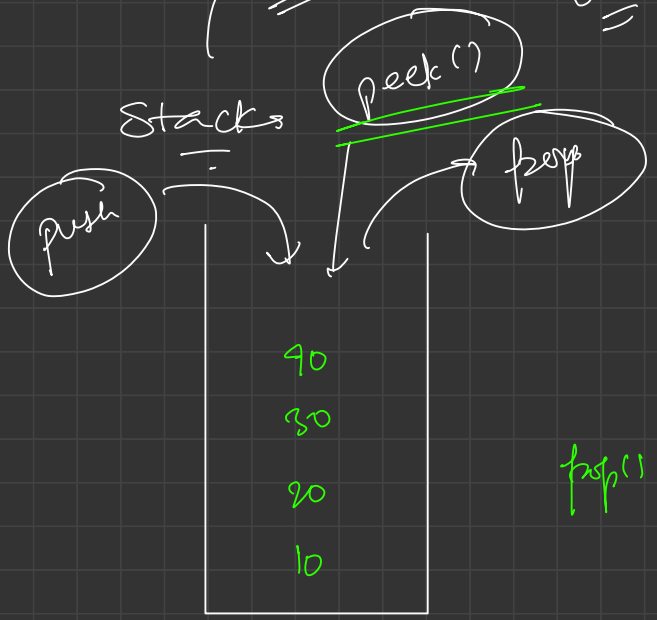
Queues

LIFO

TC: O(1)

Stacks



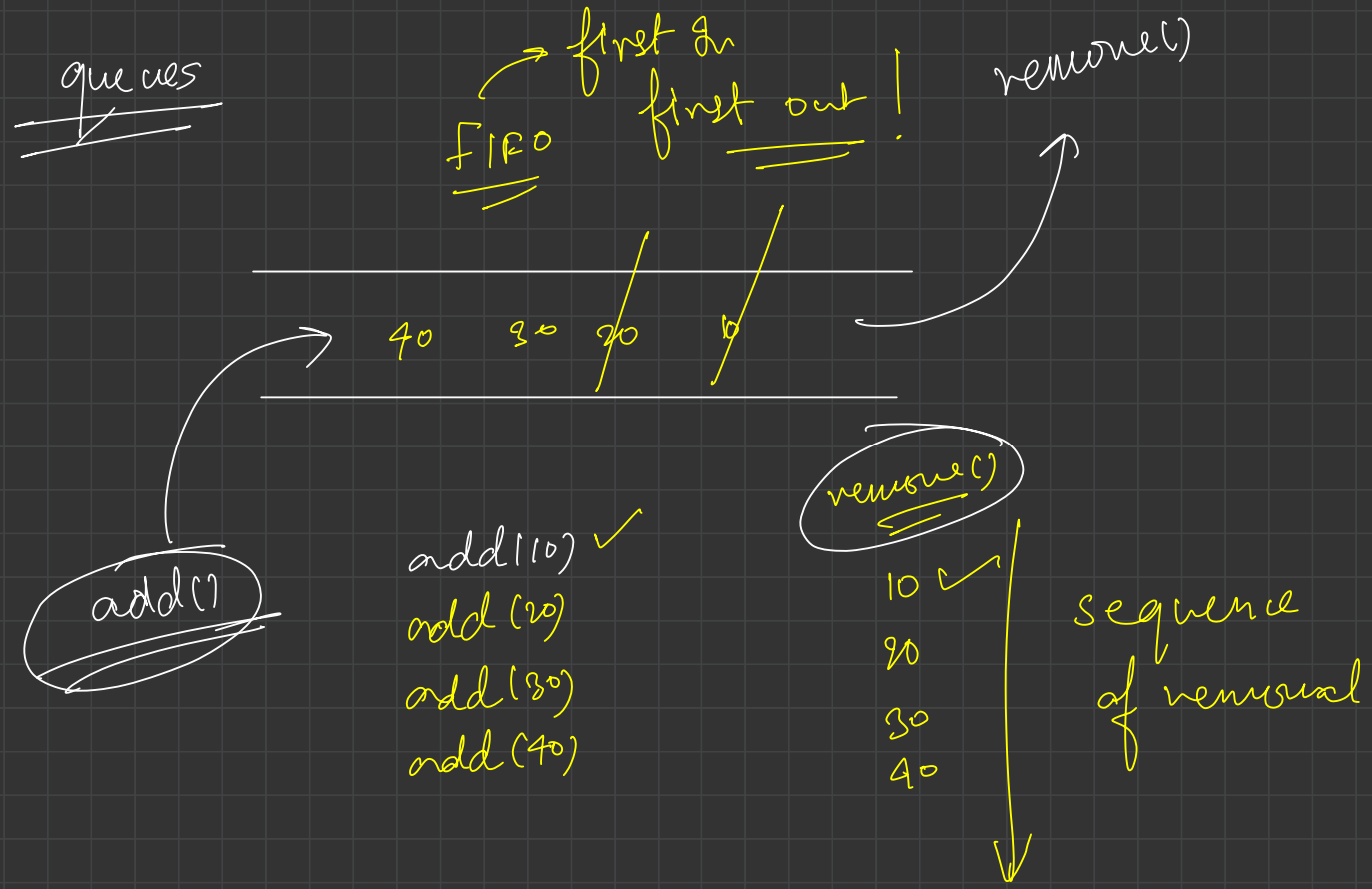
Stacks

eg
push(10)
push(20)
push(30)
push(40)

Order of pop

pop()

