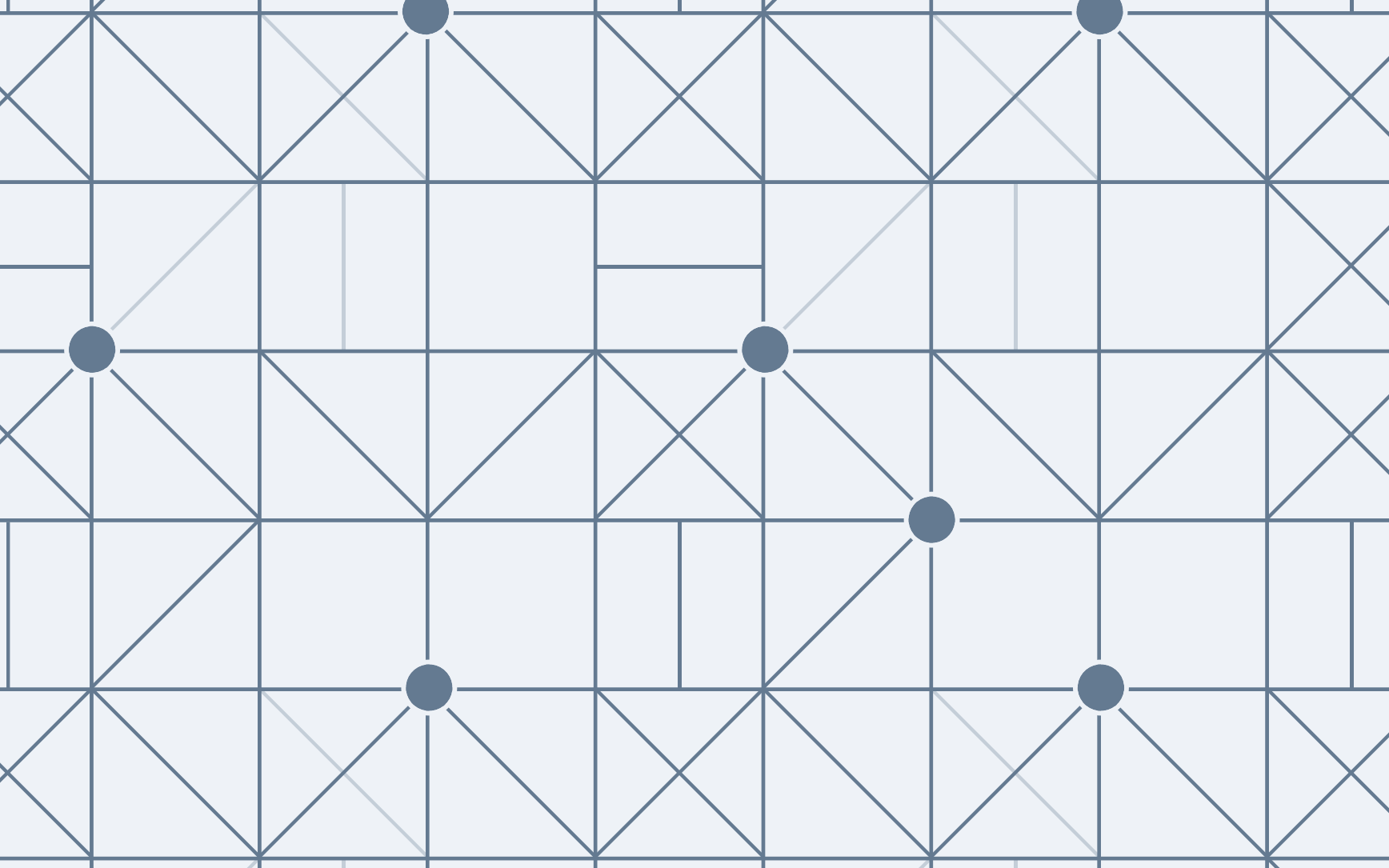


Queues.

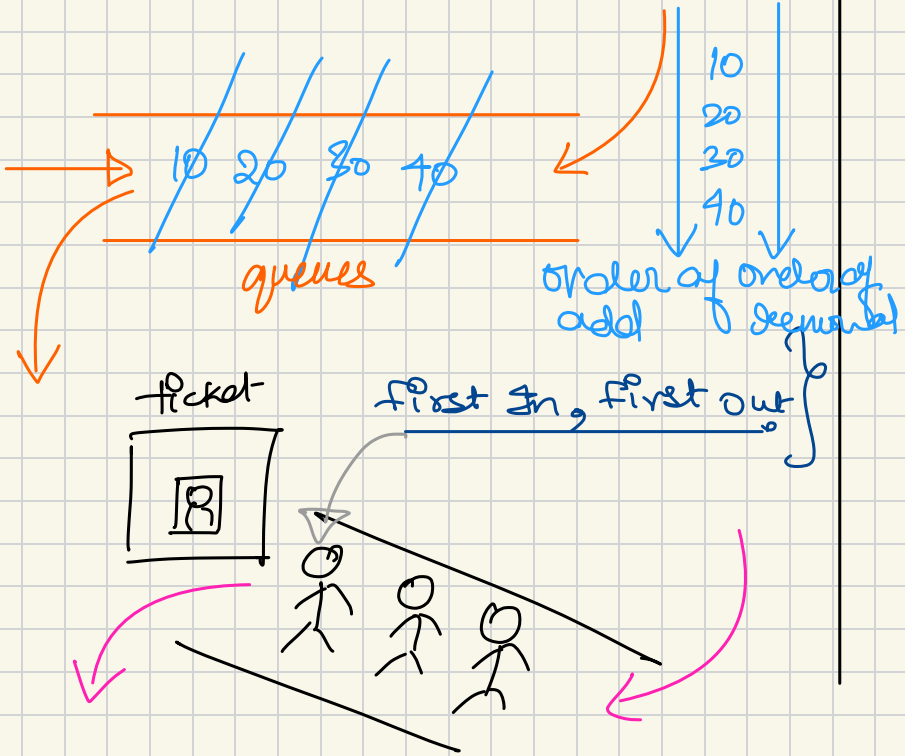
Agenda.

