Queue to Stack

push - Efficient → O(1)

peek , remove → O(n)

size → main.q.size()

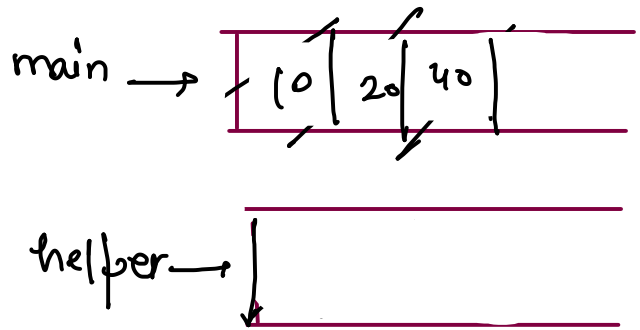
Stack

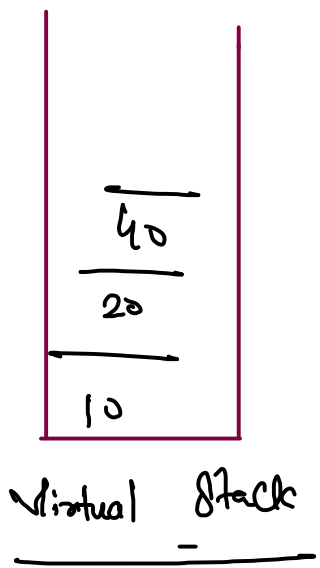
push

pop

peek

size





push → 10 O(1)

push → 20 O(1)

push → 30 O(1)

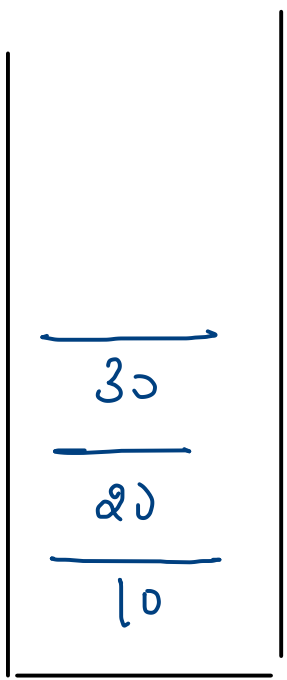
pop() → 30 } n-1 popout and add it helper Que.

push → 40 } nth pop

peek() → 40

Stack	Queue
push → addlast → O(1)	add → addlast
pop → Removelast → O(1)	remove → RemoveFirst
peek → getlast → O(1)	peek → get first
size → size → O(1)	size → size.

add Efficient



helperQ 10 | 20 | 40

mainQ 10 | 20 | 40

push(int val) {

mainQ.add(10);

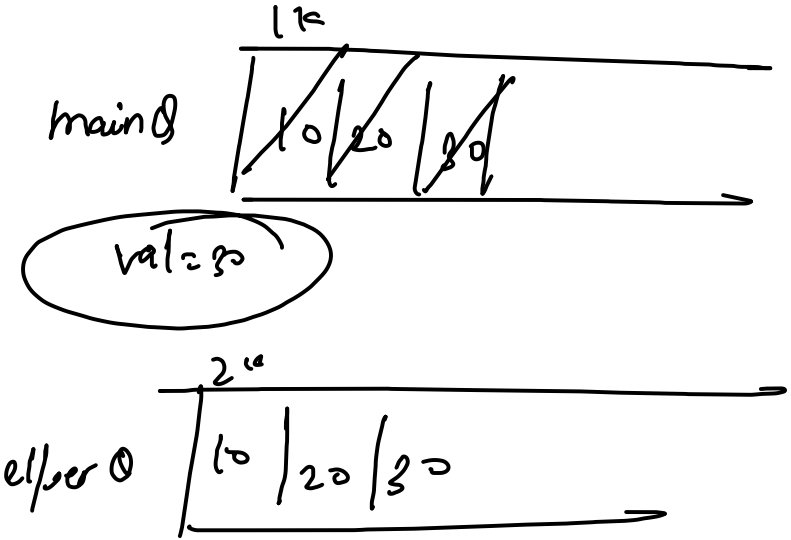
peek →
→ n-1 Element
remove addHelper
get value of
nth Element and
add it in helperQ
change reference.
→ Return value.