40
30
20
10


queues
FIFO
first in
first out !
remove()
40
30
20
10
add()
add(10)
add(20)
add(30)
add(40)
remove()
10
20
30
40
sequence
of removal

enqueue → Adding a element to queue

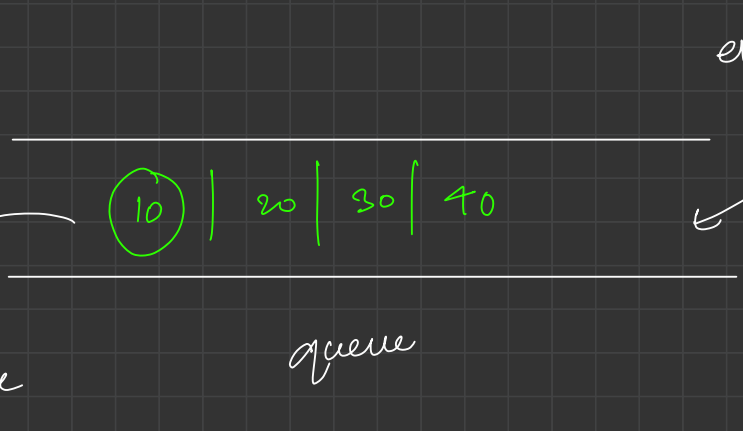
TC: O(1)

dequeue → Removal of a element from a queue

TC: O(1)

Front() TC: O(1)

tells the front ele.



queue

↳ linear data Structures.

↳ follows FIFO (first in, first out)

Methods

enqueue, dequeue, front(), size()

enqueue → add()

dequeue → remove()

arrays

Stacks

ArrayList

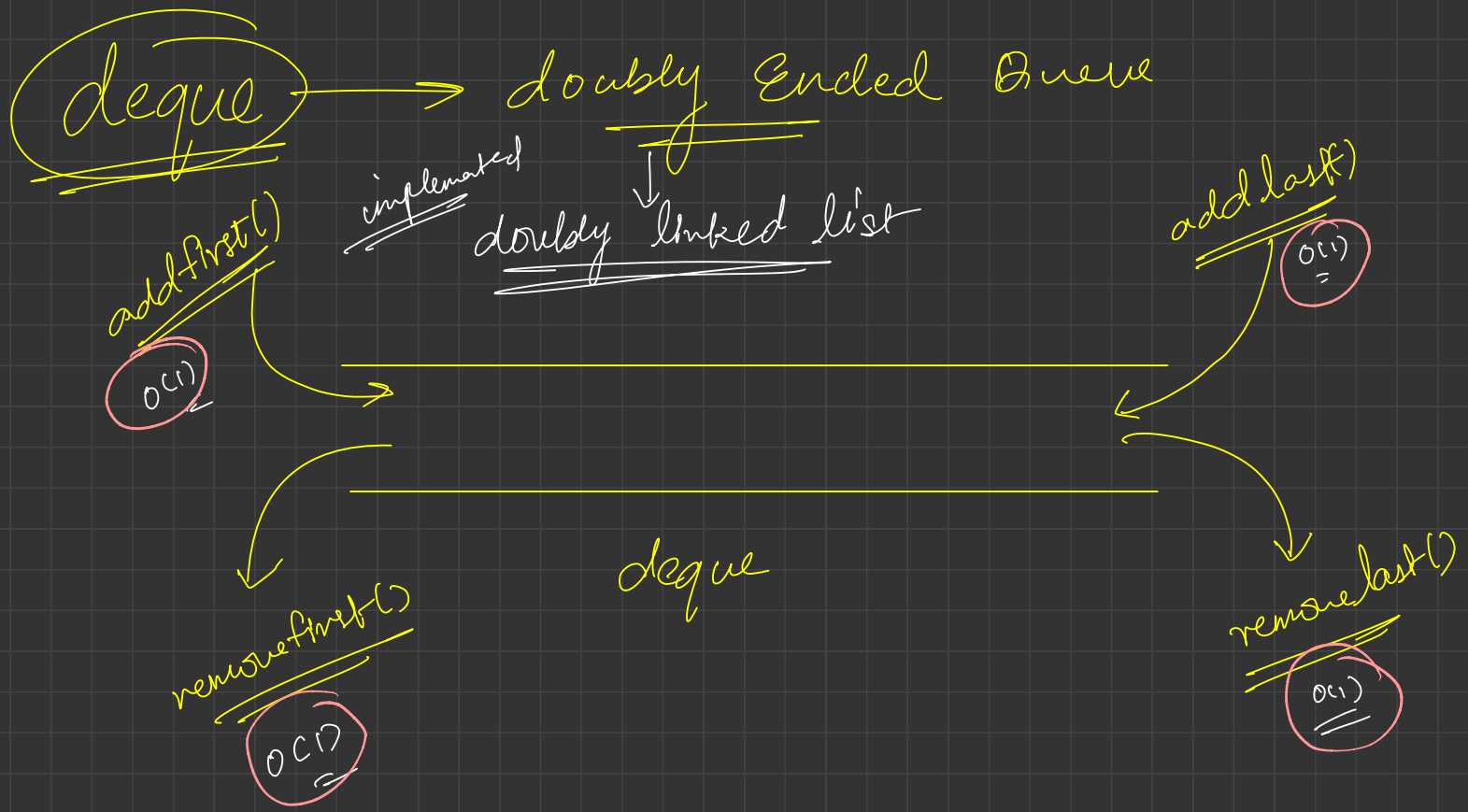
linkedlist

queue

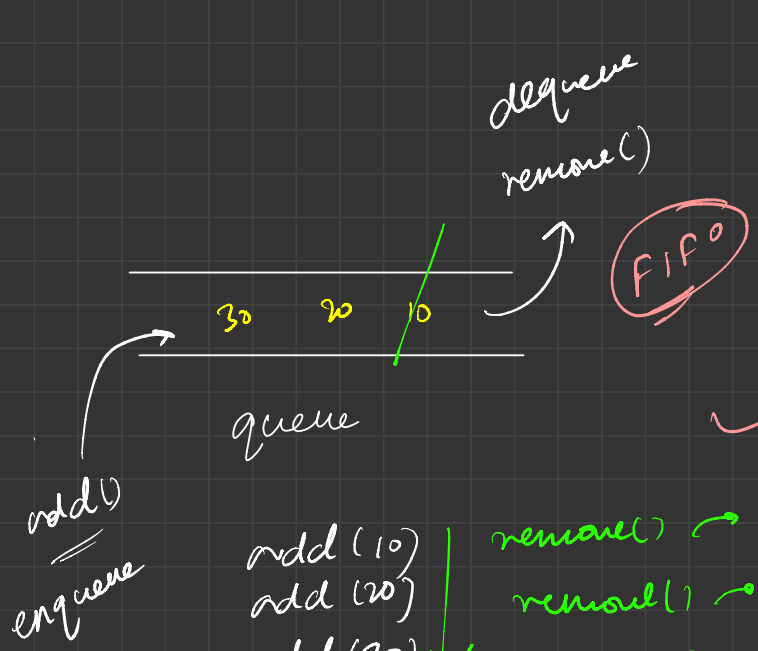
BFS Algo!