1. Introduction of queues
2. Queues and stack easy problems
3. 2 stacks in an Array.
4. Implement a que using stack

Queues

↳ Linear Data Structure

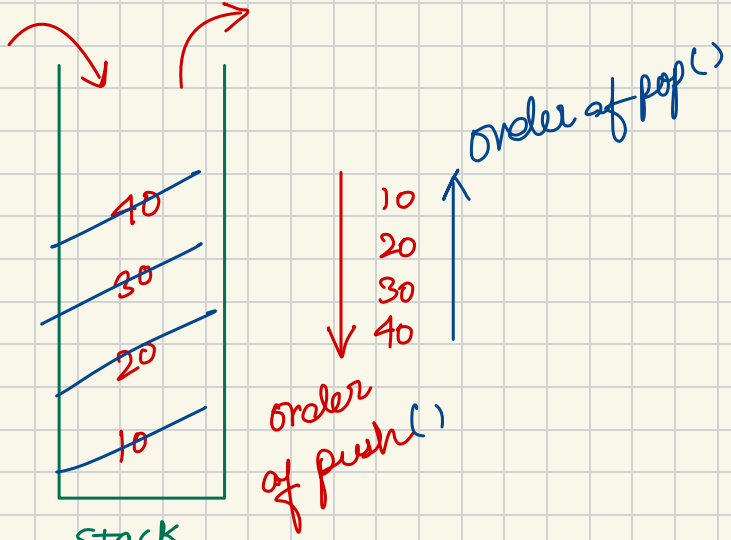
↳ FIFO {first in, first out}

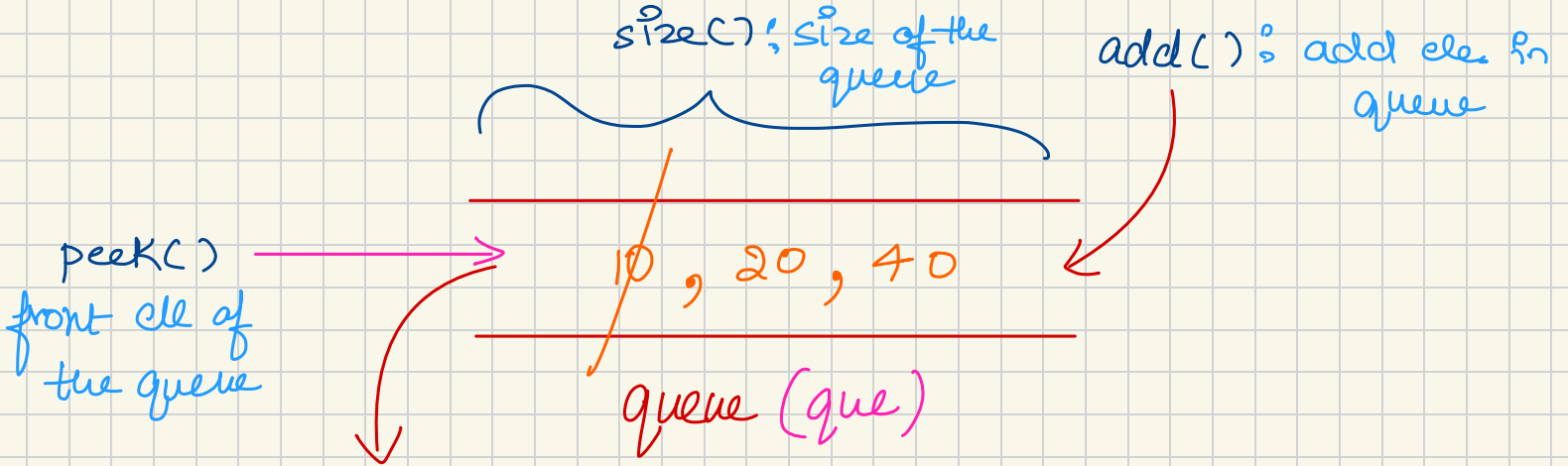


Stacks

↳ Linear DS

↳ follows LIFO






```
que.add(10)
que.add(20)
que.add(40)
que.size() ⇝ 3
que peek() ⇝ 10
que remove() ⇝ 10
```

Queue

↳ interface in Java

1. `Queue<G> que_name = new ArrayDeque<>();`
2. `Queue<G> que_name = new LinkedList<>();`

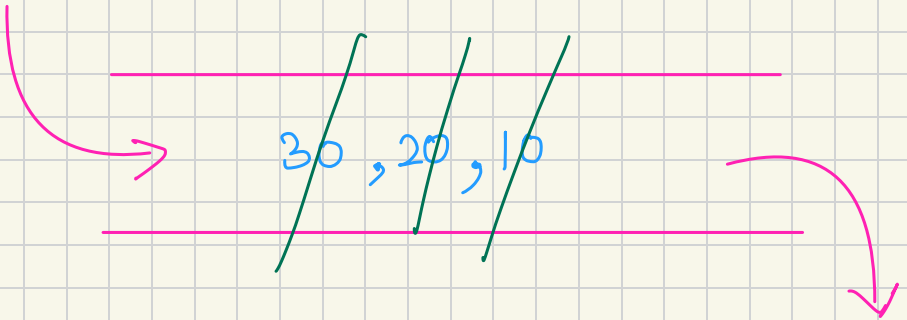
Methods

1. add / offer
2. remove / poll
3. peek
4. size

TC: O(1) SC: O(1)

