Doubt Class with Lakshay

Special class

Love Babbar • Aug 3, 2024

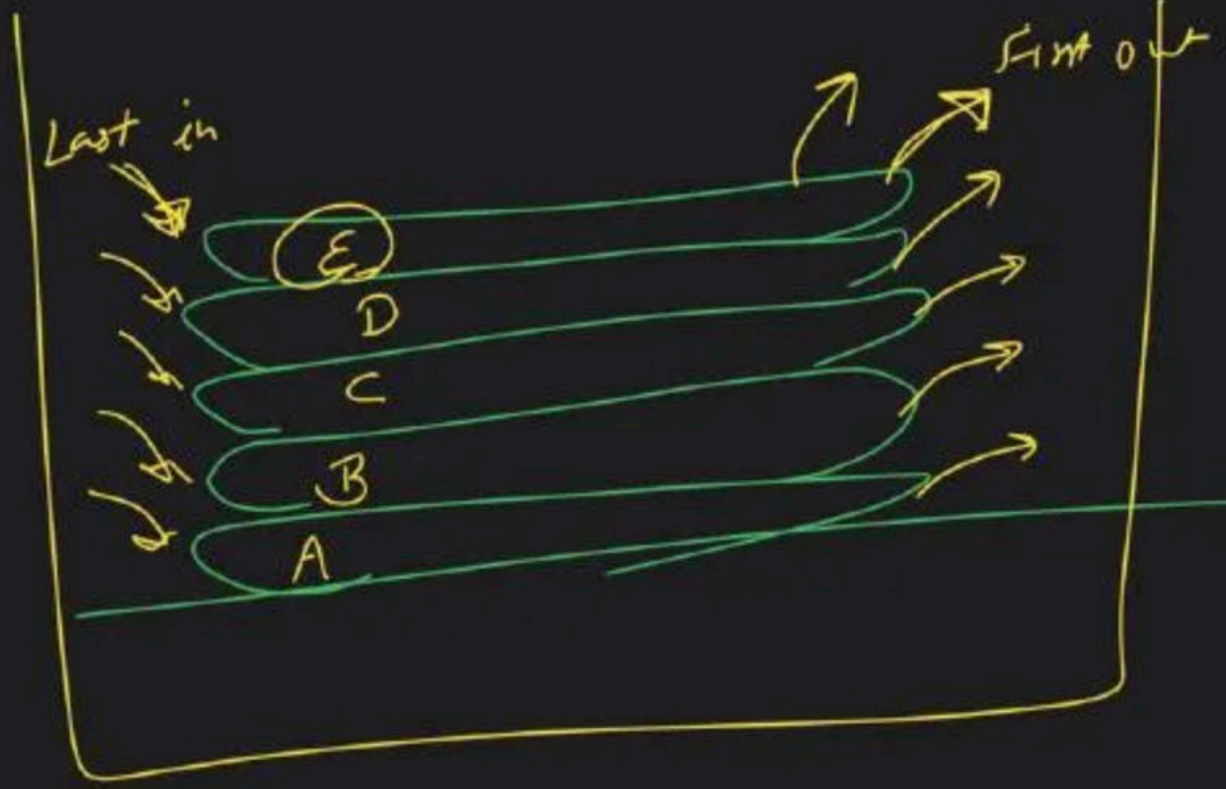
Stack Class - 1

Special class

Love Babbar • July 29, 2024

Stack:

D.S → LIFO ordering



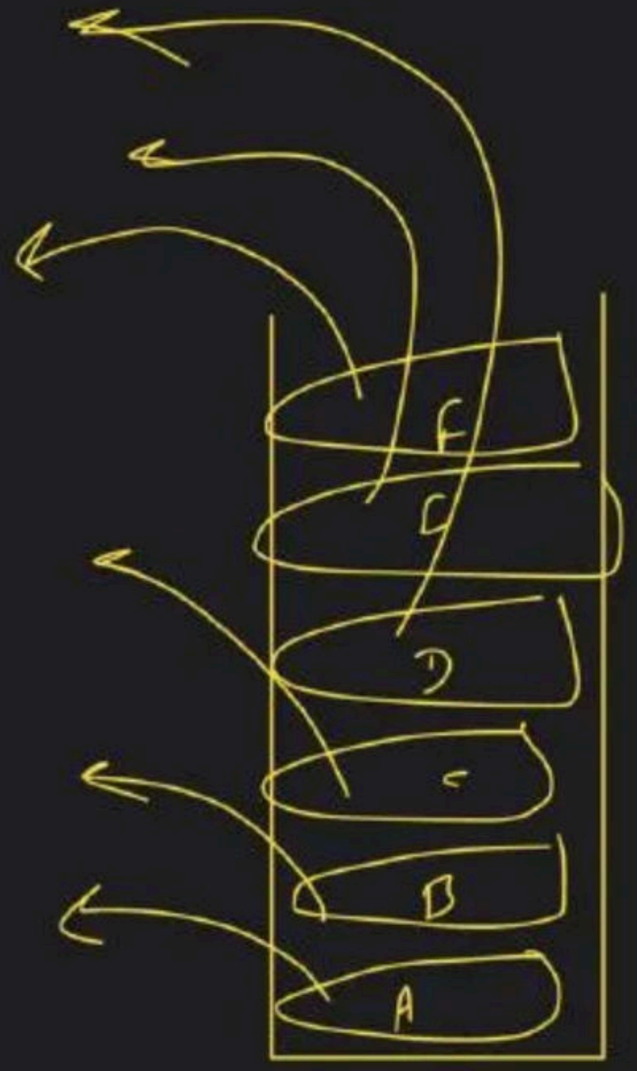
Stack

property

A B C D E

E D C B A

reverse



s+h

STL :-

s.push(10) ✓

s.push(20)

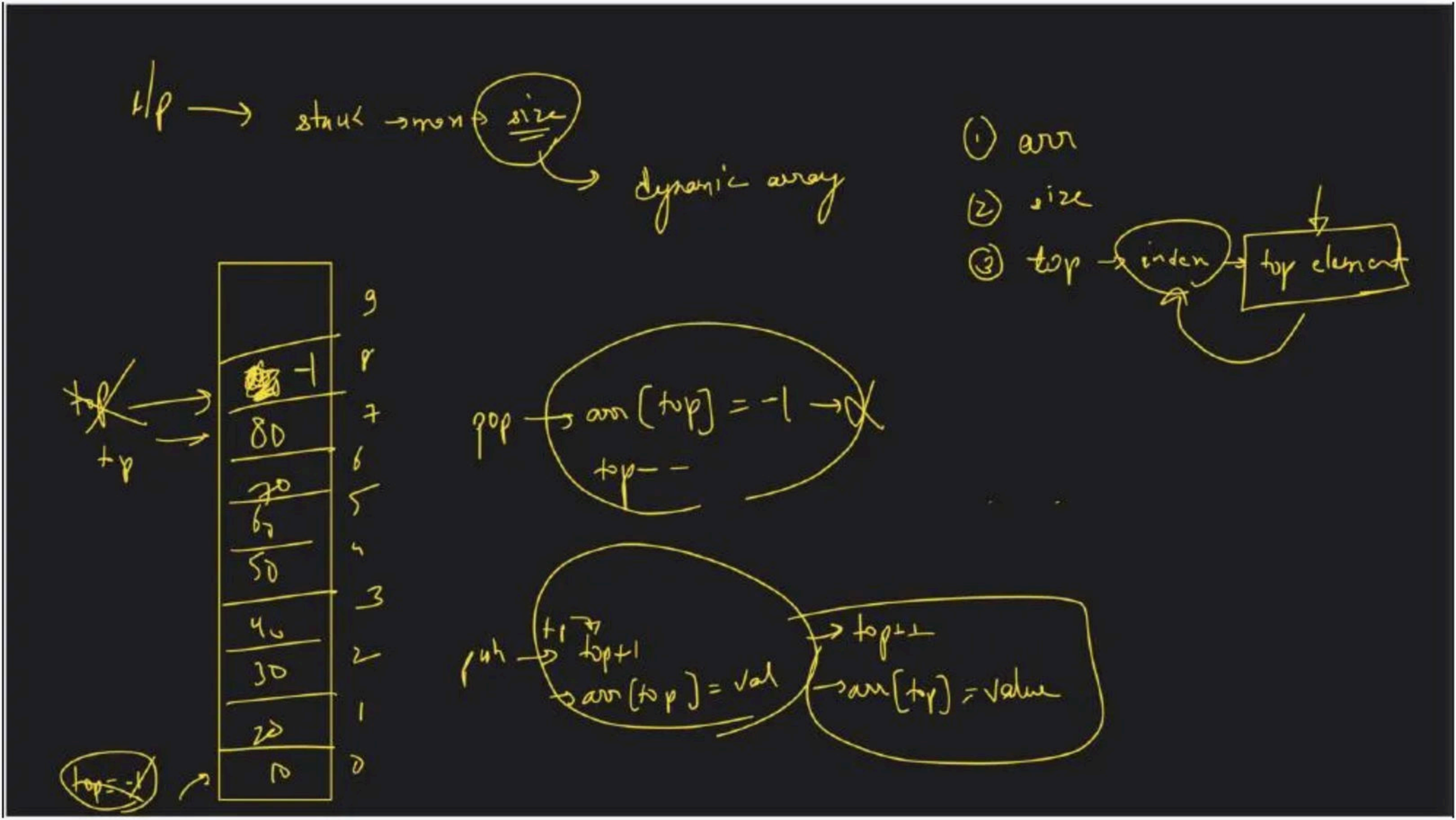
s.push(30)