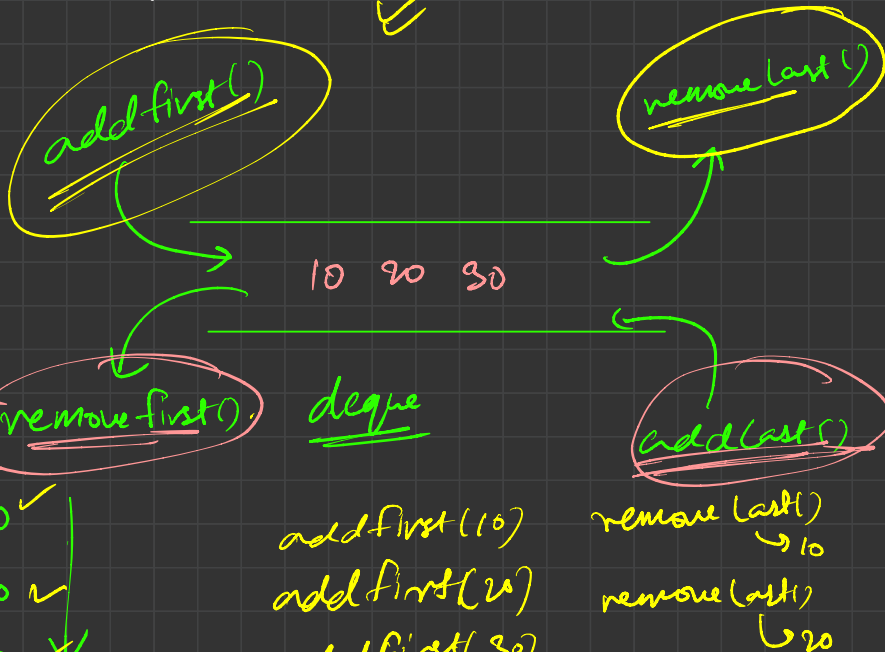
{ Breath first search }

linear data Structures



Q Can you implement a queue using a deque?





to implement a que using deque

addFirst() & removeLast()

OR

addLast() & removeFirst()

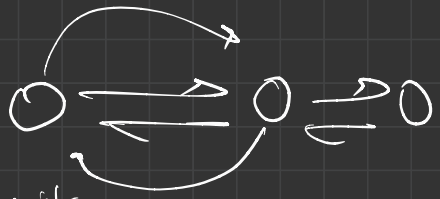
this follows FIFO

hence implemented!

Why deque takes more memory?

queue → implemented using linked list!

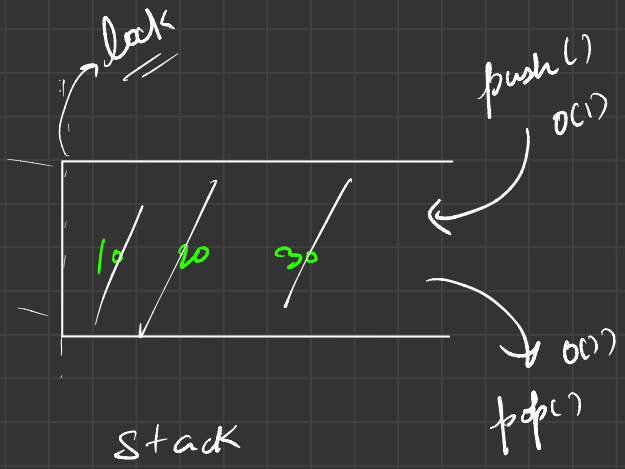
deque → implemented using dLL!

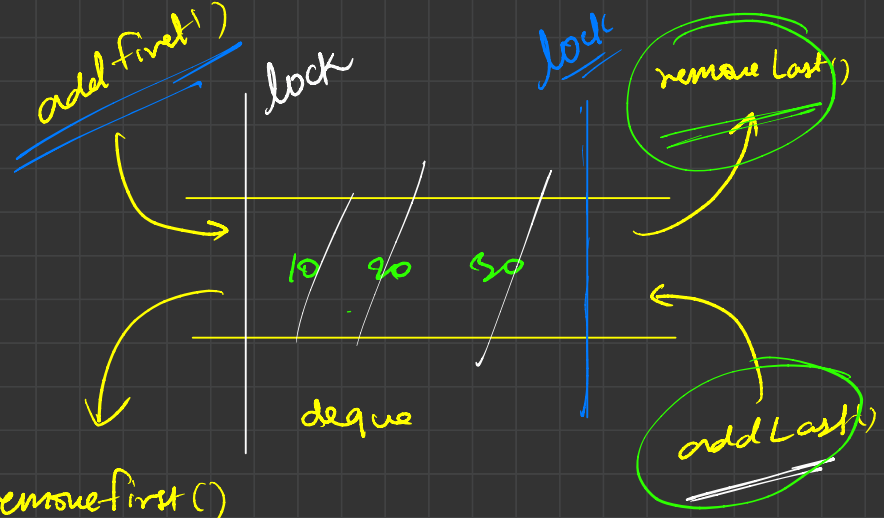


Linked list

doubly linked list

Can you implement Stack using a deque?