Enqueue. → { Enter in Queue } add() | offer()

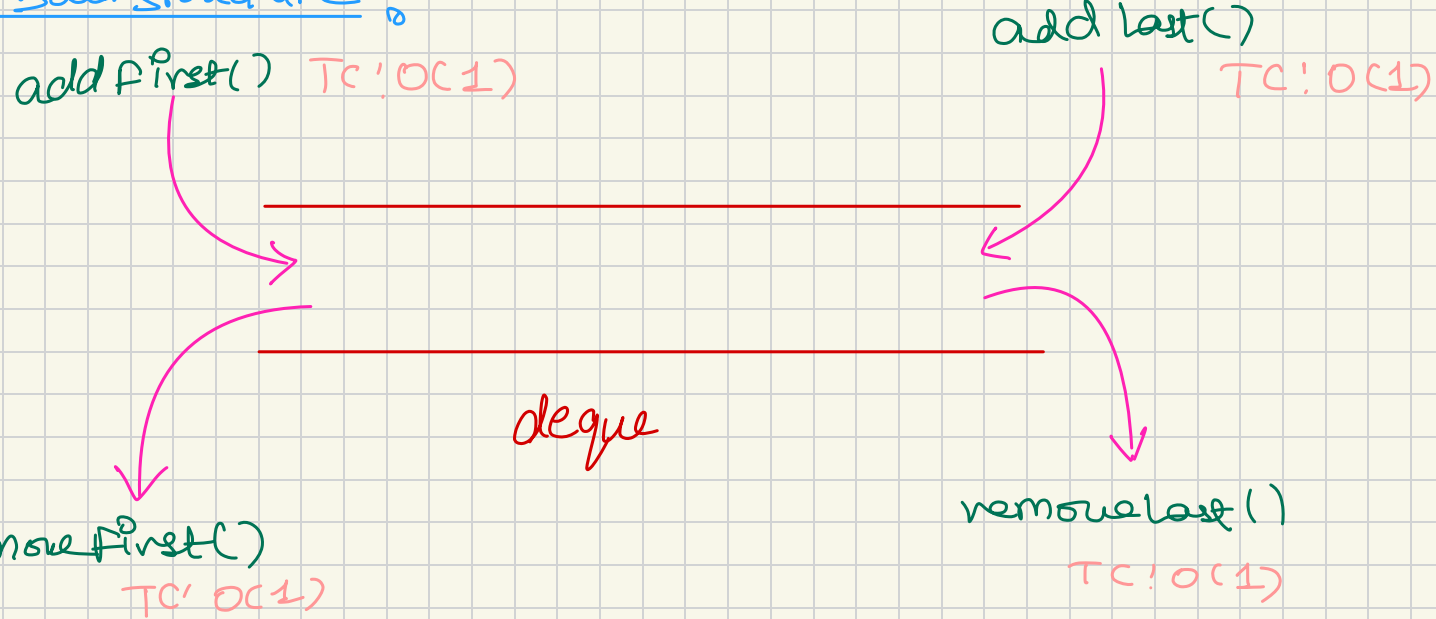
Dequeue. → { Remove from Queue } poll() | remove()



Deque. { Doubly ended queue }

→ { Implemented using doubly linked list }

Linear Data structure.



queue

↓

Linked List

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

(x+y)