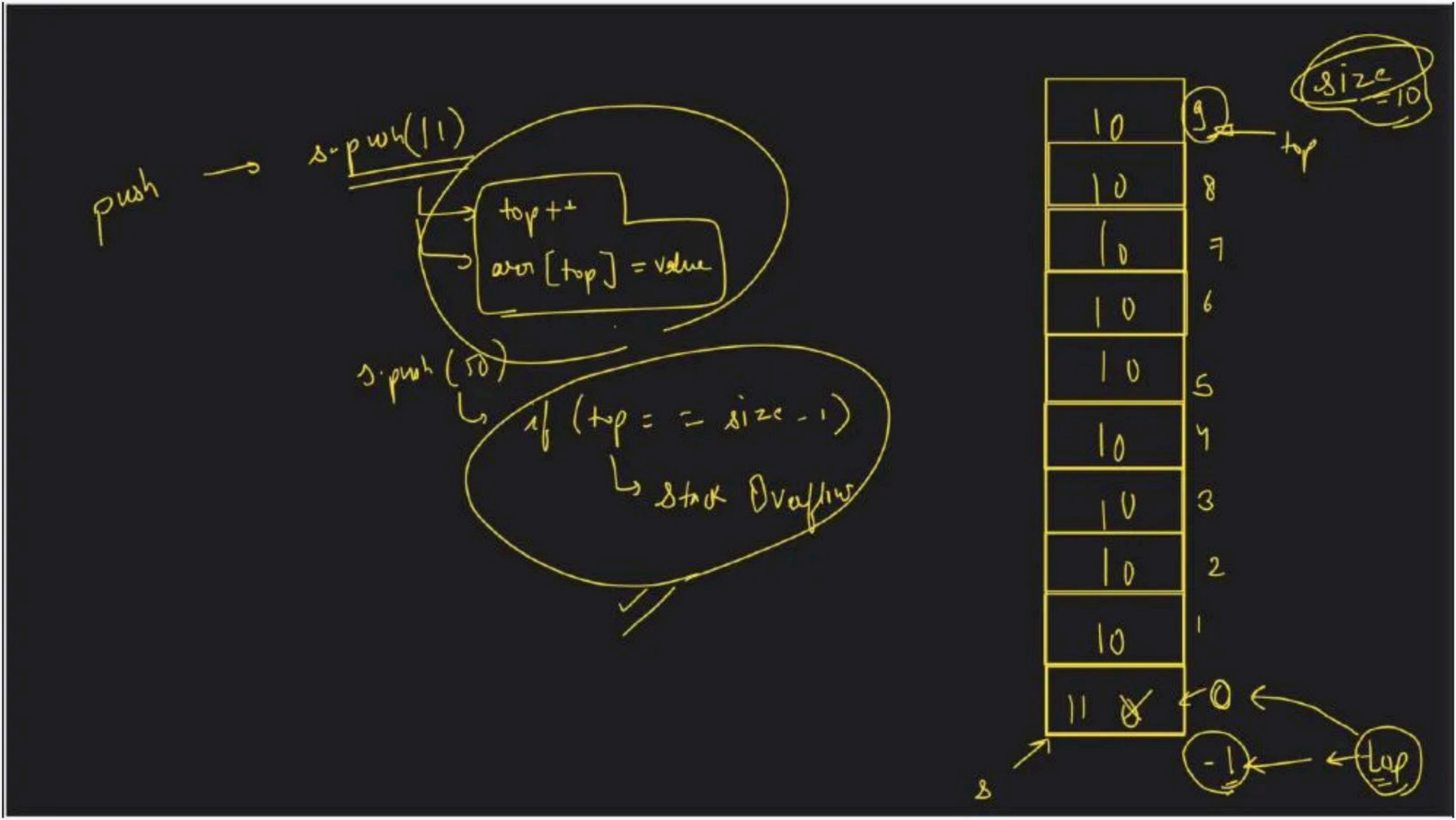
s.size() ✓

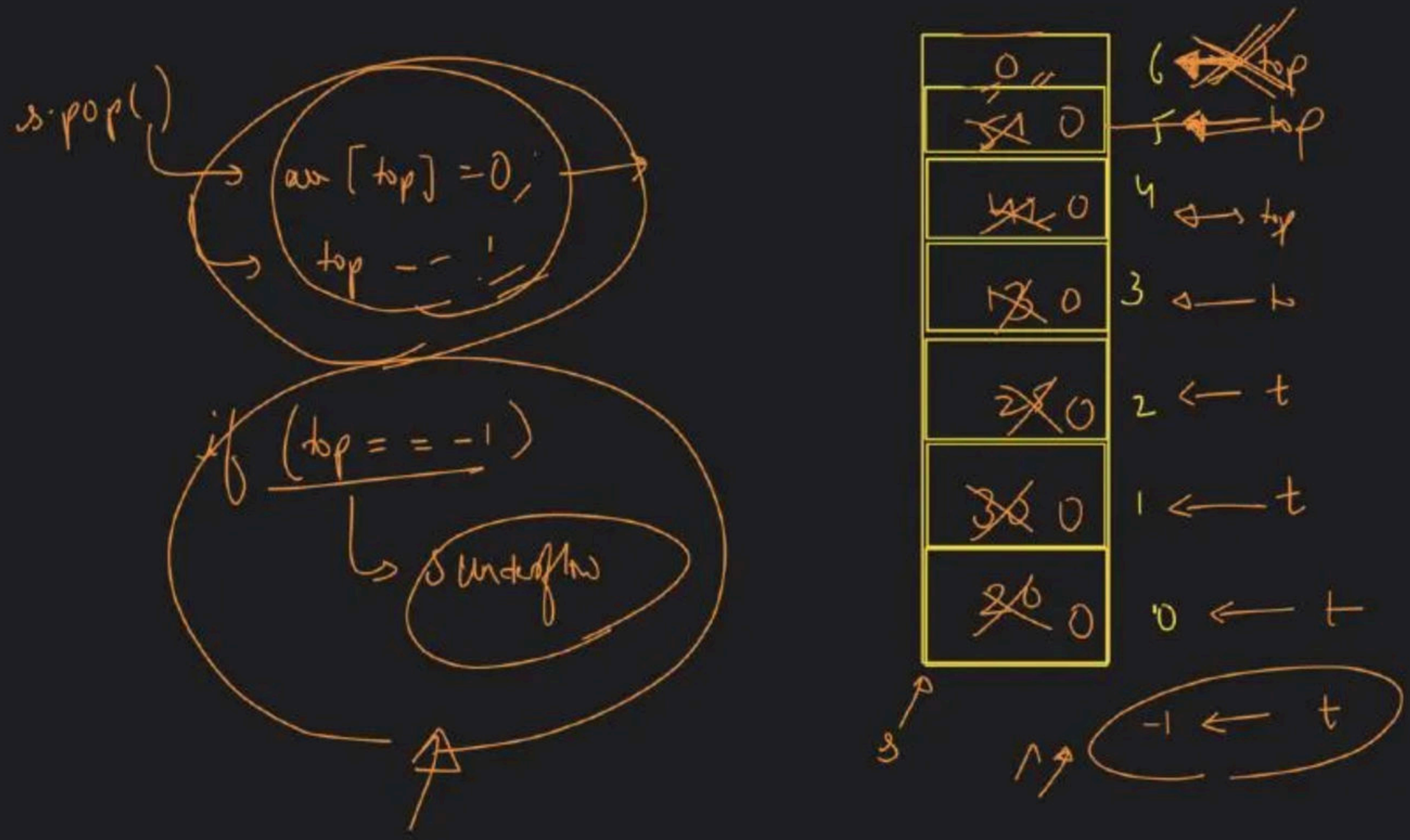
s.empty() -> F ✓

s.pop() ✓

s.top() -> 20 ✓ access only

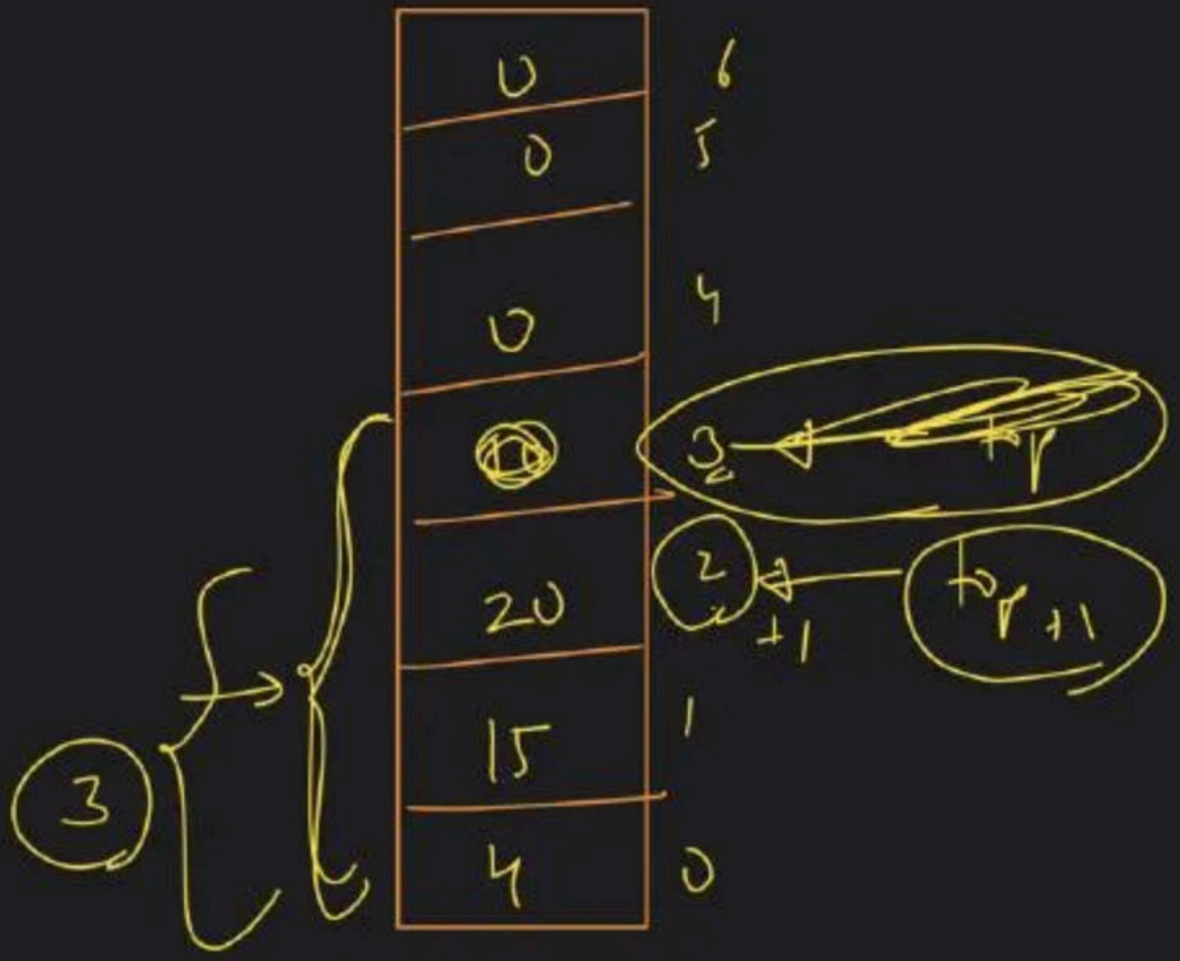

i/p → stack → mem → size
dynamic array
① arr
② size
③ top → index → top element
9
8
7
6
5
4
3
2
1
0
-1
80
70
60
50
40
30
20
top
tp
top = -1
pop → arr[top] = -1 → top--
push → ti → top+1 → arr[top] = val
top+1
arr[top] = value


push → s.push(11)
top++
arr[top] = value
s.push(10)
if (top == size - 1)
Stack Overflow
size = 10
9
top
10
8
7
6
5
4
3
2
1
0
11
-1
top
s


s.pop()
arr[top] = 0;
top --;
if (top == -1)
Underflow
top
top
top
top
top
top
s
t
t
t
t
t
t

getSize()