Stack

push
10
20
30

pop()
30
20
10

removeFirst()

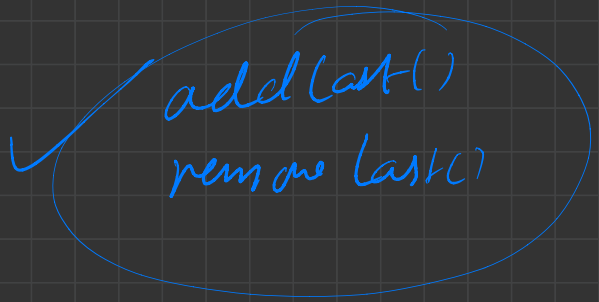
addLast()
10
20
30

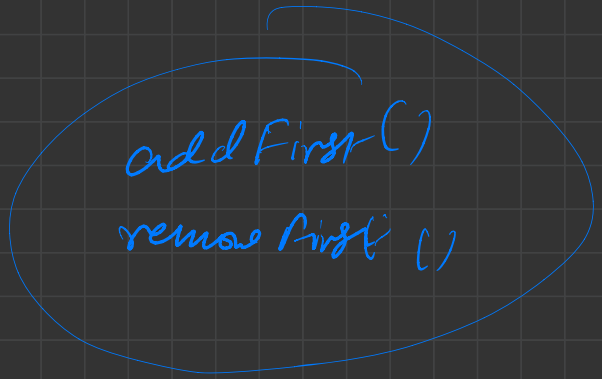
removeLast()
30
20
10

LIFO

Stack can be implemented by deque

using



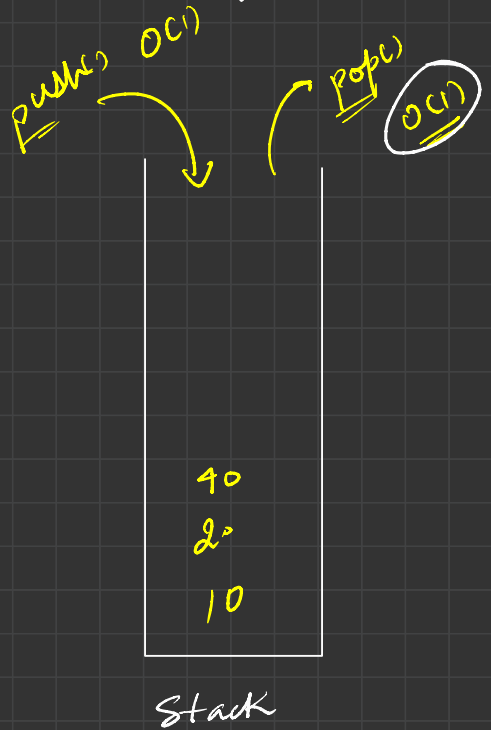


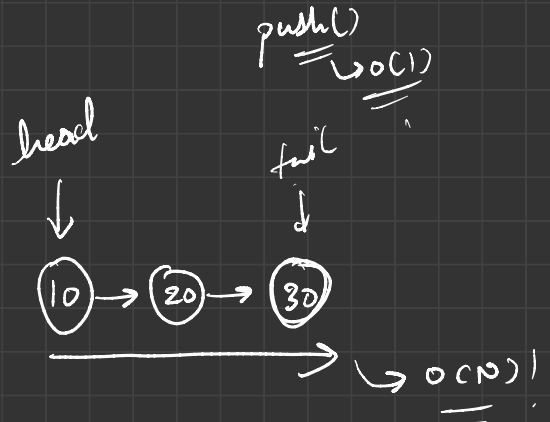
as they follows LIFO

Design a Stack using Linked List

eg:

push(10)
push(20)
push(30)
pop() → 30
push(40)

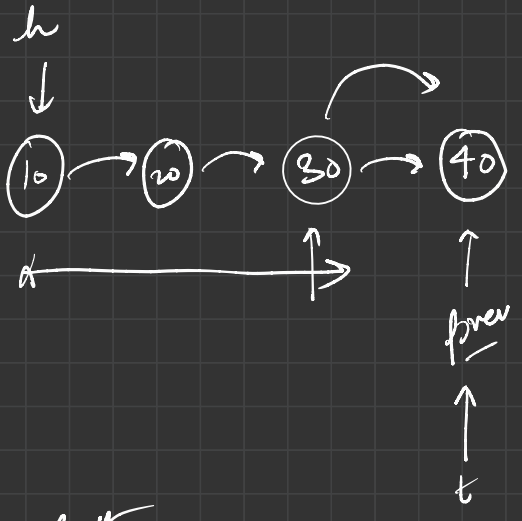




pop()

Linked list → addLast()
removeLast()

are not good!



push()

10
20
30
40
50

pop()

pop()

we can't implement stack using ll

using addLast() & operation of ll
removeLast()

push()

10

20

30

pop()

```
{ Node node = new Node(x);
  node.next = head;
  head = node; }
```

push
O(1)

```
{ Node temp = head.next;
  head.next = null;
  head = temp; }
```

removal O(1)
O(1)

to implement Stack using:

1. addFirst()
2. removeFirst()

```
public void push(int x)
{
    //Complete the function
    Node node = new Node(x);
    if (top == null)
    {
        top = node;
    }
    else
    {
        node.link = top;
        top = node;
    }
}
```

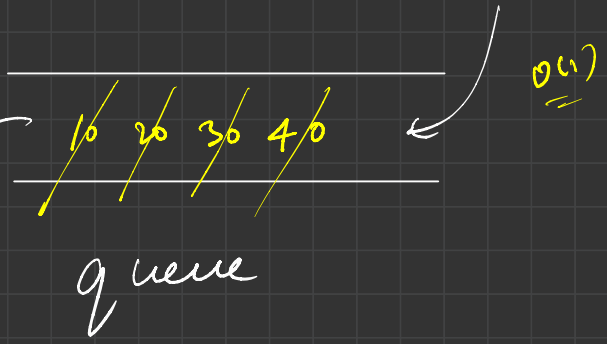
```
public void pop()
{
    //Complete the function
    if (top == null)
    {
        return;
    }

    Node temp = top.link;
    top.link = null;
    top = temp;
}
```

push(10)
push(20)
push(30)

top
↓
null

pop()
pop()
pop()