deque

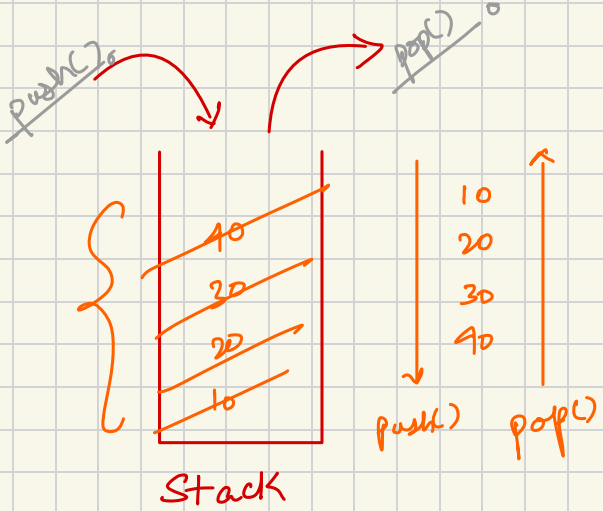
↓

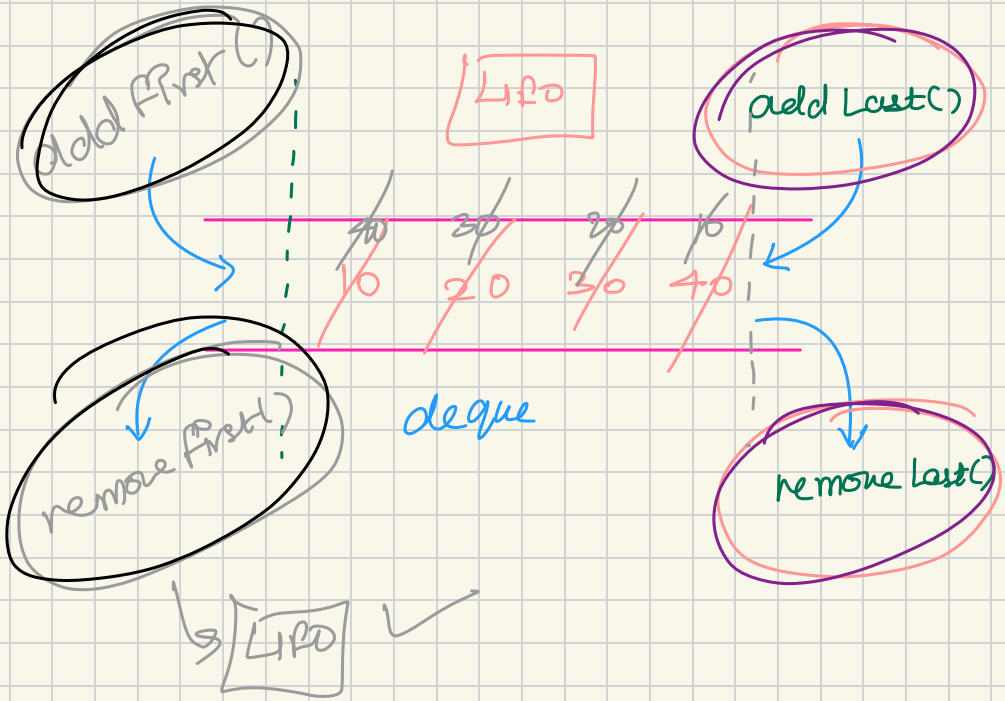
Doubly Ended Linked List

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

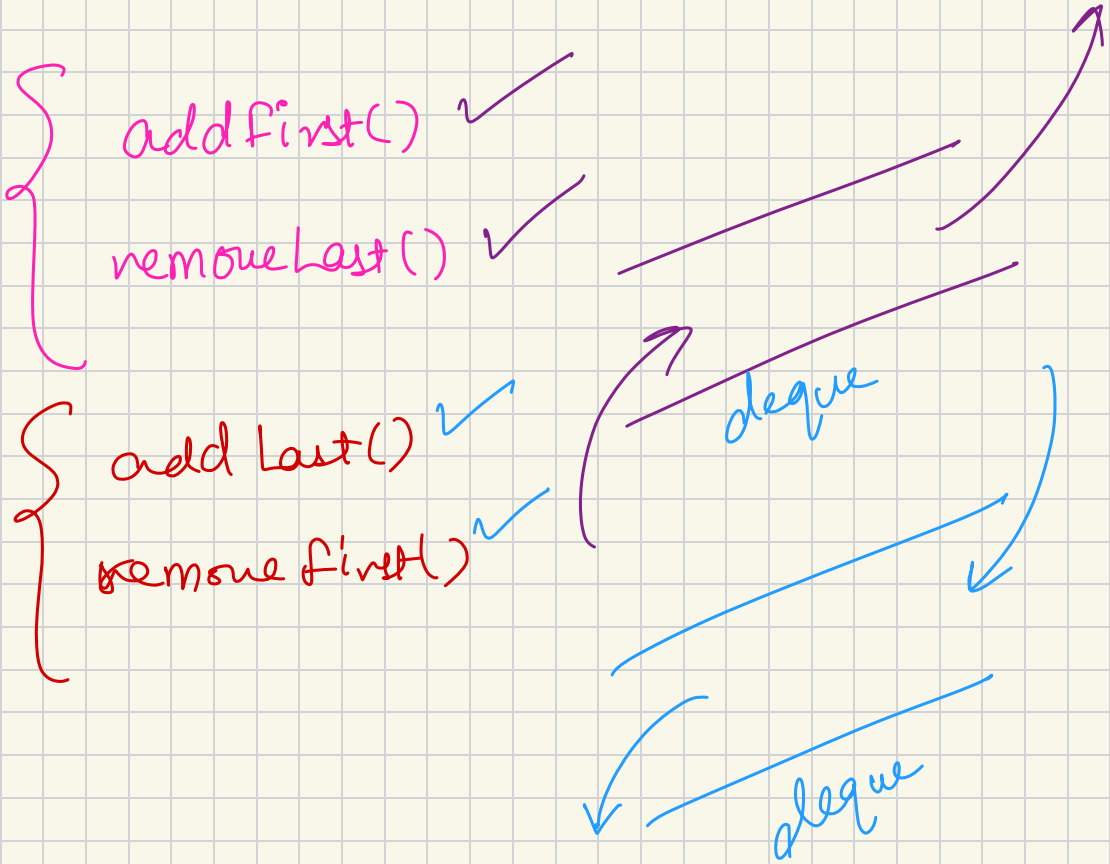
(x + 2y) → Extra Space

Q. Can you implement a stack using a deque?

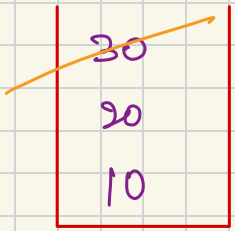




Que. Can you implement a queue using a deque?

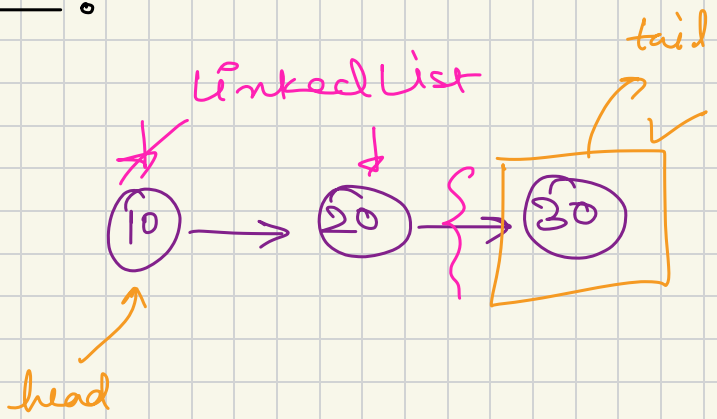


Design a stack using linked list.