return top + 1



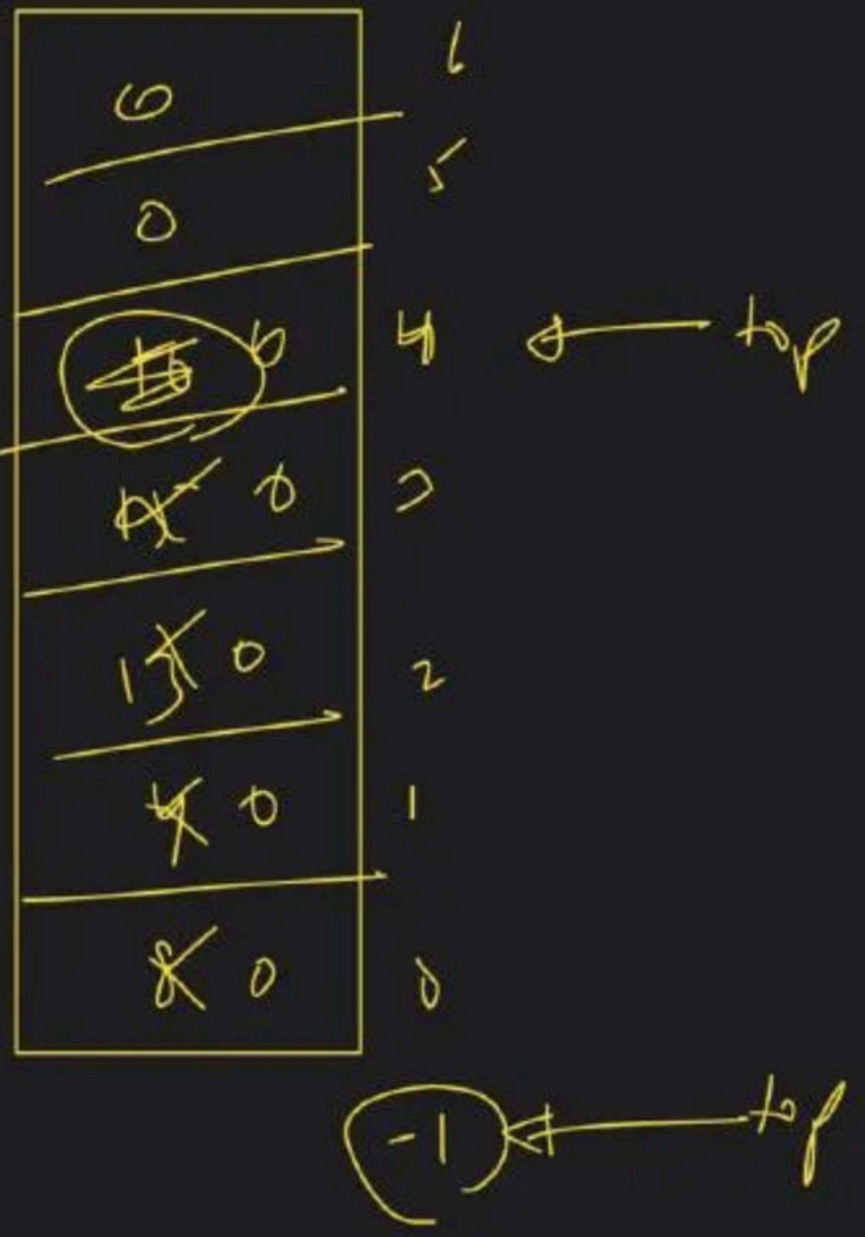
getTop()

↳ return arr[top];

↳ if (top == -1)

