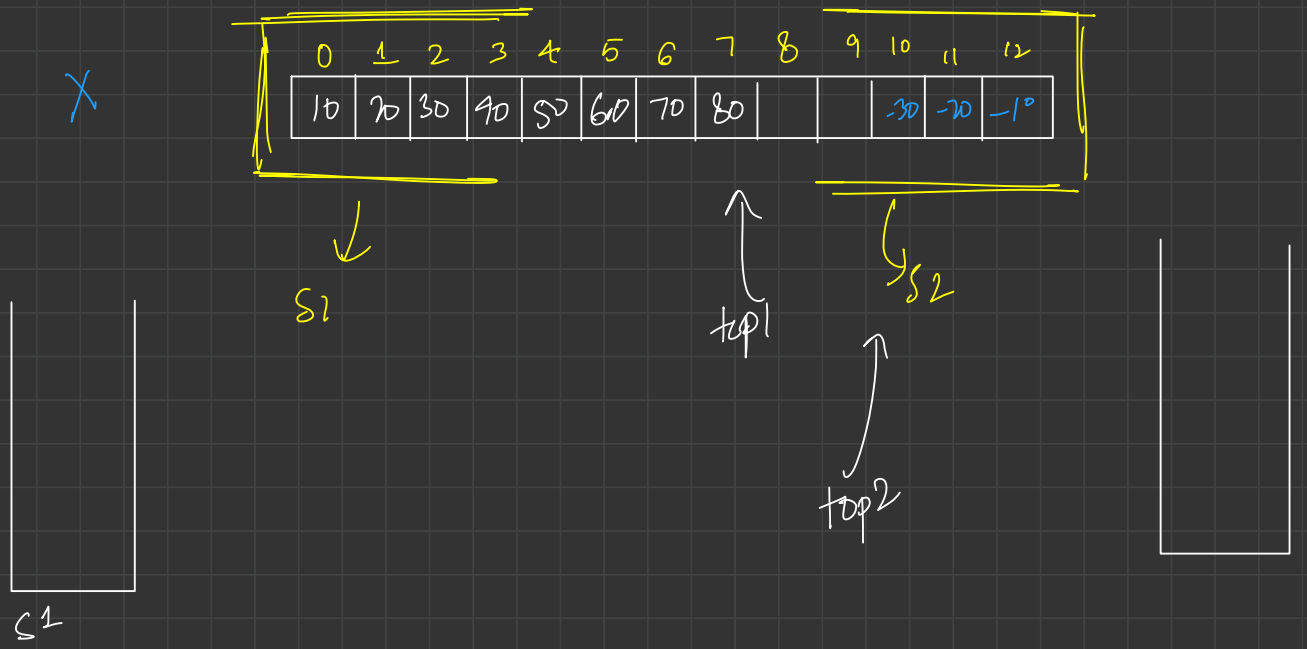
S2.pop() → 20



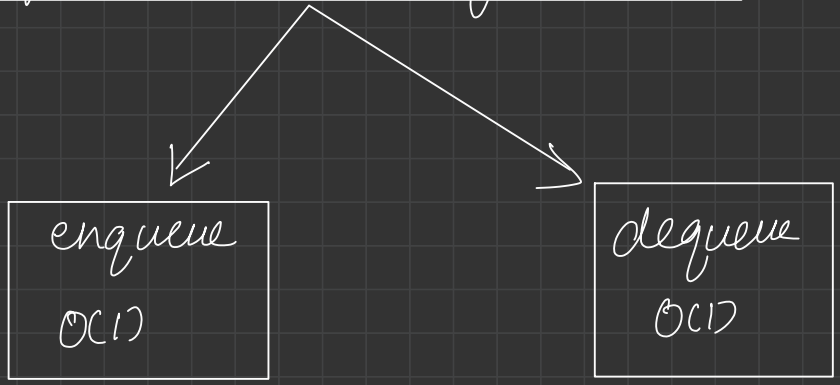

OS
fixed space
Code
Data
Stack
Heap
2GB
4GB
2GB
RAM
dynamic divider
2GB Stack
0GB Heap
Crash process
decided at run time