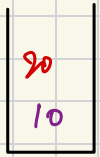
Stack

push(10) ✓
push(20) ✓
push(20) ✓
peek()
pop() → O(N)

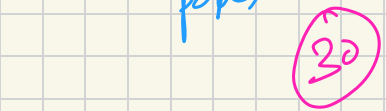


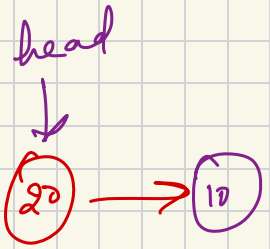
push() → addLastLL()
peek() → tail.data
pop() → removeLastLL()

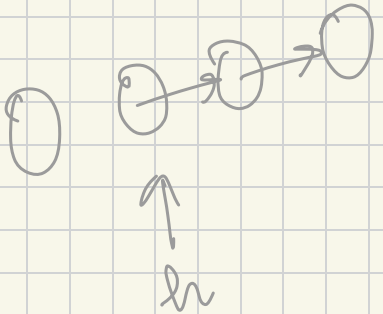
TC: O(N) ✗



push(10)
push(20)
push(30)
peek()
pop()





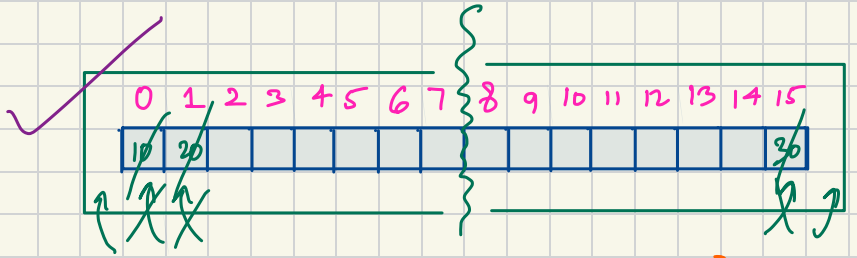


push() → addFirstLL TC: O(1)

peek() → head.data

pop() → removeFirstLL TC: O(1)

implement 2 stacks using Array.

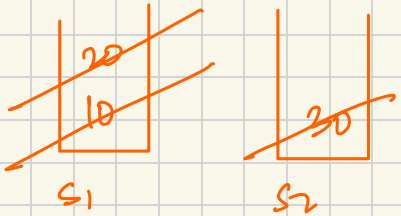


agree? ✓

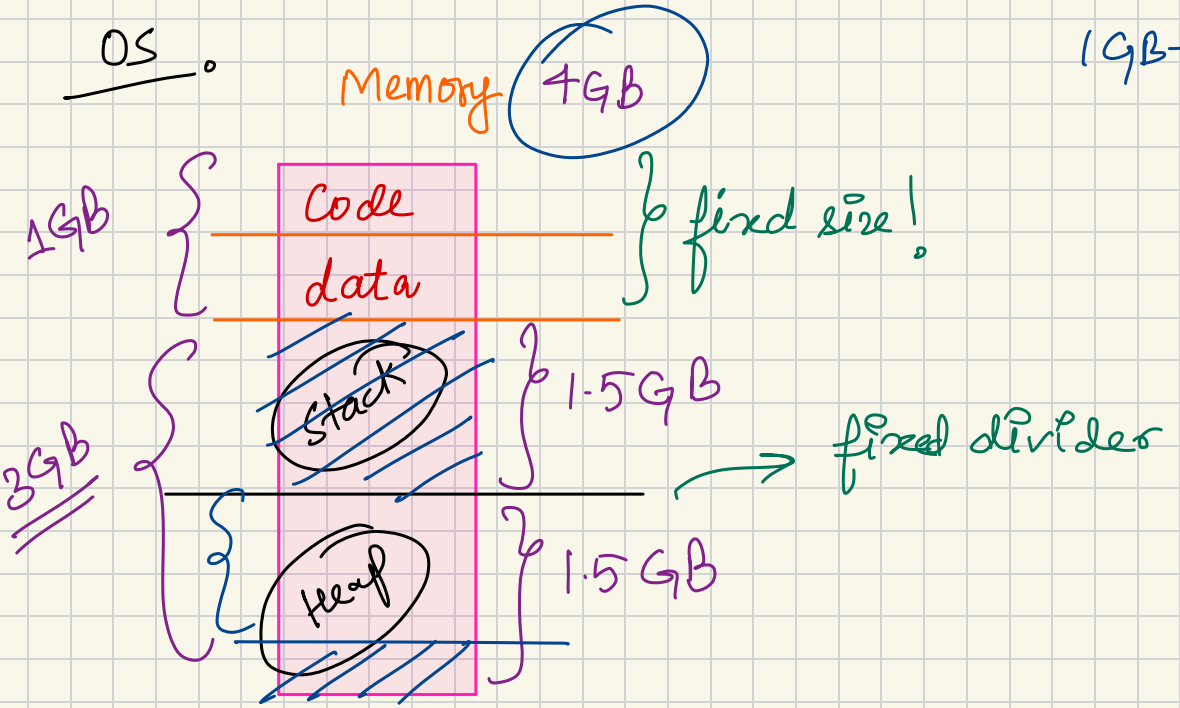
yes work fine!