Implement a queue using Linked list



add()

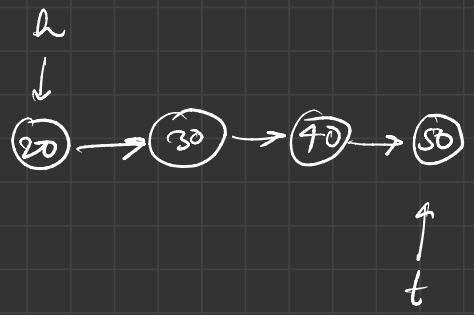
10
20
30
40

FIFO

remove()

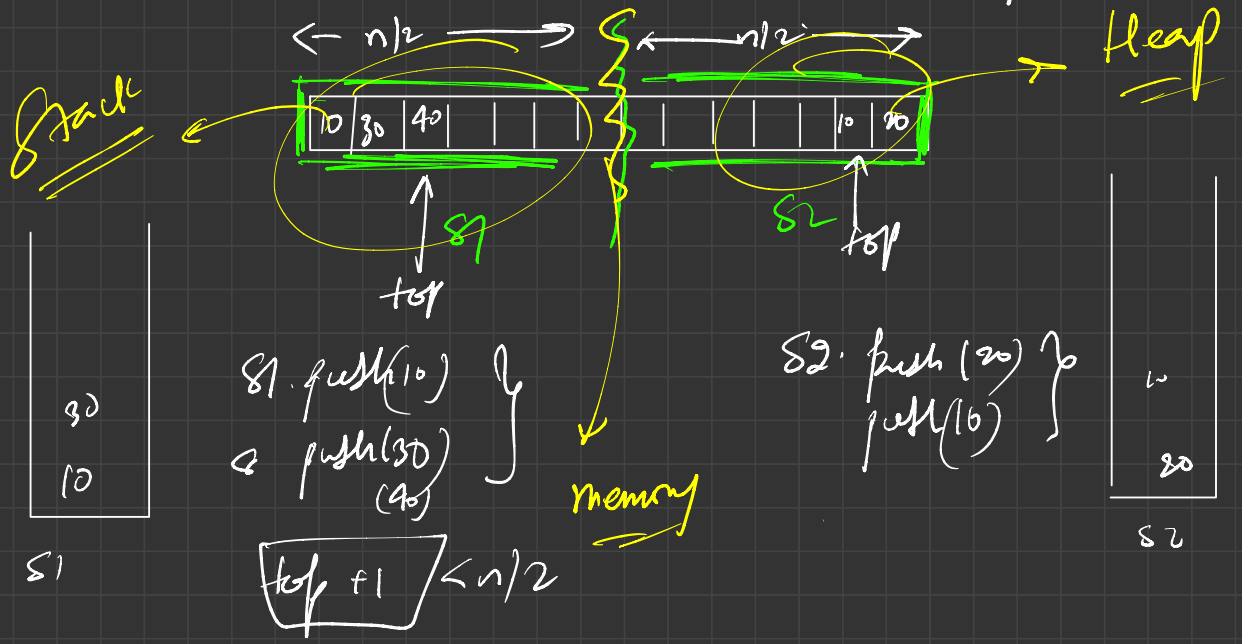
10 ✓
20
30
40

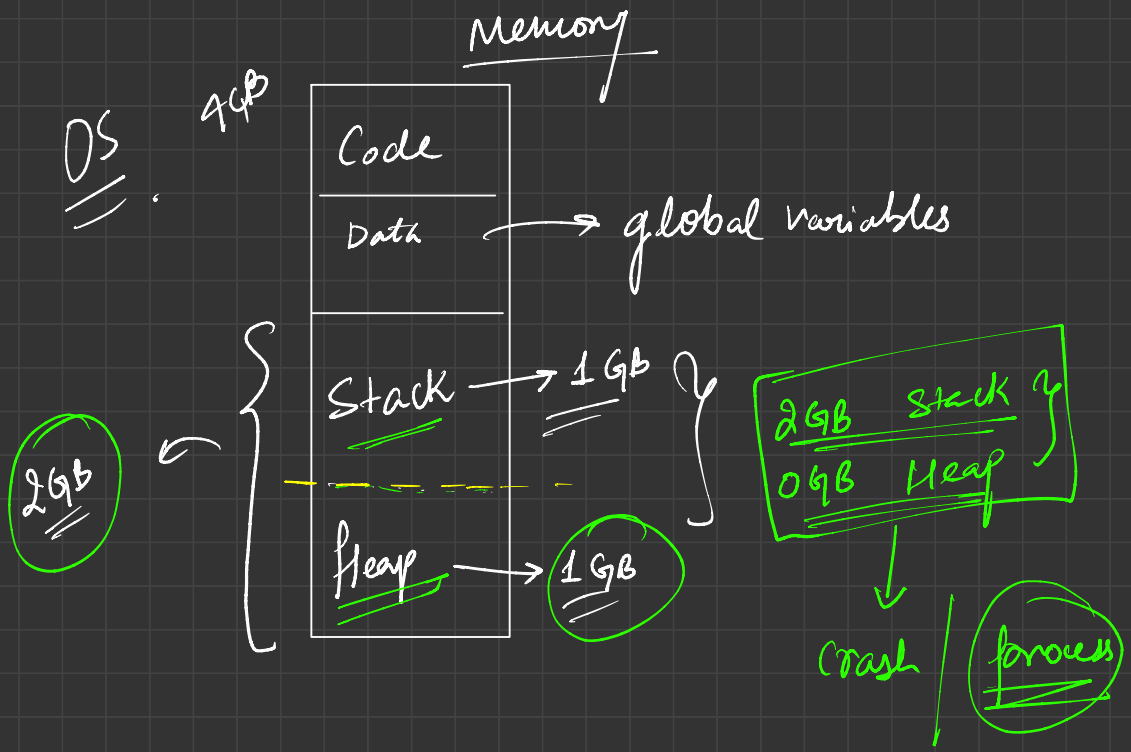
```
{ tail.next = node;
  tail = node; }
```