↳ if true

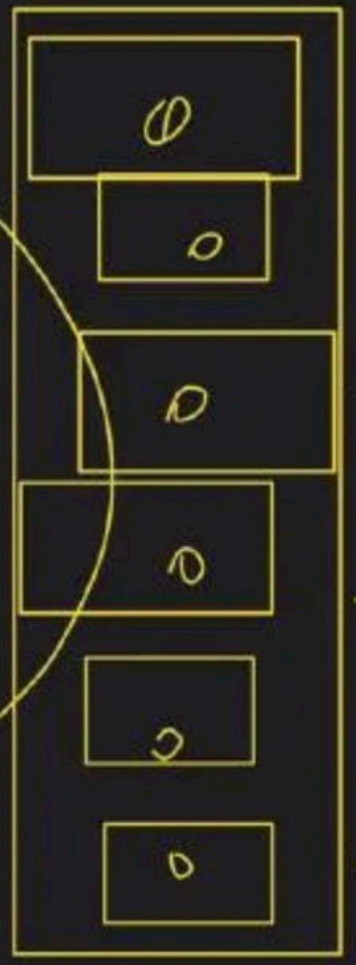
↳ return -1

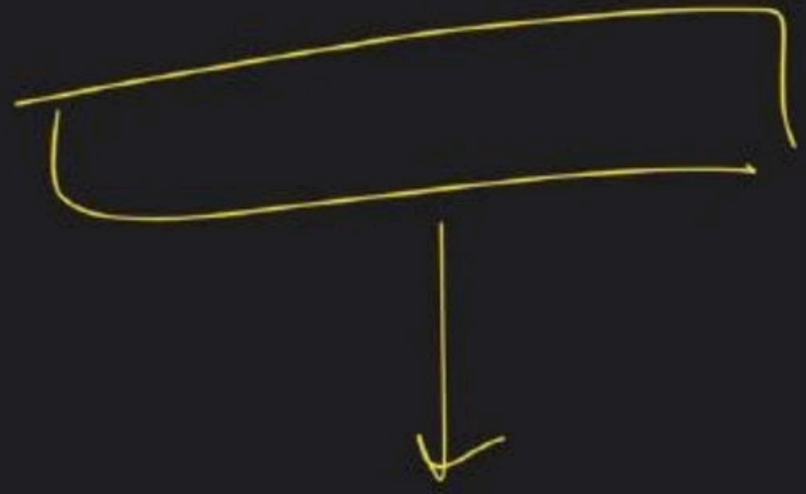


isEmpty

```
if (top == -1)
    return true

else
    return false
```




I stall


2 stacks

→ Implement 2 stacks in an Array

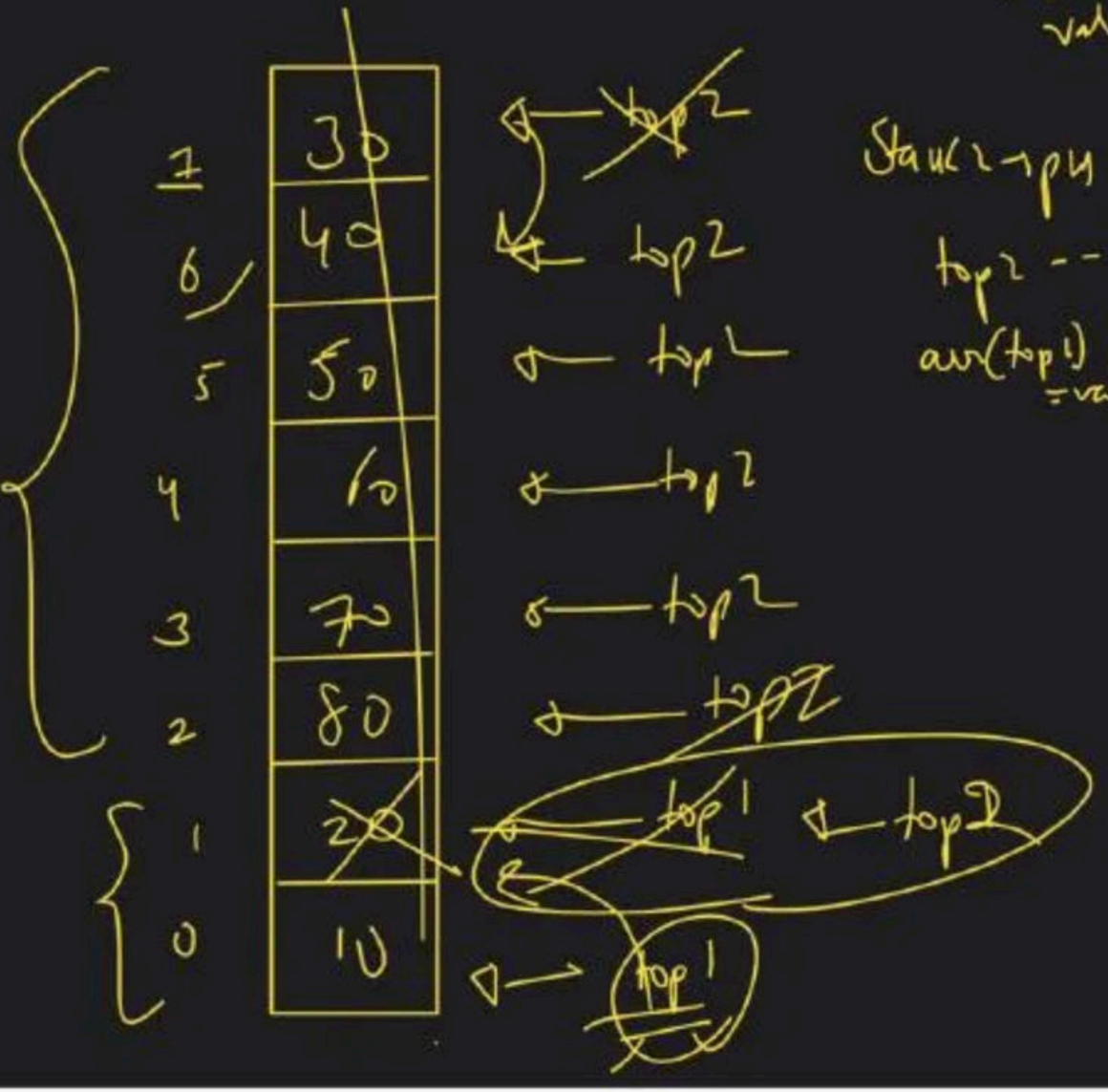
Stack1→push
↳ top1++
↳ arr(top1) = val

#1 → array /2 parts

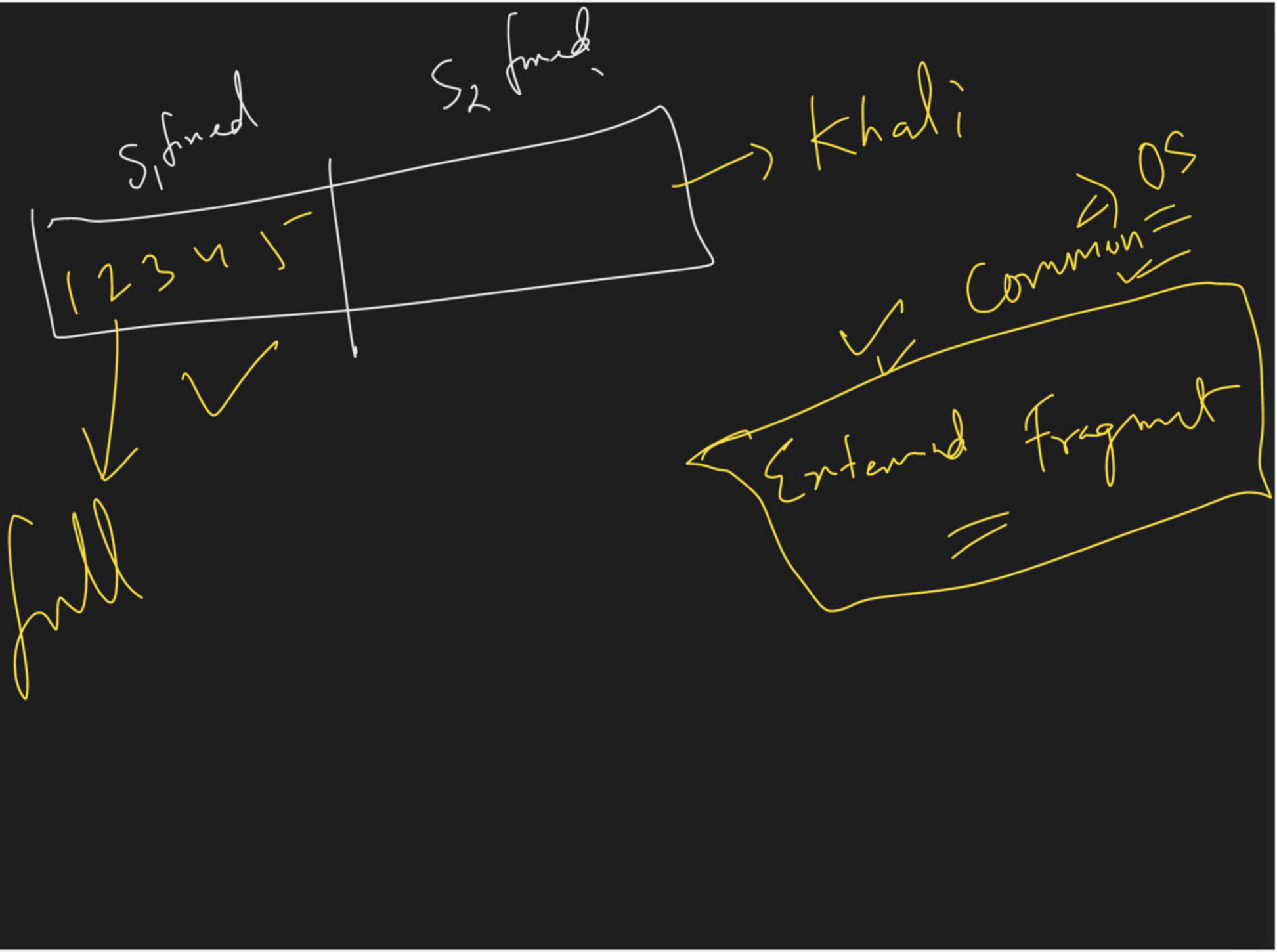
#2 top1, top2

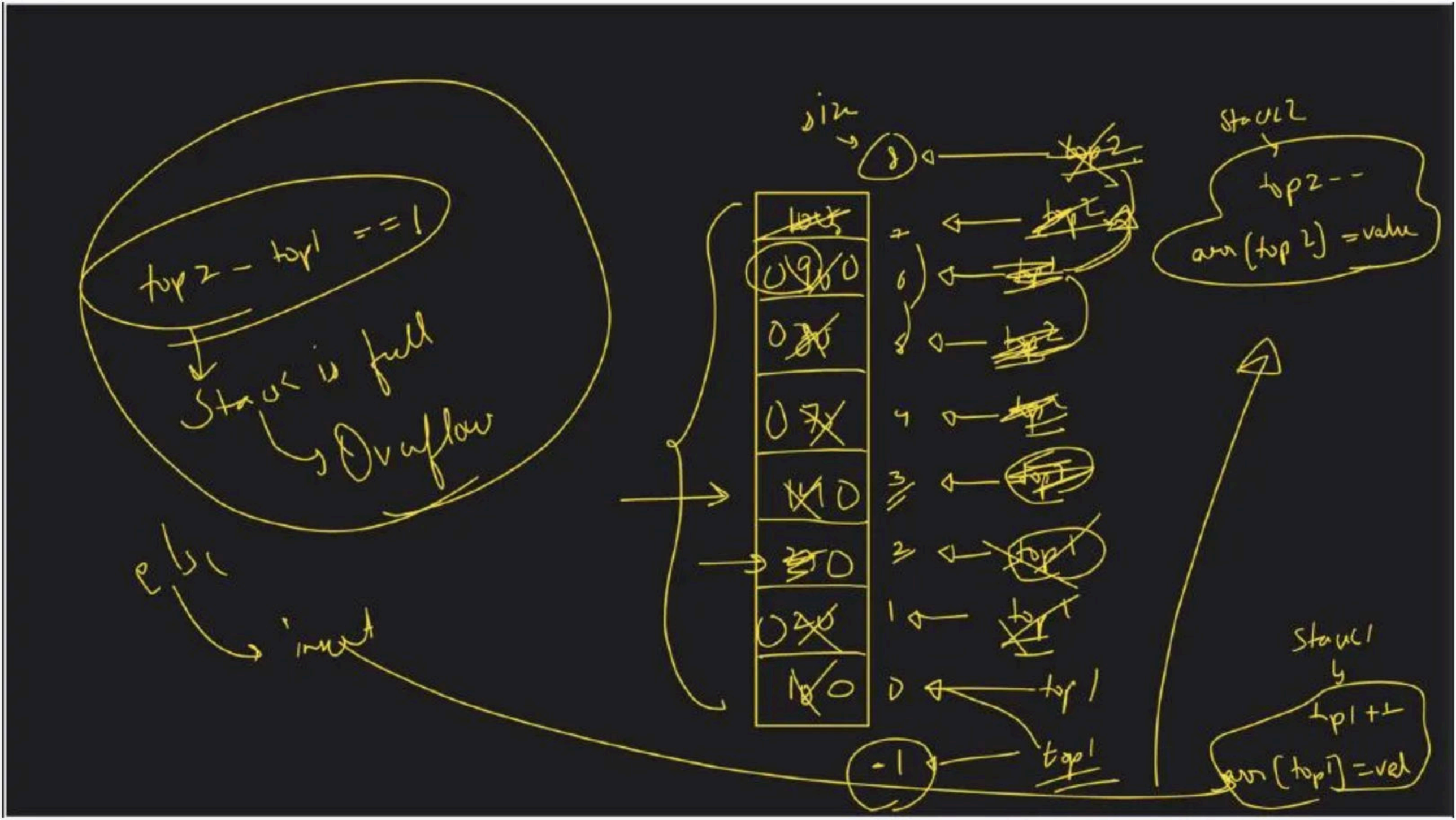