(I/P) S1.push(10), S1.push(20), S2.push(30), S1.pop(), S1.pop()

S2.pop()



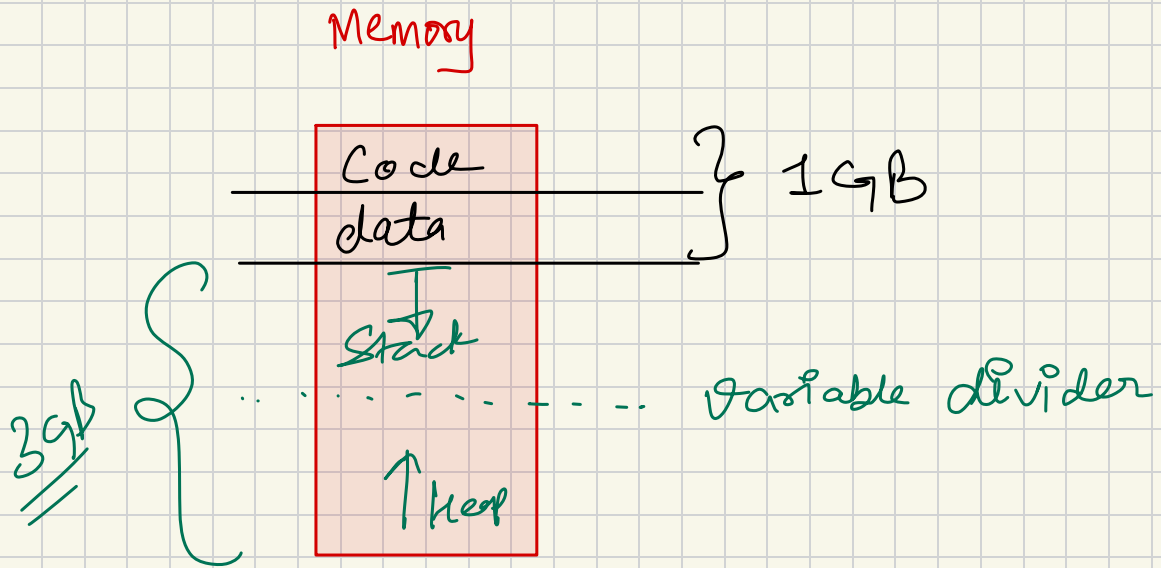
OS.