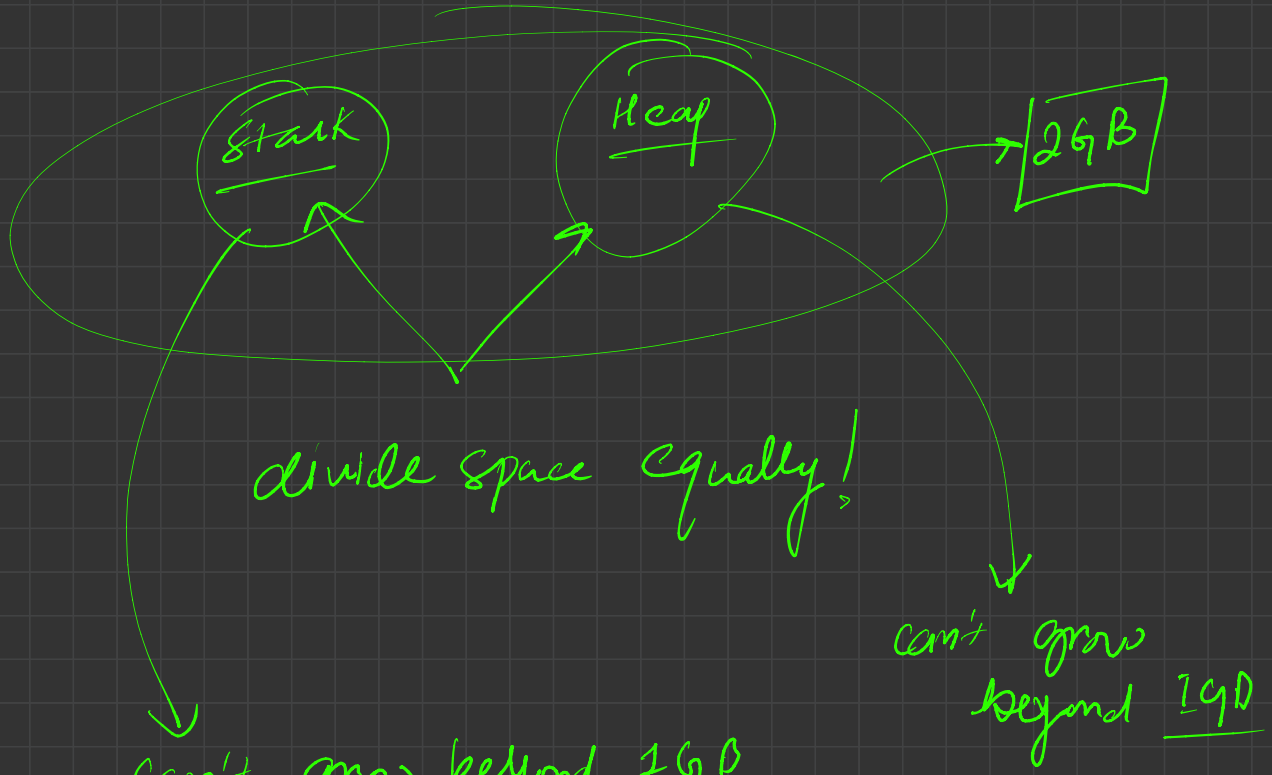


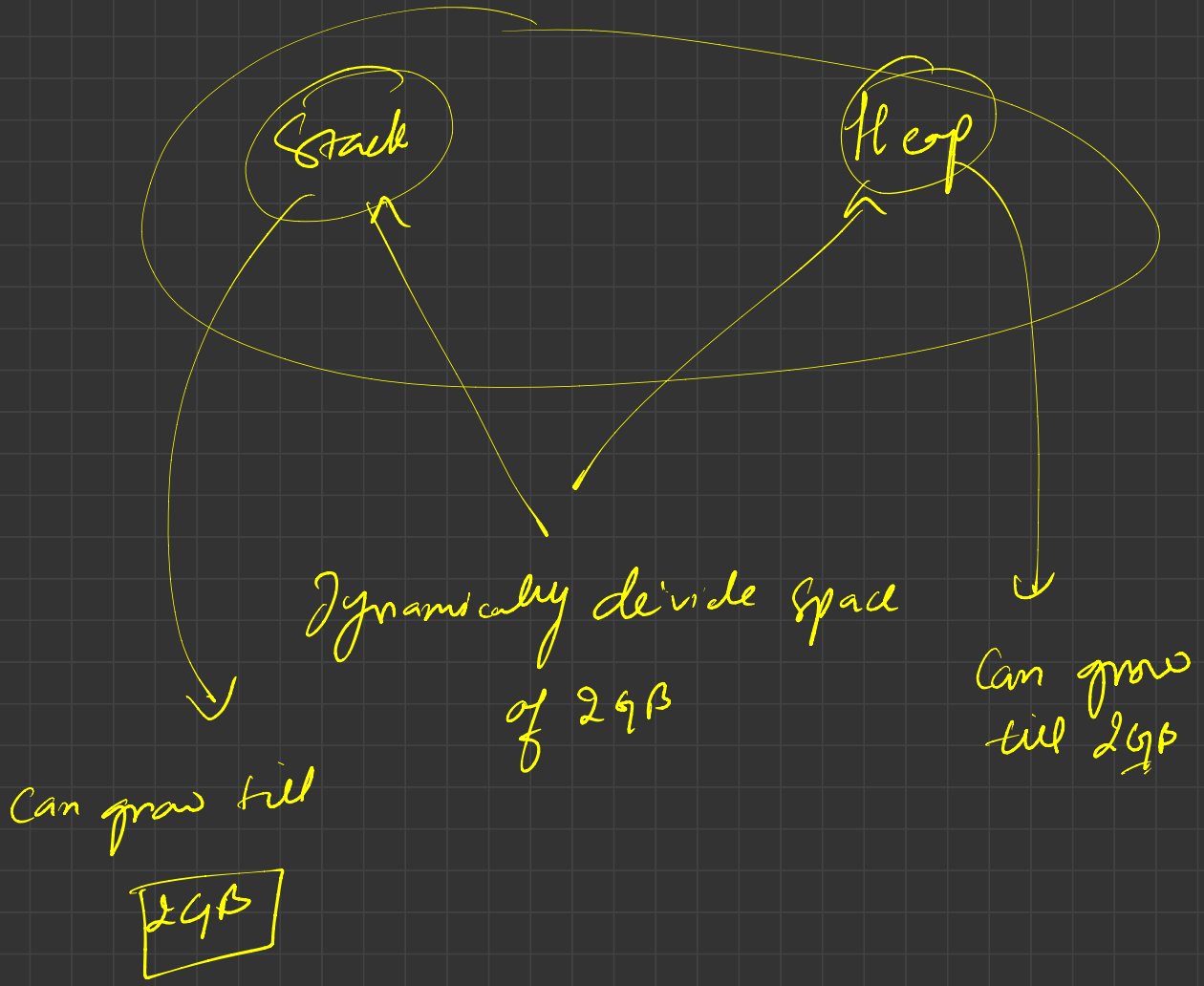
{ addLast()
removeFront() }

Implement 2 stacks using an array!



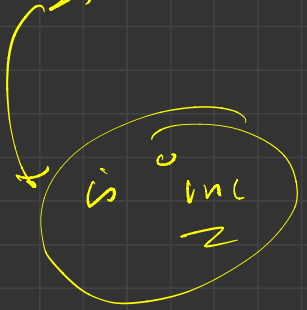

Memory
OS
4GB
Code
Data
global variables
Stack
1 GB
Heap
1 GB
2GB
2GB stack
0GB Heap
Crash
process