top1 == top2 - 1

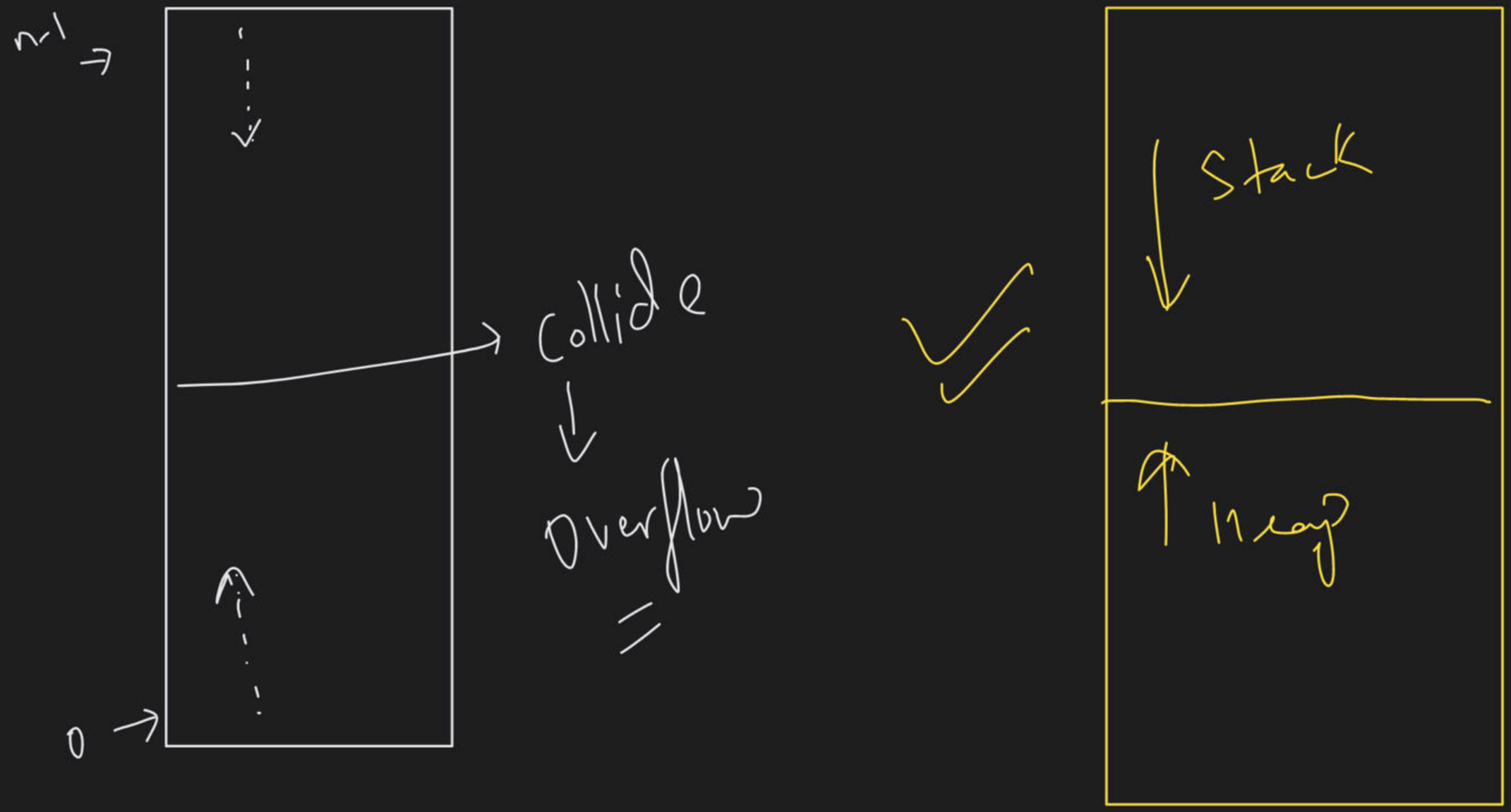
↳ S. is Full
↳ Overflow

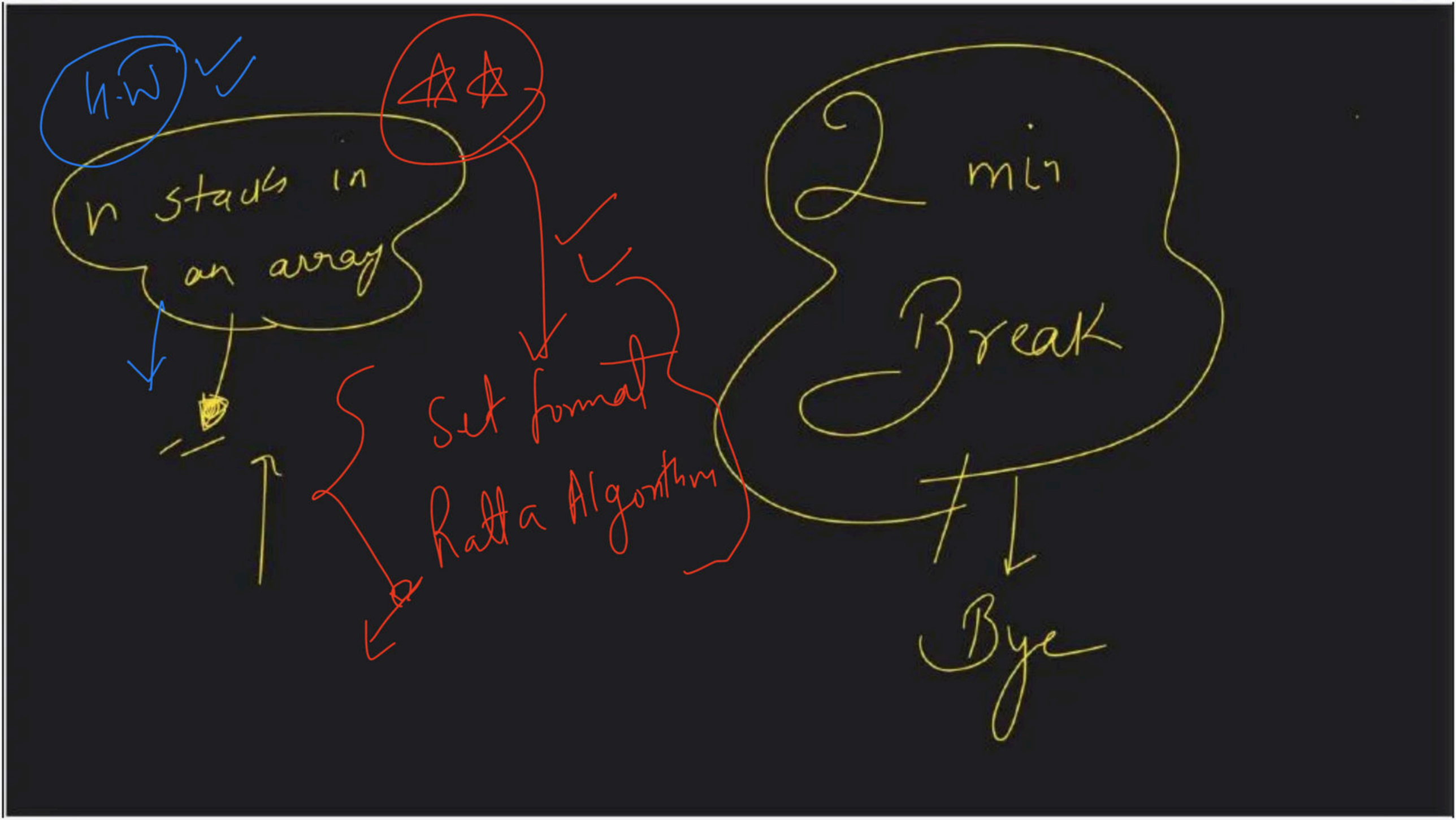


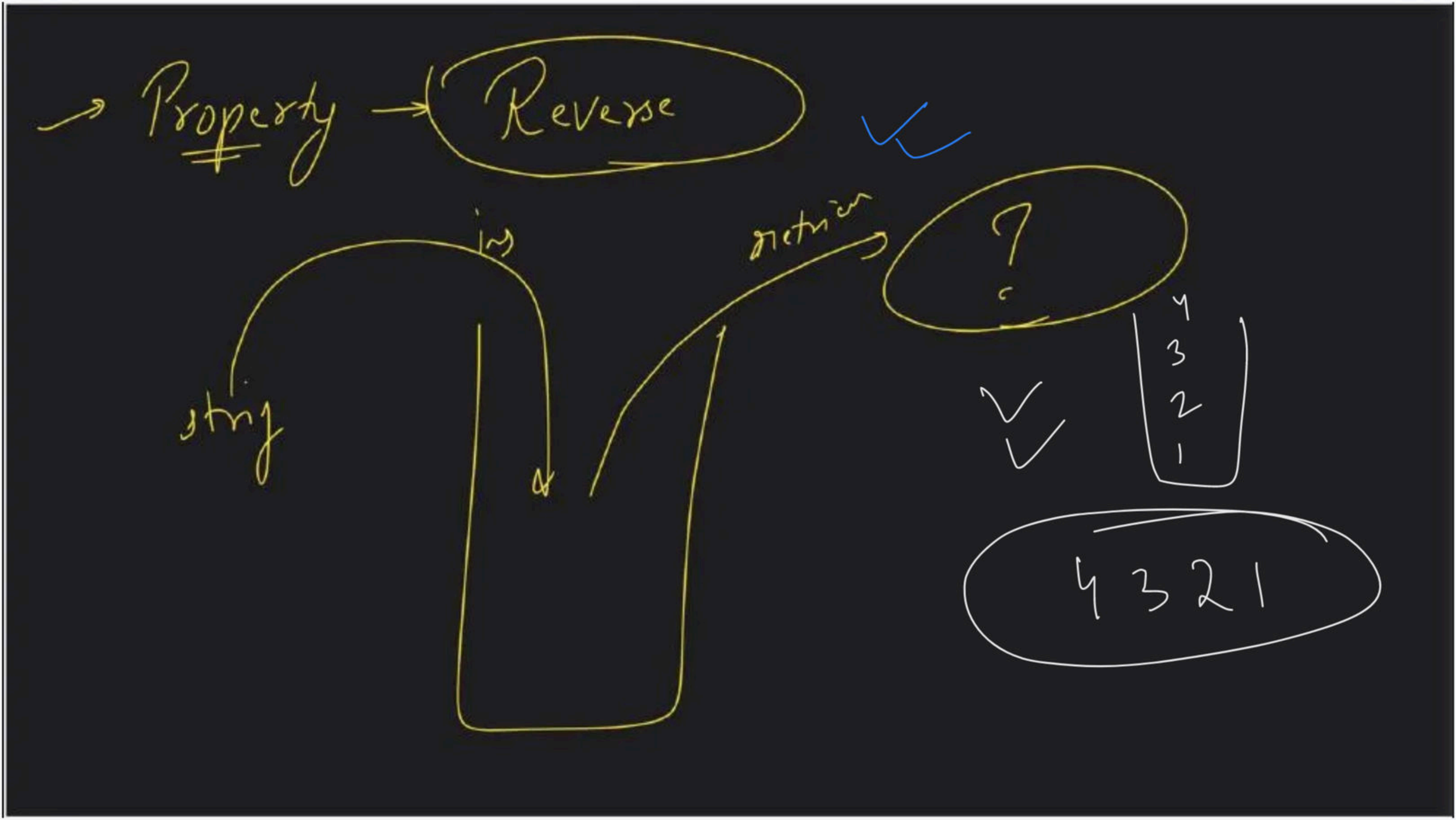
Stack2→push
top2--
arr(top1) = val


S_1 fixed
S_2 fixed
1 2 3 4 5
Khali
Common OS
Extended fragment
full


top2 - top1 == 1
Stack is full
Overflow
else
insert
size
8
top2
top1
top1
-1
Stack2
top2--
arr[top2] = value
Stack1
top1++
arr[top1] = val