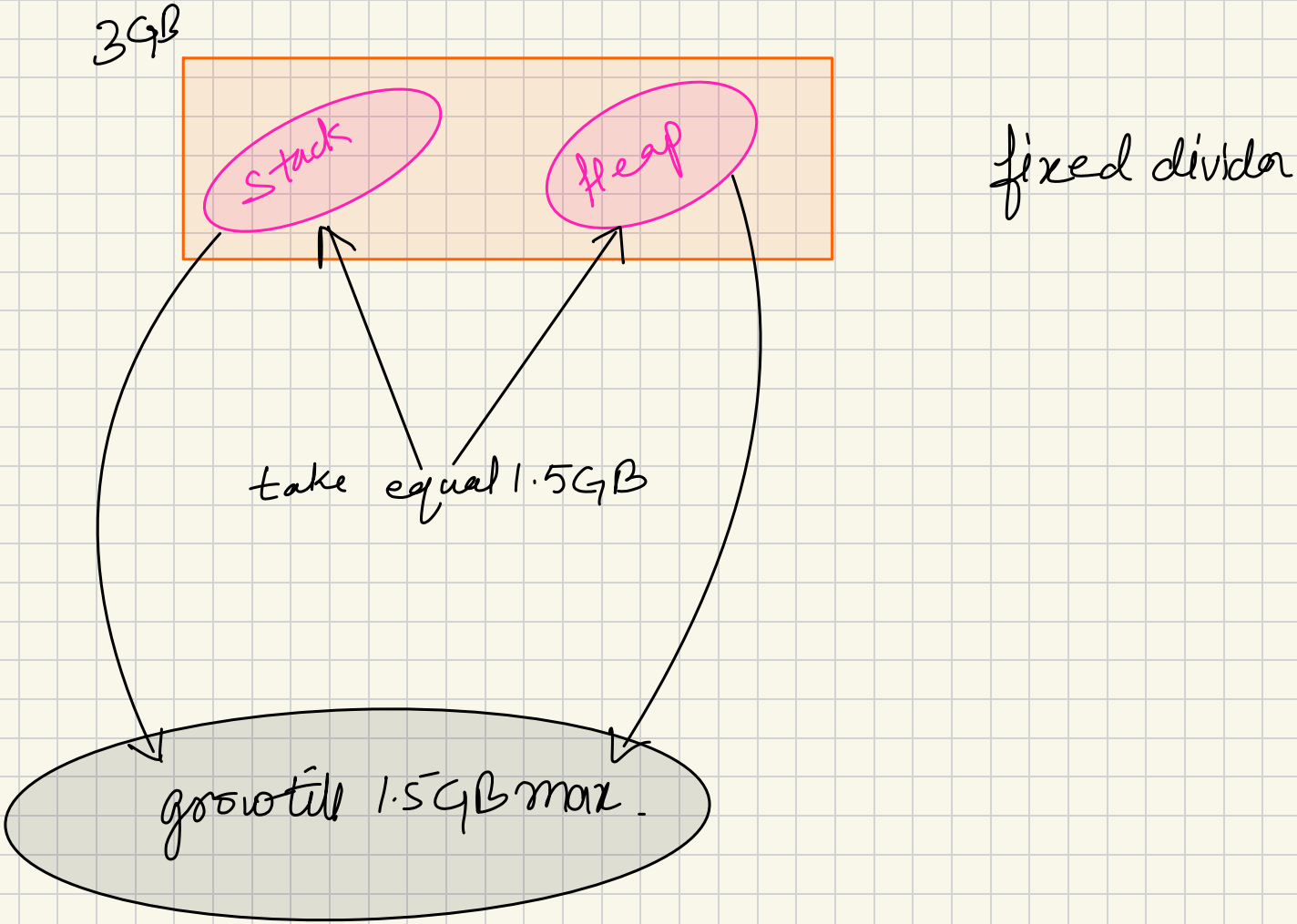


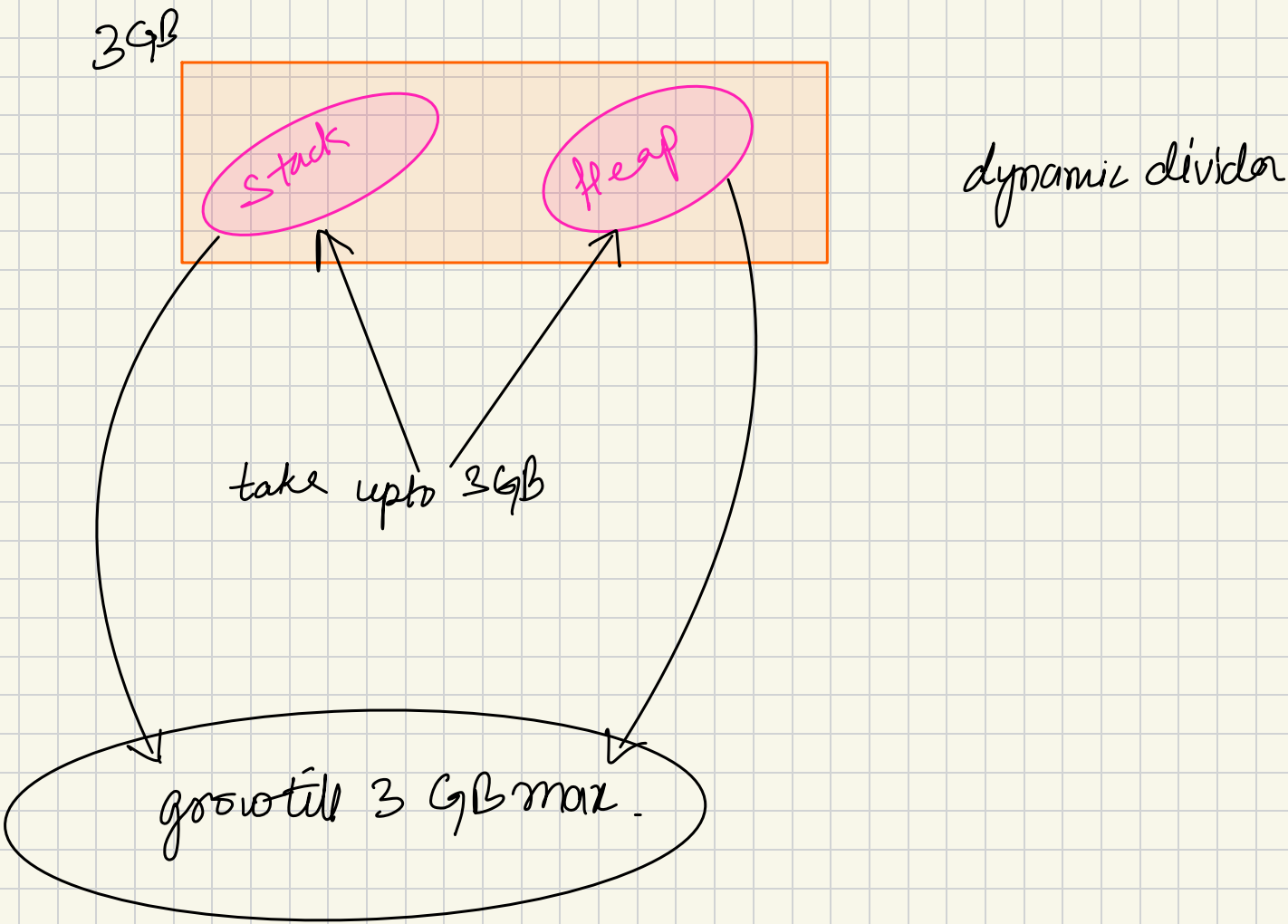
1GB + 1.5GB + 0.5GB

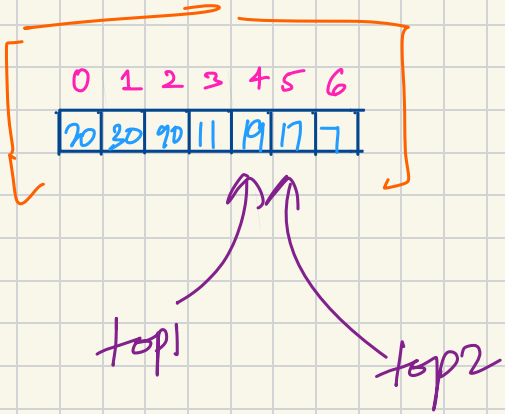
= 3GB

(Crash)!


Memory
Code
data
1GB
Stack
3GB
variable divider
Heap


3GB
stack
heap
fixed divider
take equal 1.5GB
growth 1.5GB max.