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

(Size-1) remove }
from mainQ
and add it in
helper
remove last Element and
return val
change reference of mainQ & helperQ

push →

pop →



peek →

main() = 1K

helper = 2K

a = 10

b = 20

temp = a;

a = b;

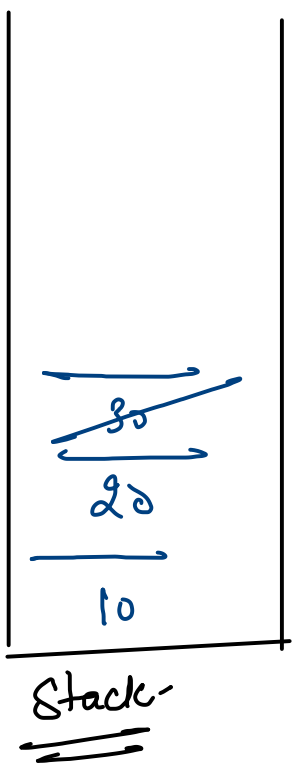
b = temp;

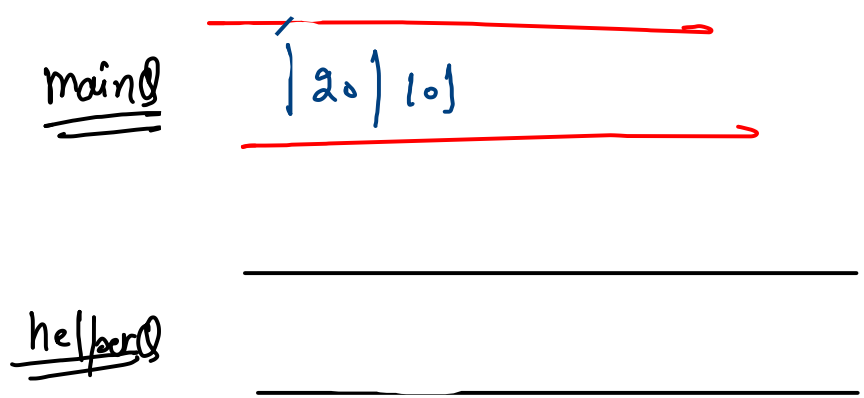
size →

pop Efficient

pop → O(1) , peek() → O(1)

push →





push 10

push 20

push 30

pop() → 30

peek() → 20

size → 2

push() →

remove all Element from main Q and add it in helper Q.

add new Elent in main Q

Fill main Q from helper Q

pop() → mainQ.remove()

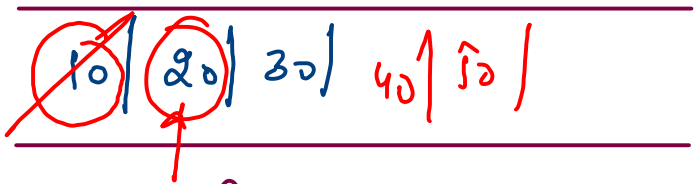
peek() → mainQ.peek()

size → mainQ.size();

Stack to Queue → AddEfficient

add - Efficient — O(1)

remove / peek — O(n)