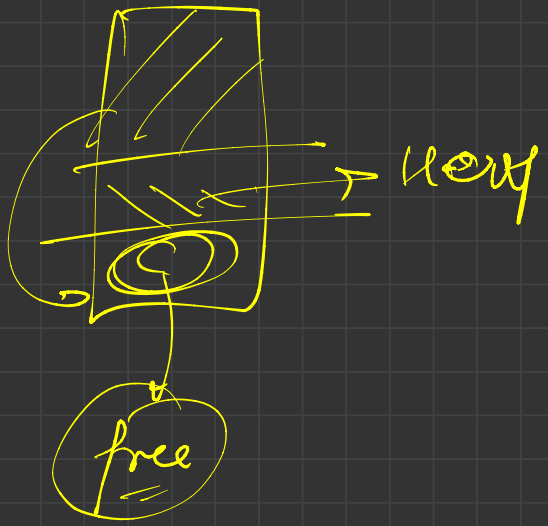

Stack
Heap
2GB
divide space equally!
can't grow beyond 1GB
can't grow beyond 1GB

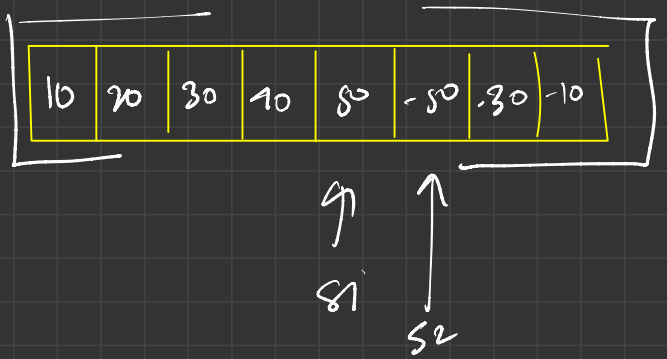

Stack
Heap
Dynamically divide space of 2GB
Can grow till 2GB
Can grow till
2GB

Heap 0.5 GB

Stack 1 GB







S1

push(10)
push(20)
push(30)
pop(40)
pop(50)

push(30)

