Q: Try to Implement Queues using Stacks.
(Time Complexity need not to be O(1))

Q) → Implement { Stacks using Queues] ✓
Queues using Stacks

we donot need a highly efficient soln

} We donot need to maintain O(1) complexity

Queues Using Stacks

fifo ← → Lifo

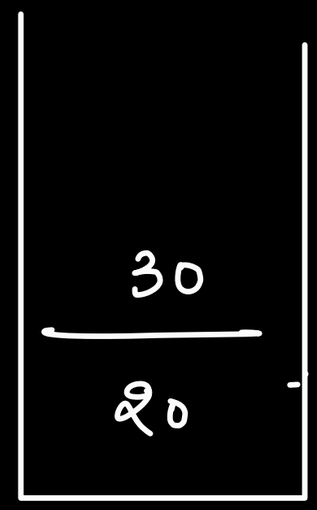
enqueue efficient

dequeue efficient

enque efficient

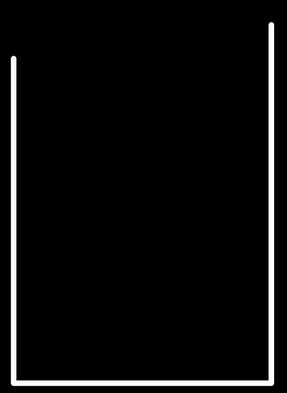
↳ enqueue (10)
→ enqueue (20)
enqueue (30)
} push → O(1)

dqueue → removal at Bottom → O(n)



→ bottom

internal stack



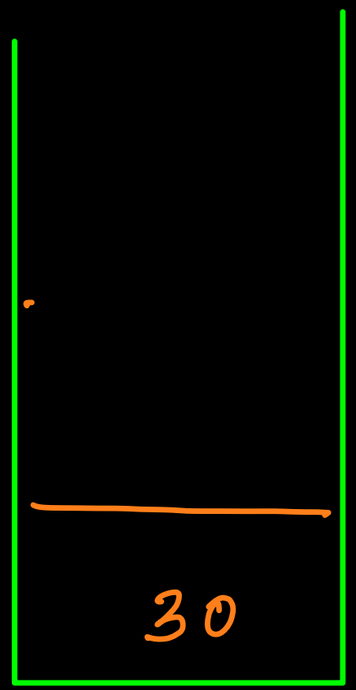
temp

temp.pop()

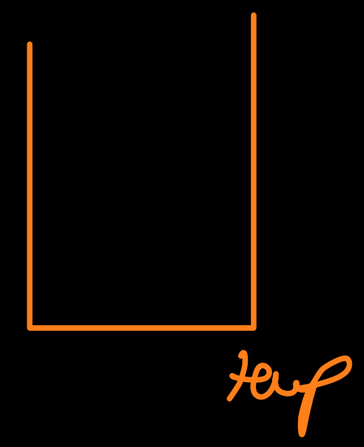
deque ~~Queue~~

enqueue(10)
enqueue(20)
enqueue(30)
} insert At Bottom → O(n)

dequeue() → pop()
↳ O(1)



↳ internal Stack



Stacks Using Queues

LIFO ← ——— → FIFO

push efficient

pop efficient

keep track of the element after whose removal queue becomes empty

push efficient ⇒

front

push(10)
push(20)
push(30)
} enqueue → O(1)

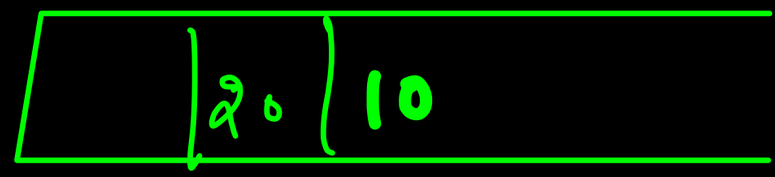
pop()
↳ O(n)
dequeue

pop efficient

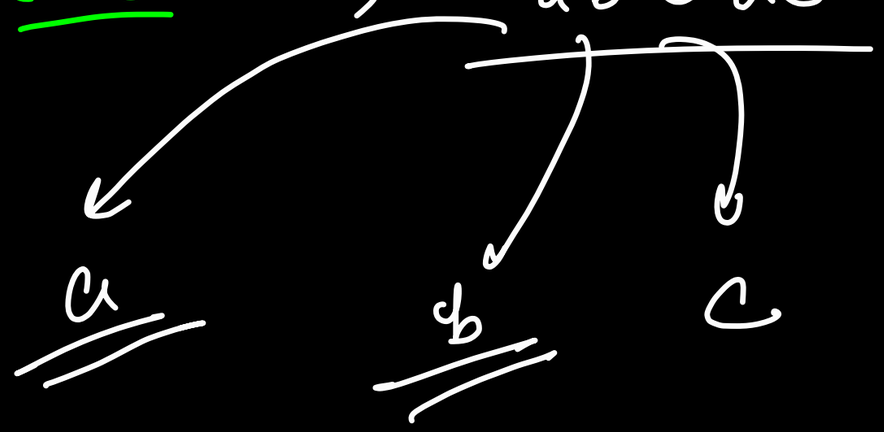
push (10)
push (20)
push (30)
pop()

} O(n)

pop() = O(1)

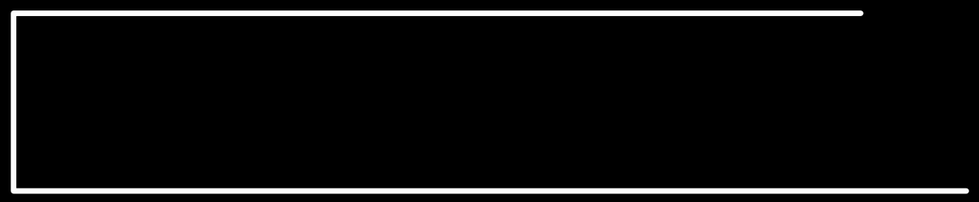


↳ Stream of chars → a b c de



freq map

a b c a a b d d c f c → queue



the moment freq of a char increases from 1, it can never be the ans.

if we have 2 chars with freq 1 then the one that came first will be our answer.

fifo → fcfs

first come first serve

a b c a a b d d c f c

→ O(1) per char

→ O(n)

this a can never be repeated so don't enqueue

{

a : ~~1~~ ~~2~~ 3

b : ~~1~~ 2

c : ~~1~~ ~~2~~ 3

d : ~~1~~ 2

f : 1

}

Count

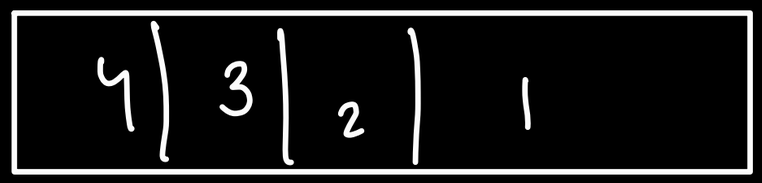
3

ans → a a a b c c c #

f f

→

Q:- How can we reverse a queue??



→ queue

→ O(n) → time
O(n) → space