Queue

add → 10

add → 20

add → 30

remove → 10

add → 40

add → 50

peek() → 20

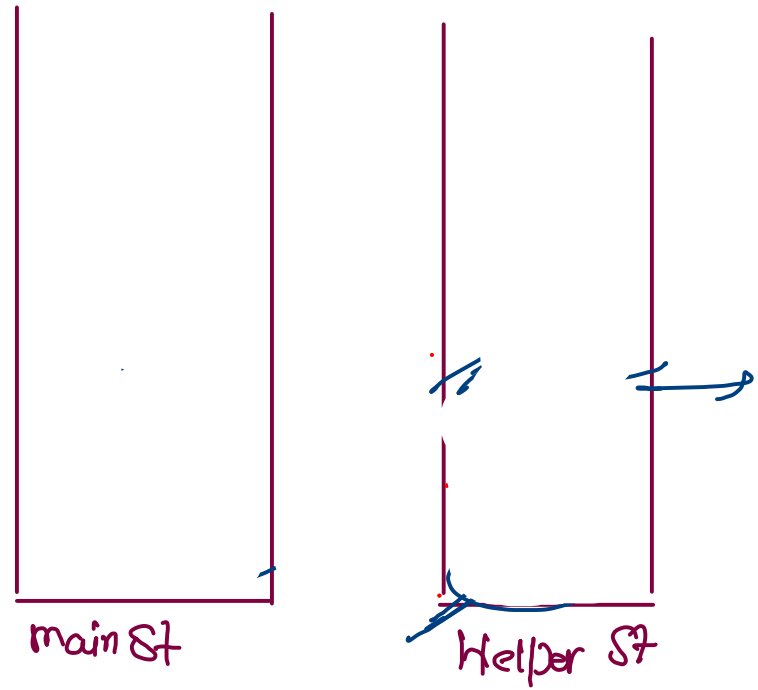
size → mainSt.size

n-1 → mainSt ↓ to helperSt

get LastEle

remove.

return



Queue from Stack: Remove Efficient

10	40	50

push → 10

push → 20

push → 30

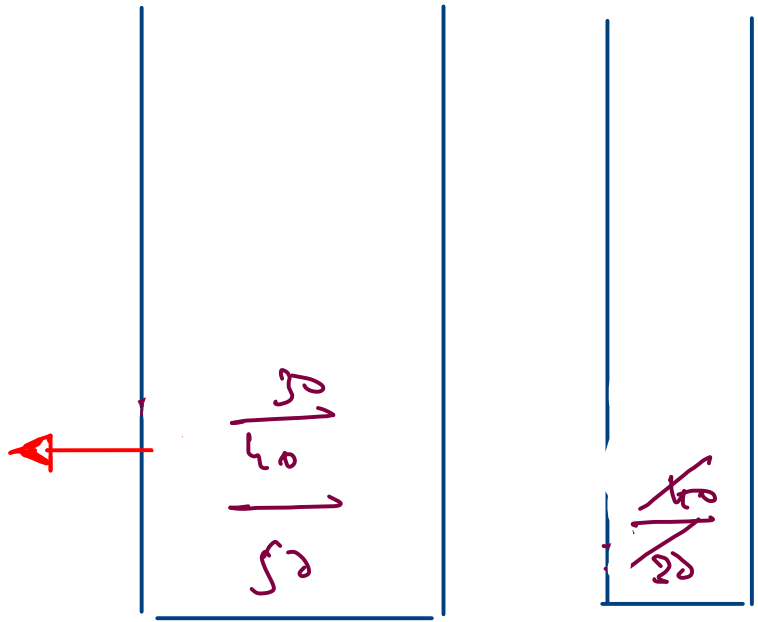
push → 40

peek() → 10 O(1)

pop() → 10 O(1)

pop() → 20 O(1)