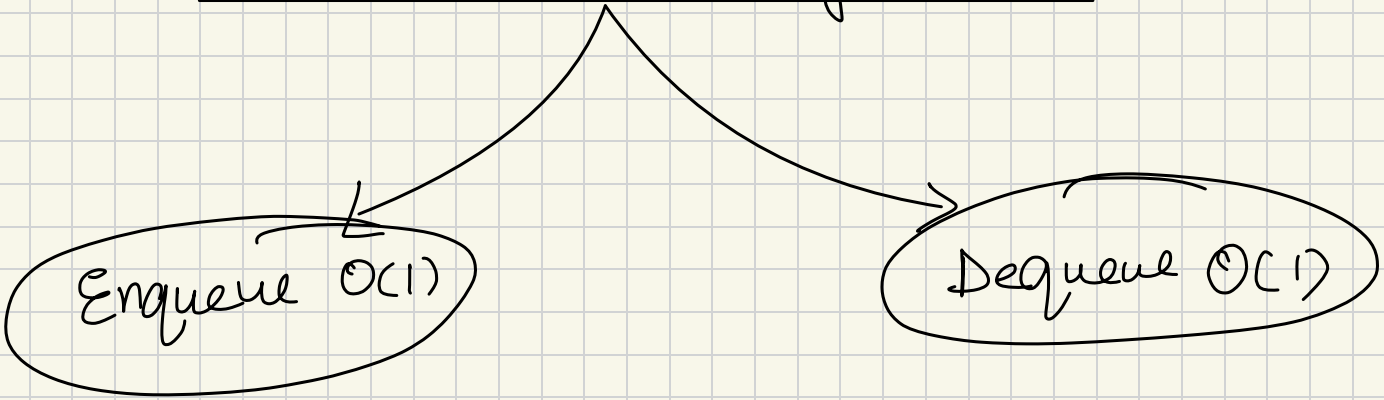

3GB
stack
heap
dynamic divider
take upto 3GB
protected 3 GB max.



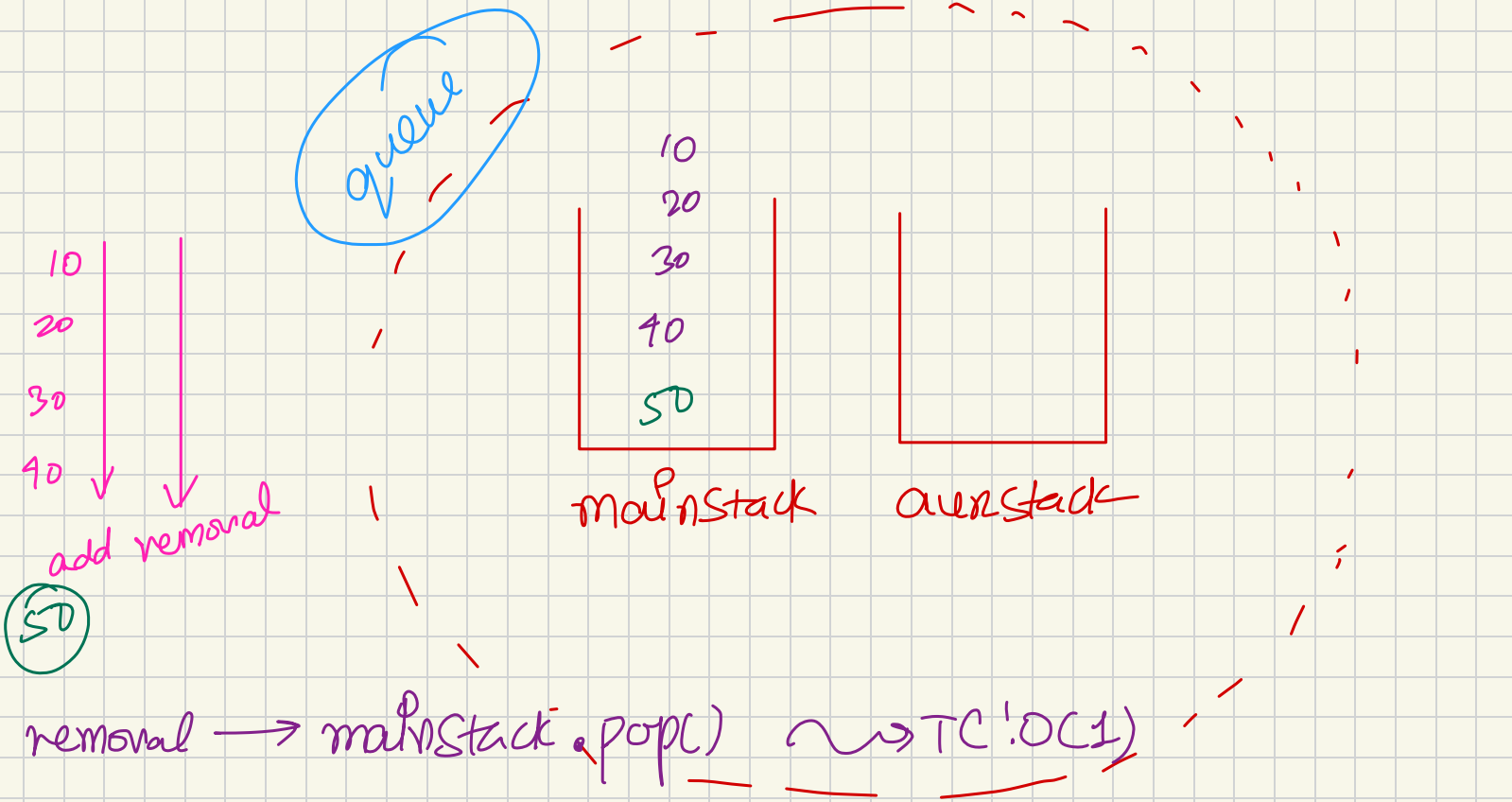
S1. push(20)
S1. push(30)
S1 push(90)
S2. push(7)
S1 push(11)
S2. push(17)
S1. push(19)
S2. push(11)

(reject)

Implement Queue using 2 stacks



Implement Queue using 2 stacks where Dequeue O(1)



removal → mainStack.pop() ⇝ TC: O(1)