2GB
Stack
Heap
Static divider
Can grow till 1GB
Can grow till 1GB


2GB
Stack
Heap
Dynamically divided
Can grow till 2GB
Can grow till 2GB


X
0 1 2 3 4 5 6 7 8 9 10 11 12
10 20 30 40 50 60 70 80 -30 -70 -1°
S1
top1
S2
top2
S1

Implement Queue using 2-stacks

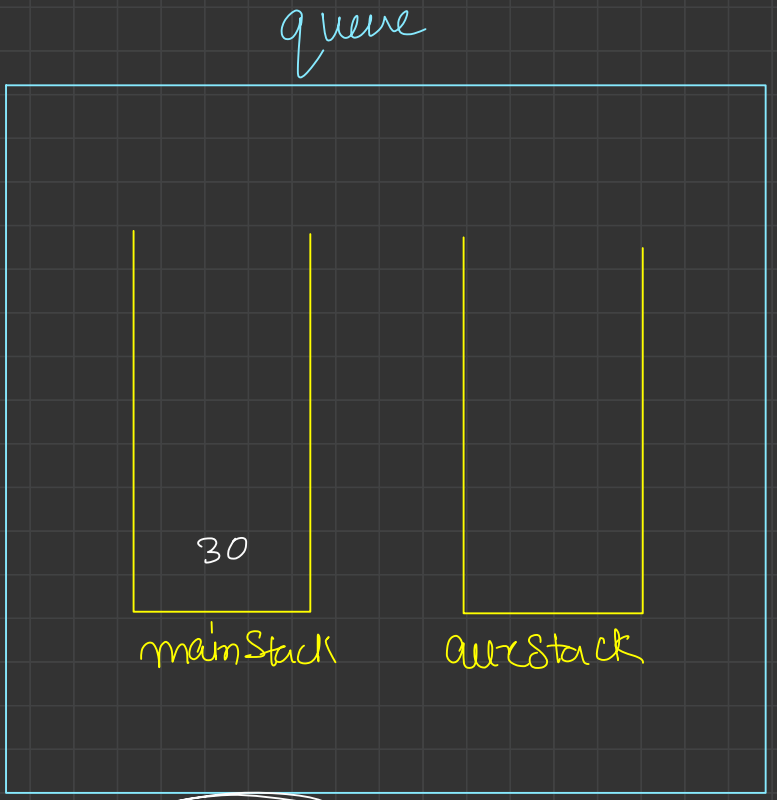


Enqueue

↳ O(1)

10

20



10 20 30

queue

add() ✓ removed() ✓

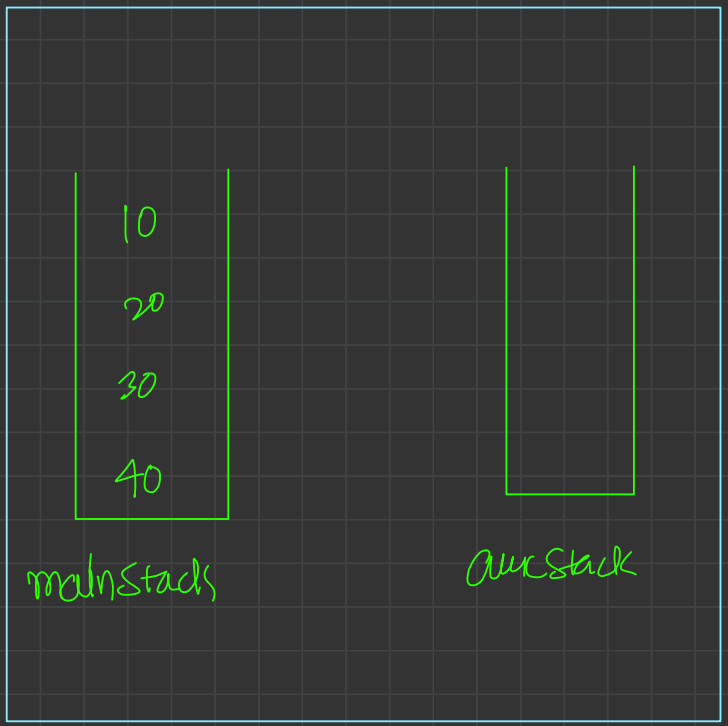
10 ✓ 10

20 ✓ 20 ✓

30 ✓ 30

dequeue O(N)

dequeue
↳ O(1)



10
20
30
40