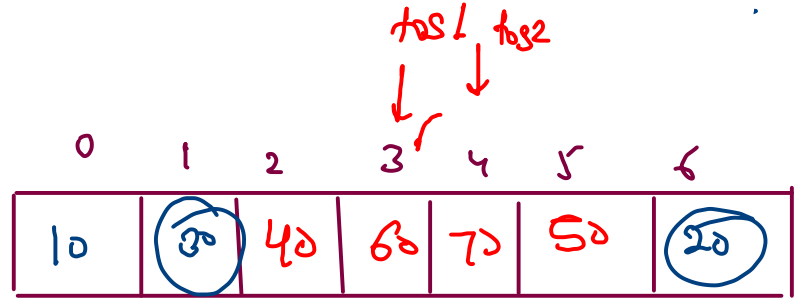
push(50)



cap = 7

push (10, 1)
push (20, 2)
push (30, 1)
push (40, 1)
push (50, 2)
size 1 → 3
size 2 → 2
pop (1) →
pop (2) →
peek (1) → 30
peek (2) → 20

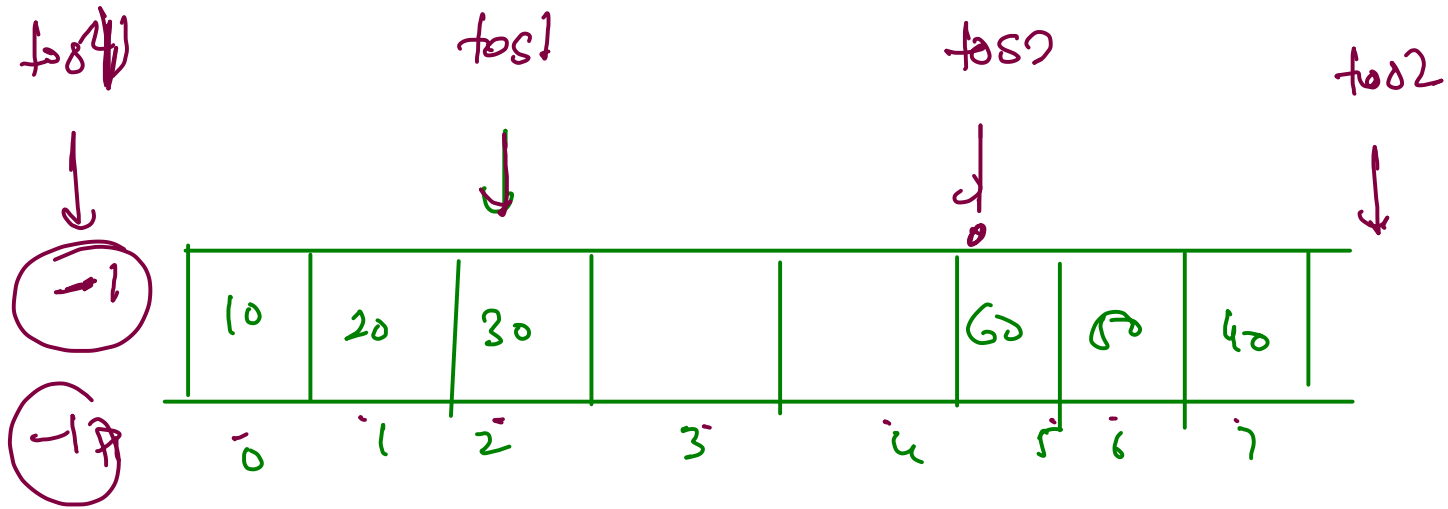
push (40, 1)
push (50, 2)
push (60, 1)
push (70, 2)
push (80, 1) → Stack overflow
push (90, 2)



if (tos1 + 1 < tos2) → for(1)

tos2 - 1 ≥ tos1 → For (2)