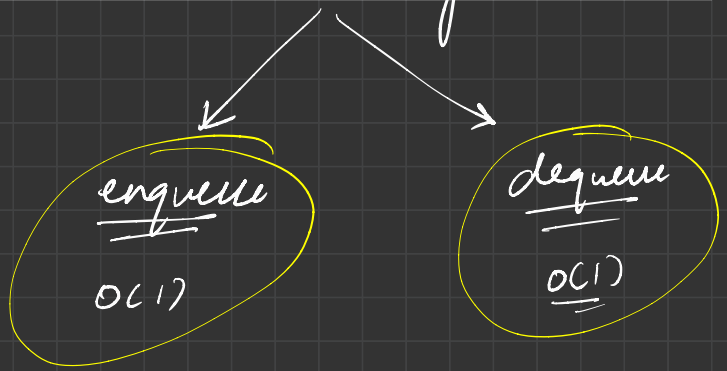
S2

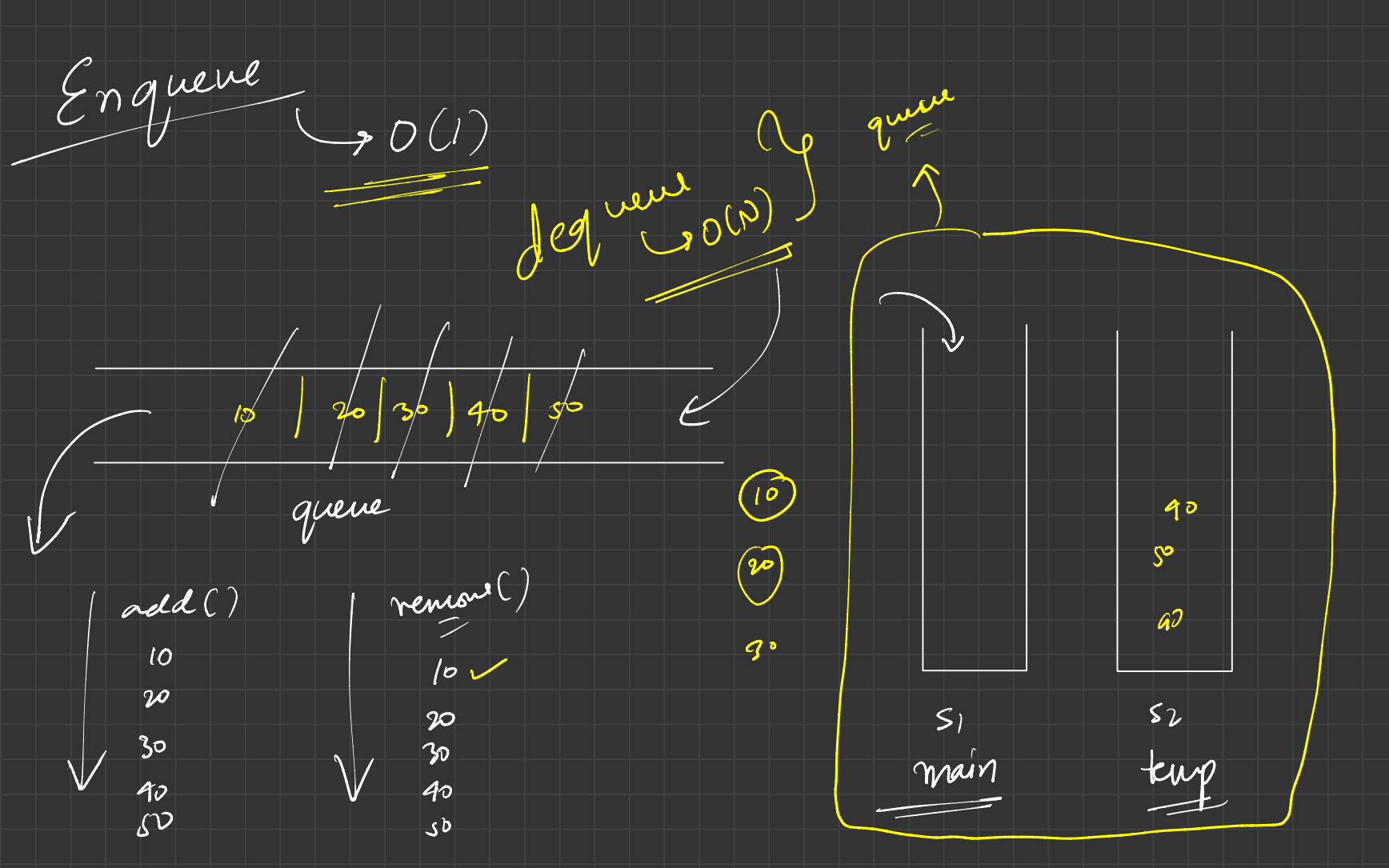
push(-10)
push(-30)
push(-50)

Crash

S1+1 == S2
OR
S2+1 == S1

Implement Queue using 2 - Stacks

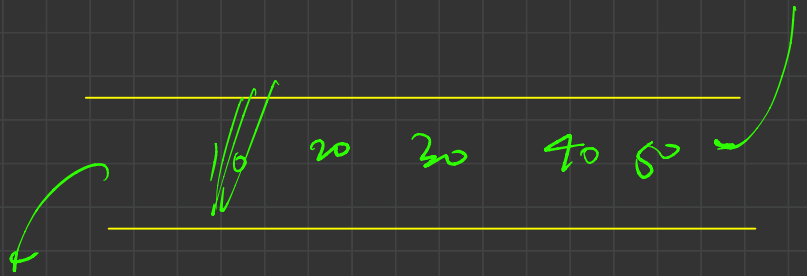


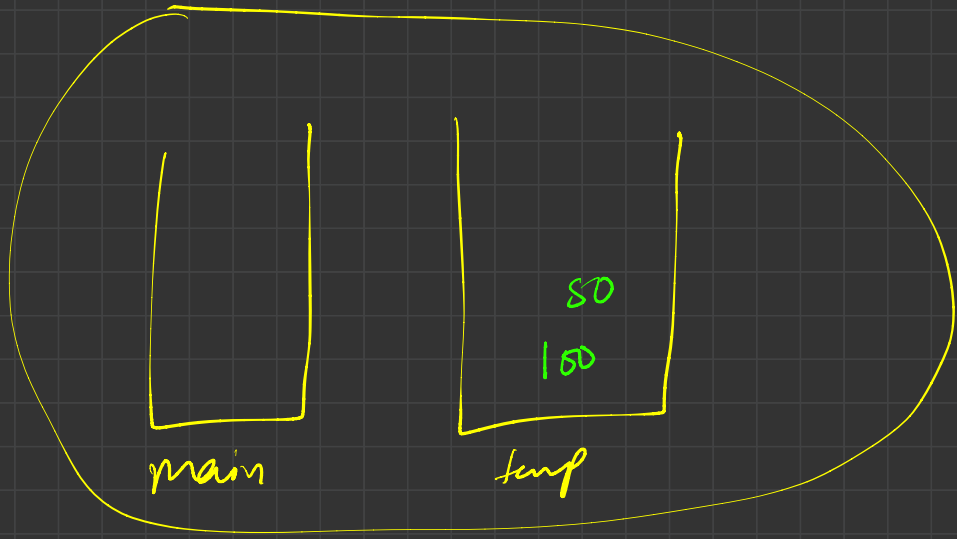

Enqueue
O(1)
dequeue
O(n)
queue
10 | 20 | 30 | 40 | 50
queue
10
20
30
add()
10
20
30
40
50
remove()
10
20
30
40
50
40
50
40
S1
main
S2
temp

push(10)
push(20)
push(30)
pop()
push(90)
push(80)
pop()
pop()

push(100)

10
20
30
40





Dequeue O(1)

10, 20, 30, 40, 50