n-1
collide
Overflow
0
stack
Heap


H.W
n stacks in an array
Set format
Ratta Algorithm
2 min
Break
Bye

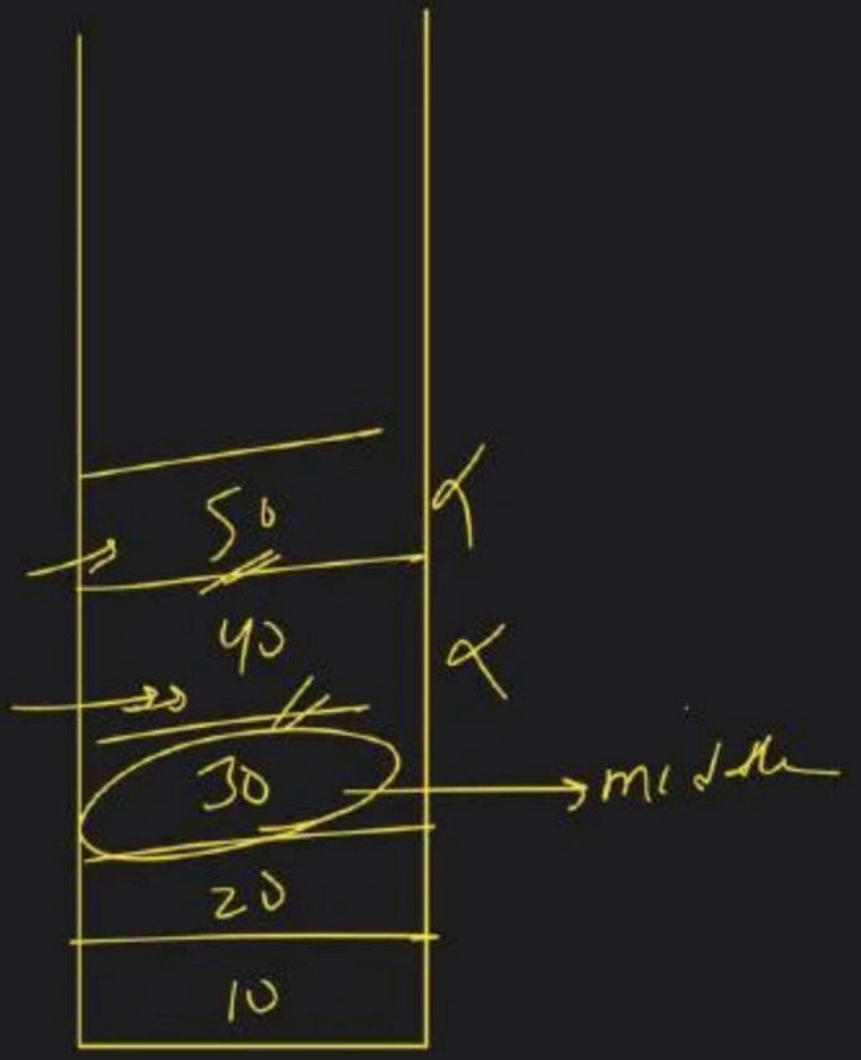

Property
Reverse
int
string
?
4
3
2
1
4321

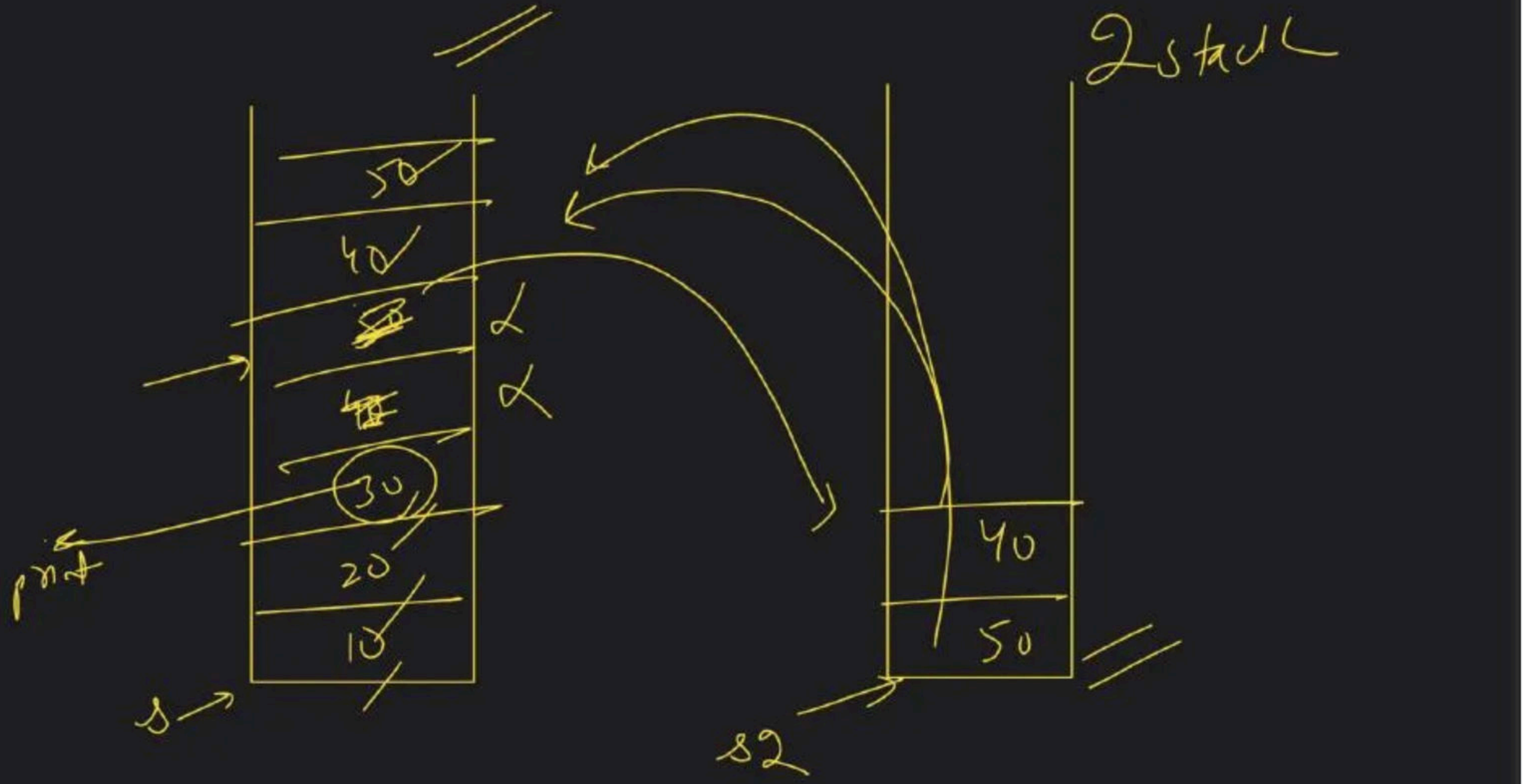
→ Pattern

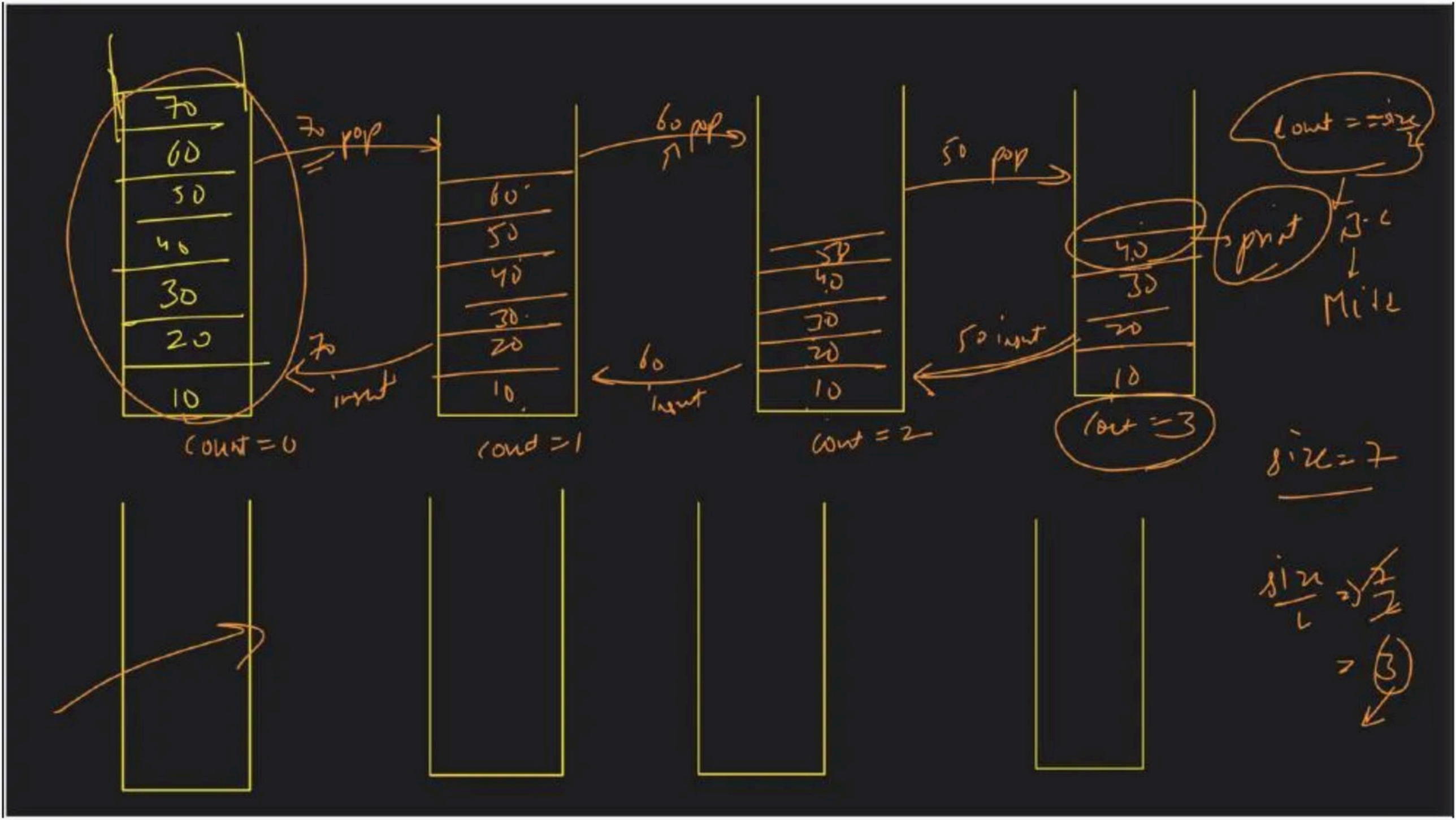
→ PrintMiddle

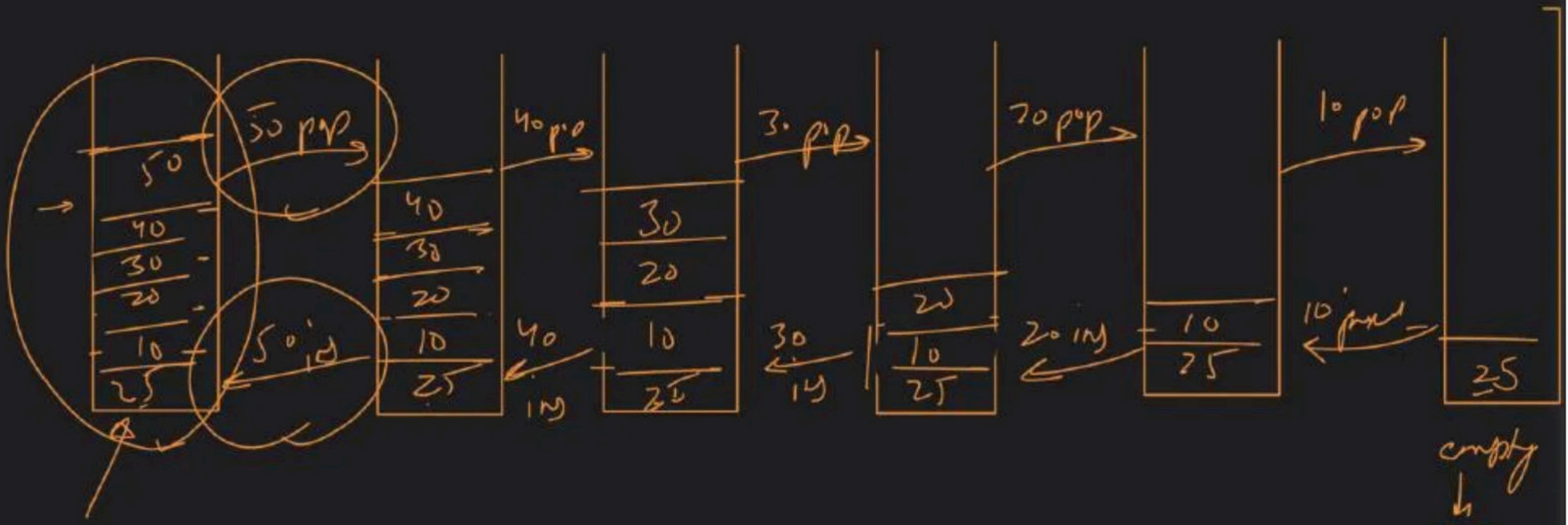
size → 5

size/2 → 5/2 (2)




2 stack
50
40
30
20
10
print
S
S2
40
50


70
60
50
40
30
20
10
70 pop
70 insert
60 pop
60 insert
50 pop
50 insert
count = 0
count = 1
count = 2
count = 3
count == size
print
Mid
size = 7
size/2 = 7/2 = 3



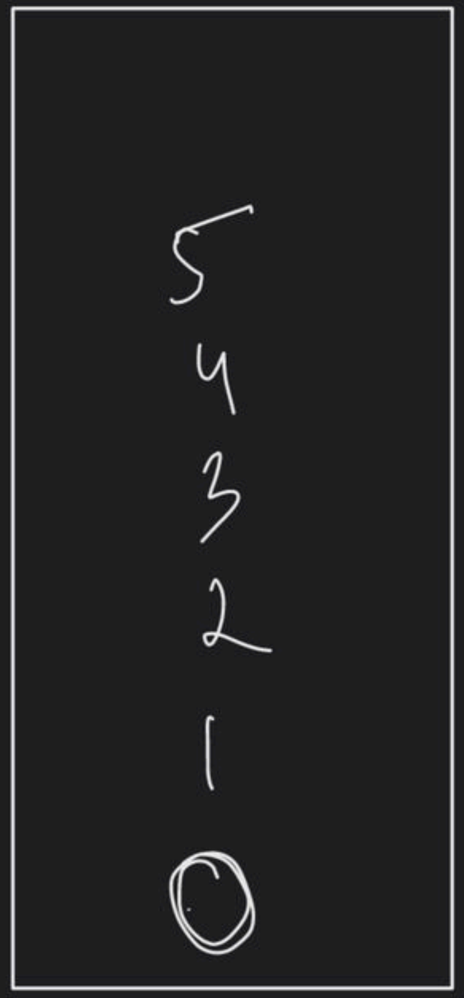
insert At Bottom (value)