tos2 > tos1 + 1



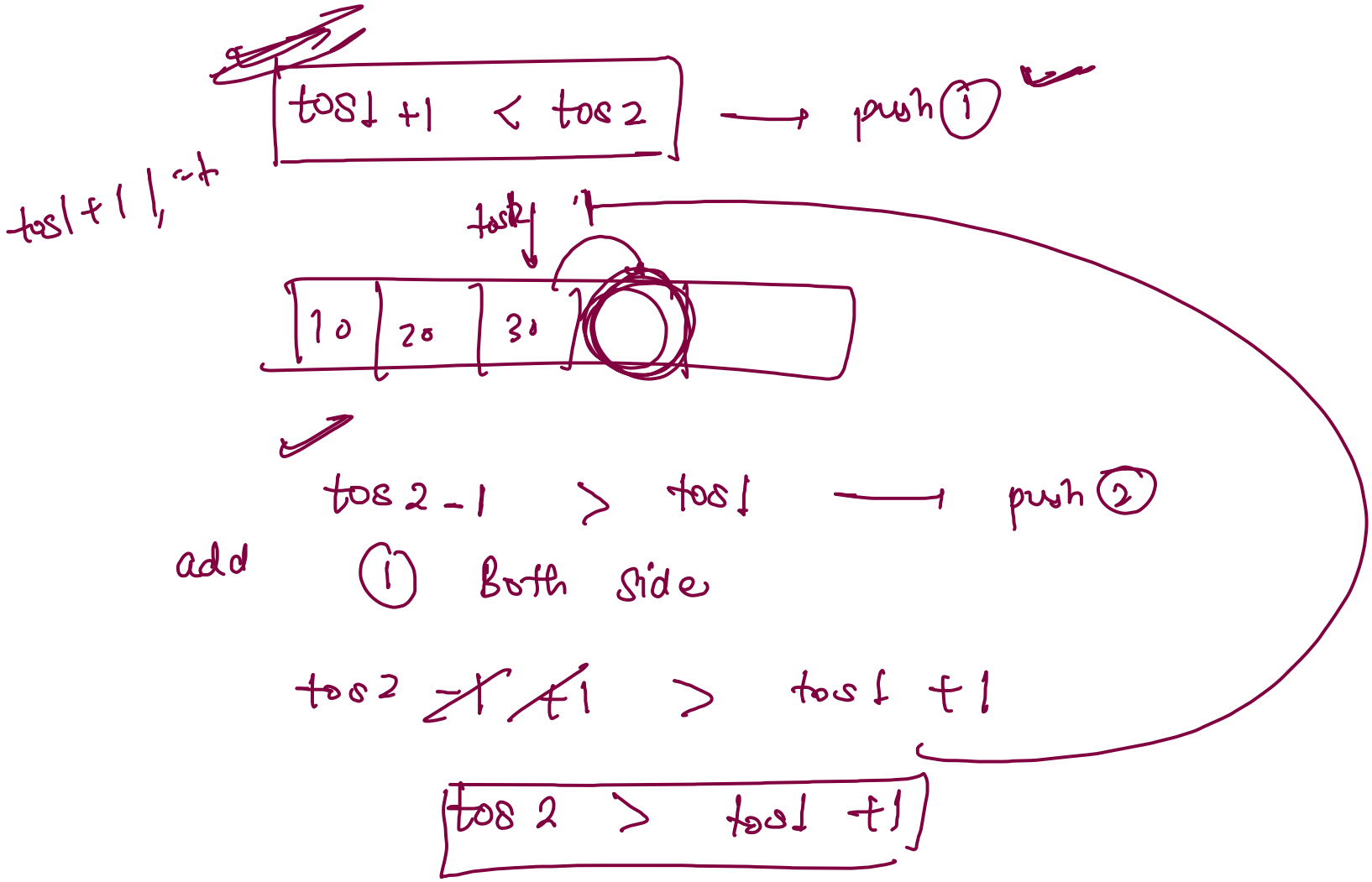
tos1 = (2)

tos1 + 1

Comp. 8 - 8 = (0)

data left - tos2

8 - 5 = (3)


tos1 +1 < tos2
push (1)
tos1 +1 ++
task
10
20
30
tos2 -1 > tos1
push (2)
add (1) Both side
tos2 -1 +1 > tos1 +1
tos 2 > tos1 +1