value → 25

empty

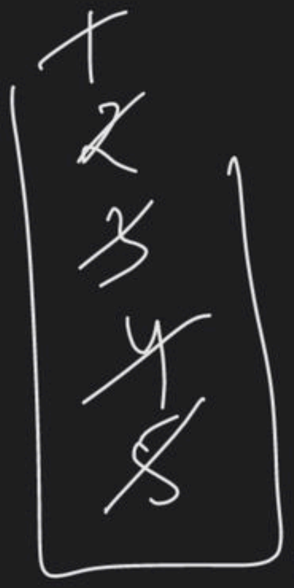
B.C

Insert value

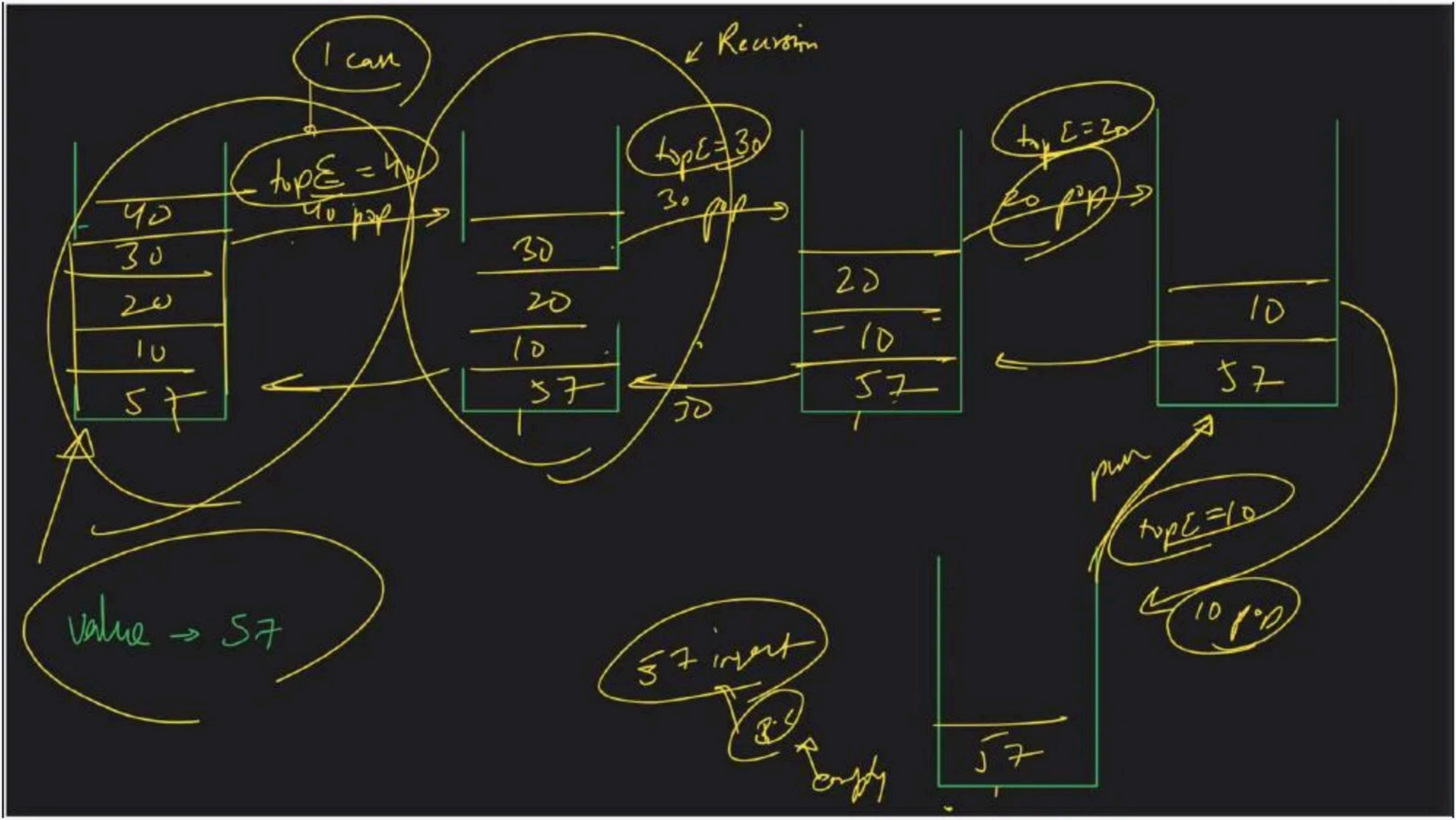


S_1

0

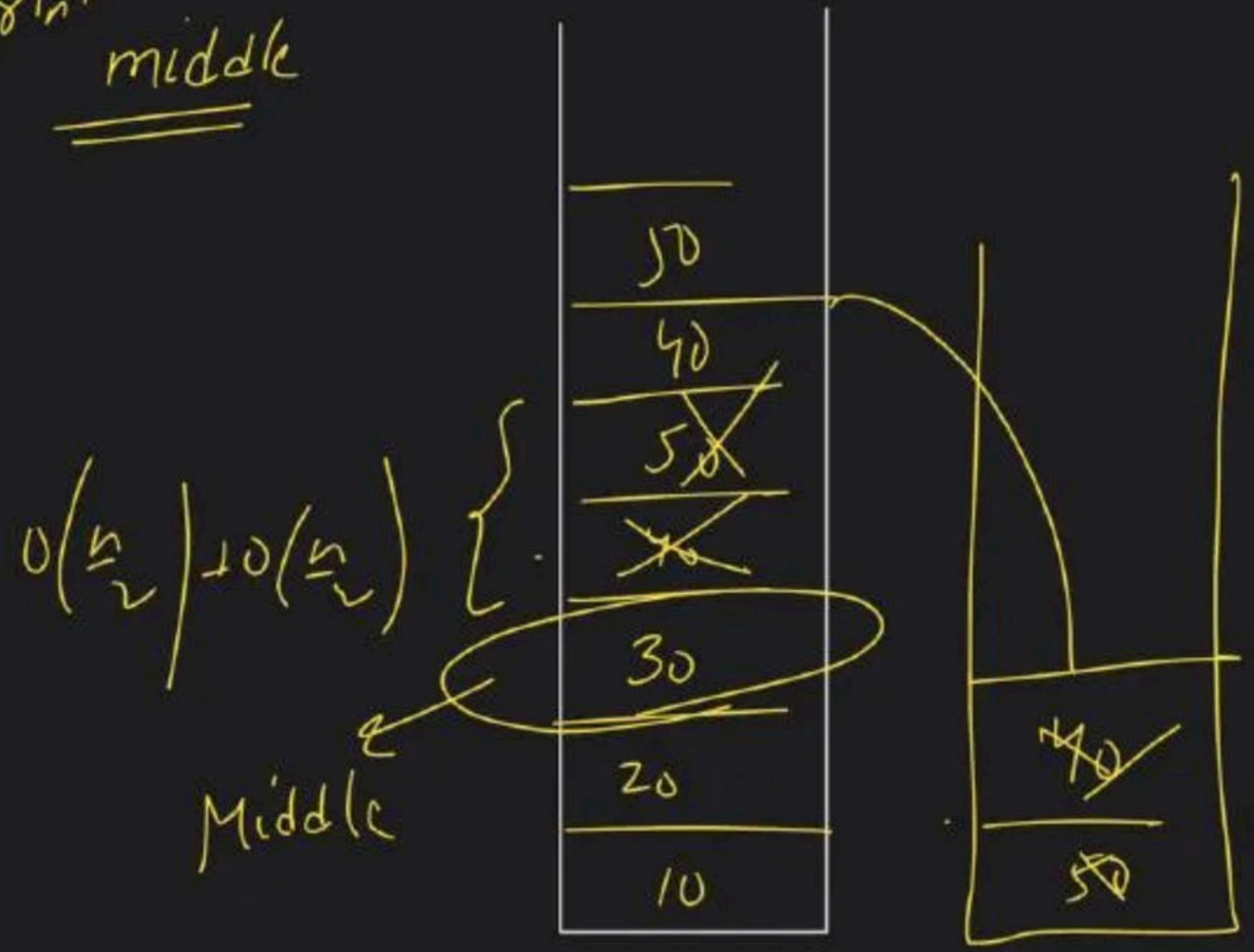


S_2 | OS RE-stack


I call
Recursion
topE = 40
40 pop
40
30
20
10
57
topE = 30
30 pop
30
20
10
57
topE = 20
20 pop
20
10
57
10
57
pop
topE = 10
10 pop
57 insert
Value → 57
57


Stack Class - 2
Special class
Love Babbar • July 31, 2024

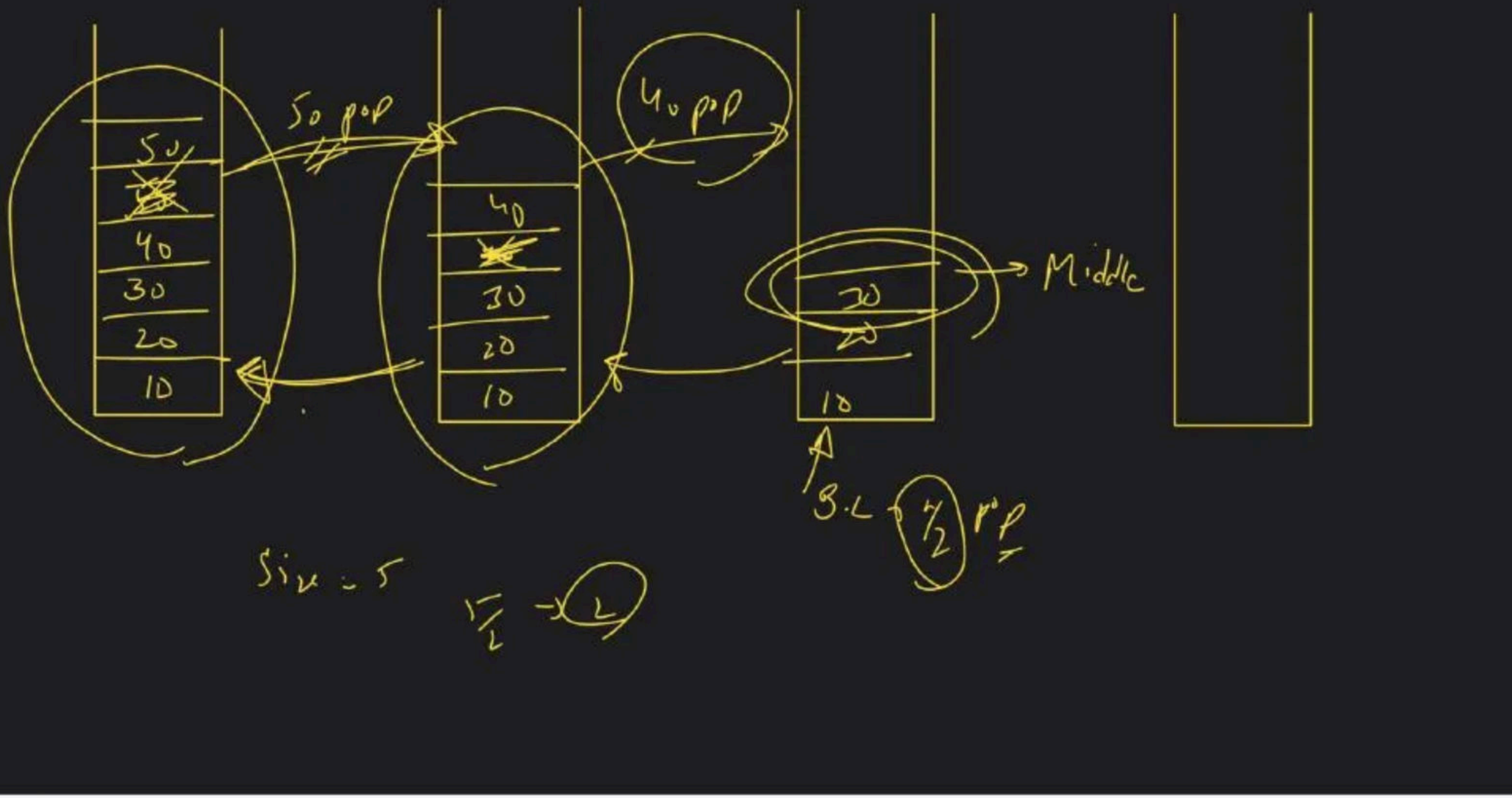
Print middle



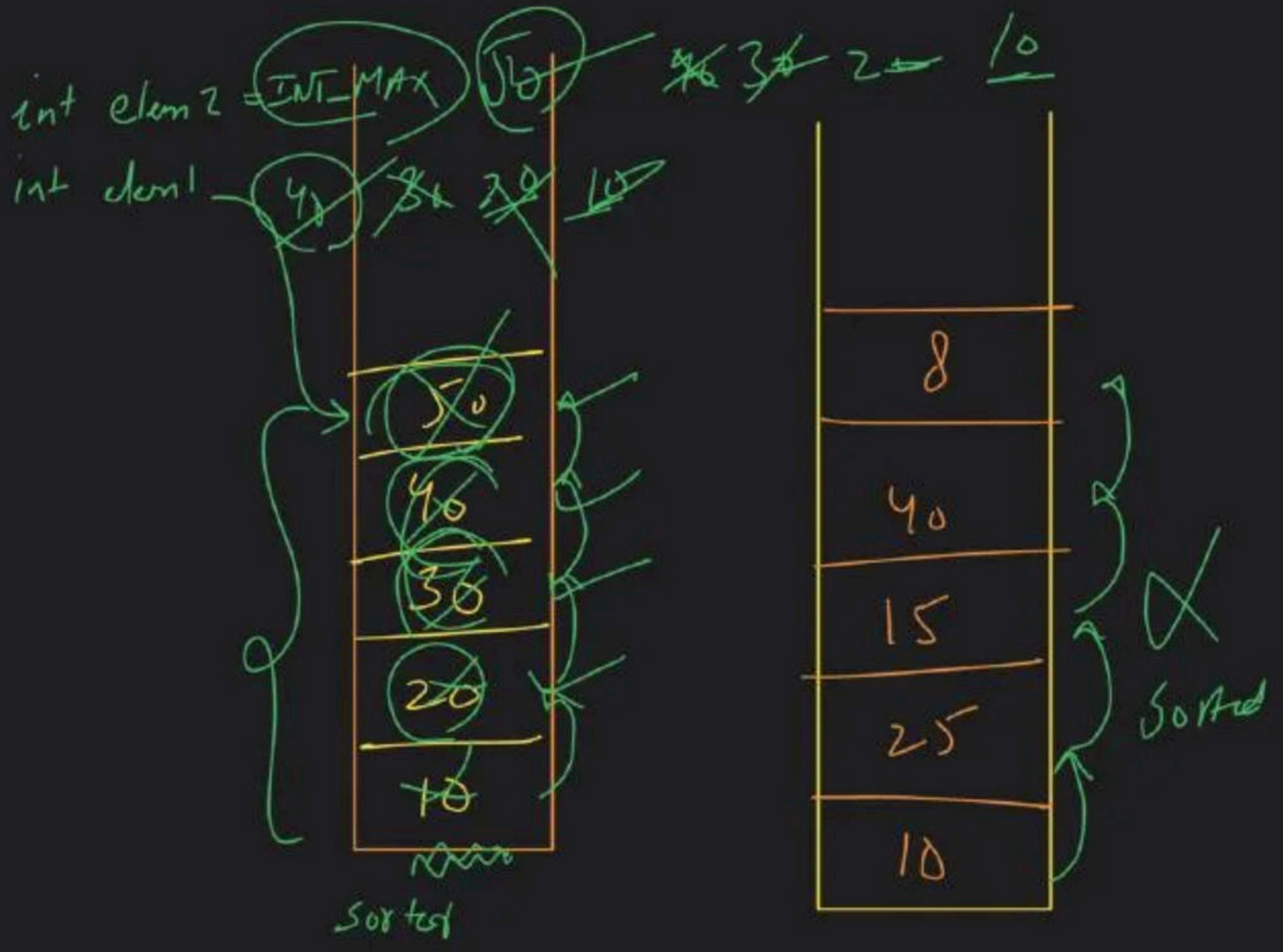
size = n

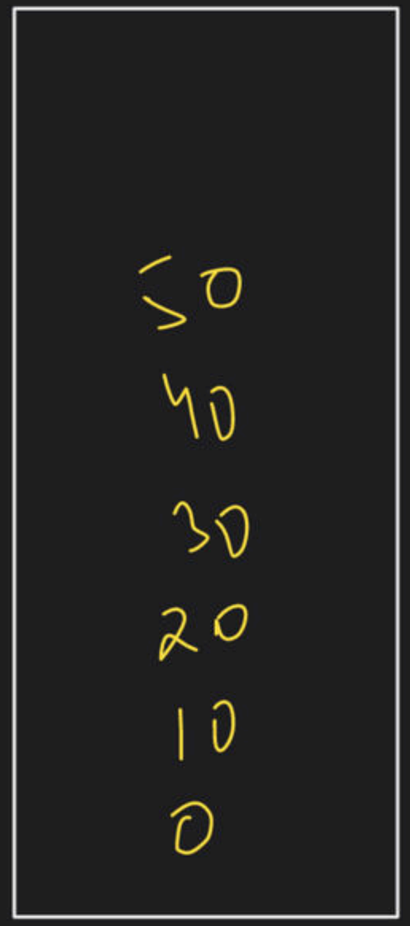
$n/2 \rightarrow \frac{n}{2}$ (2)

S.C $\rightarrow$ O(n)


50
40
30
20
10
50 pop
40
30
20
10
40 pop
30
20
10
Middle
S.L
½ pop
Size = 5
5/2 → 2

check Sorted

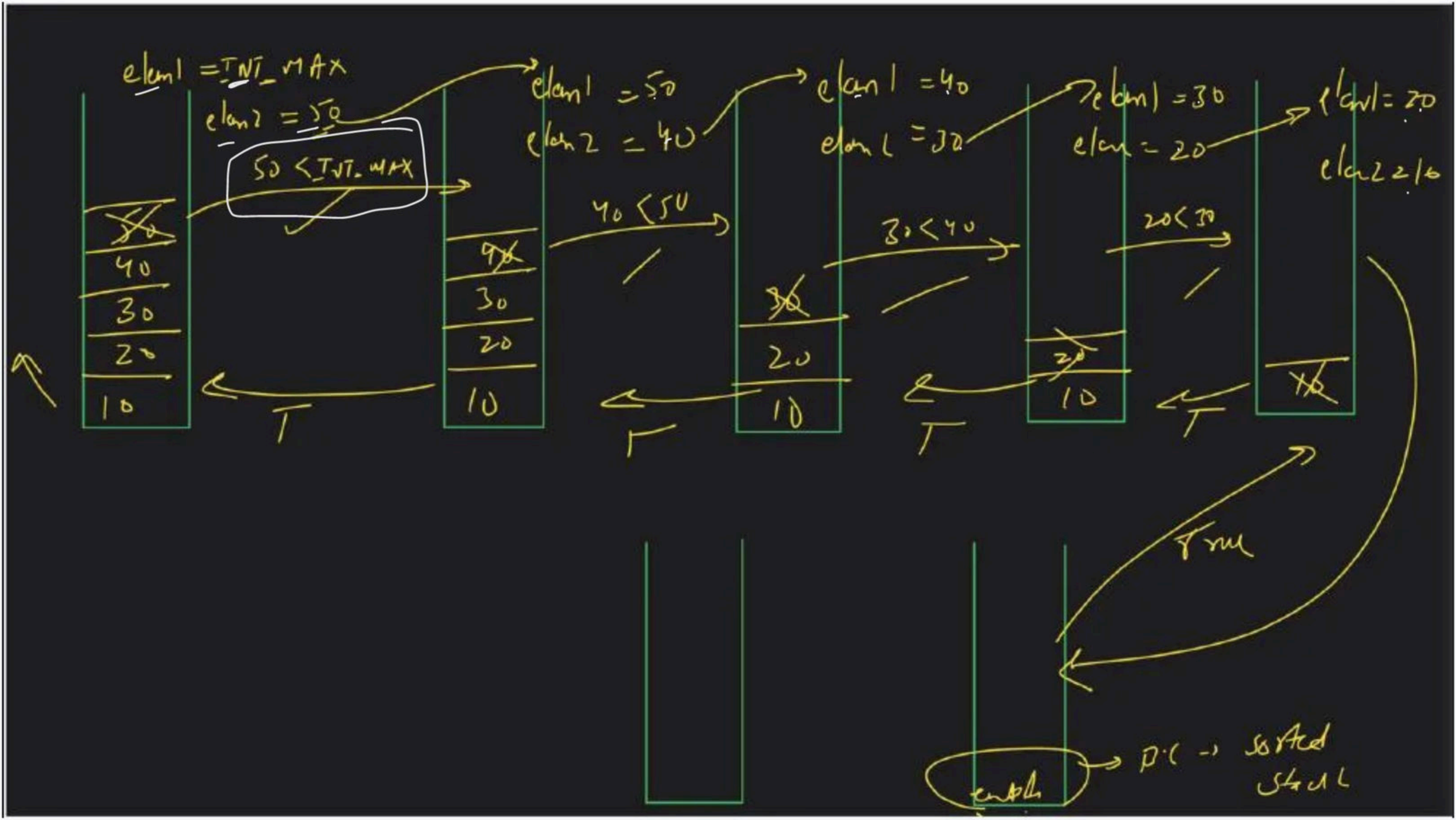


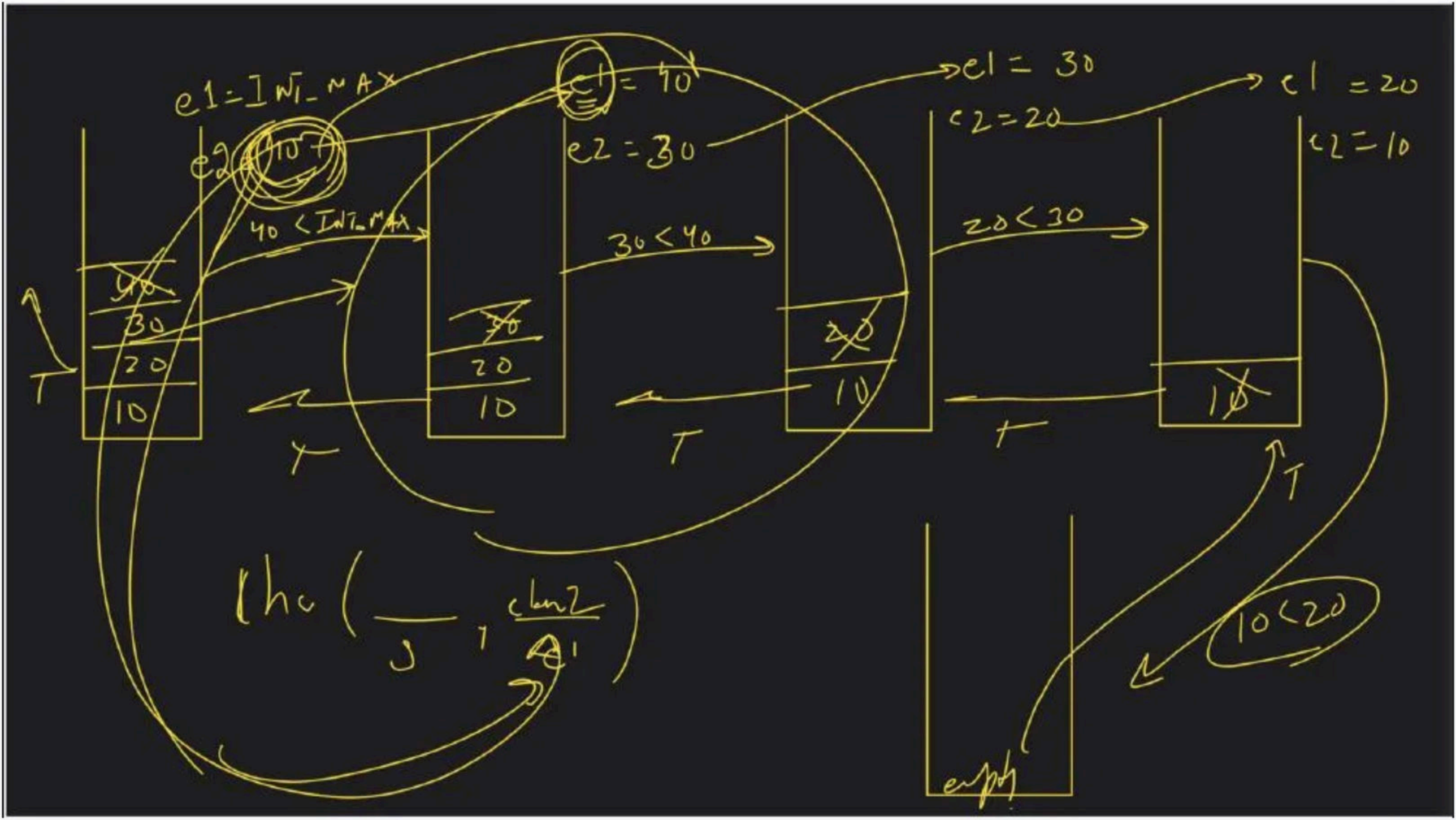


(elemt > s.top())

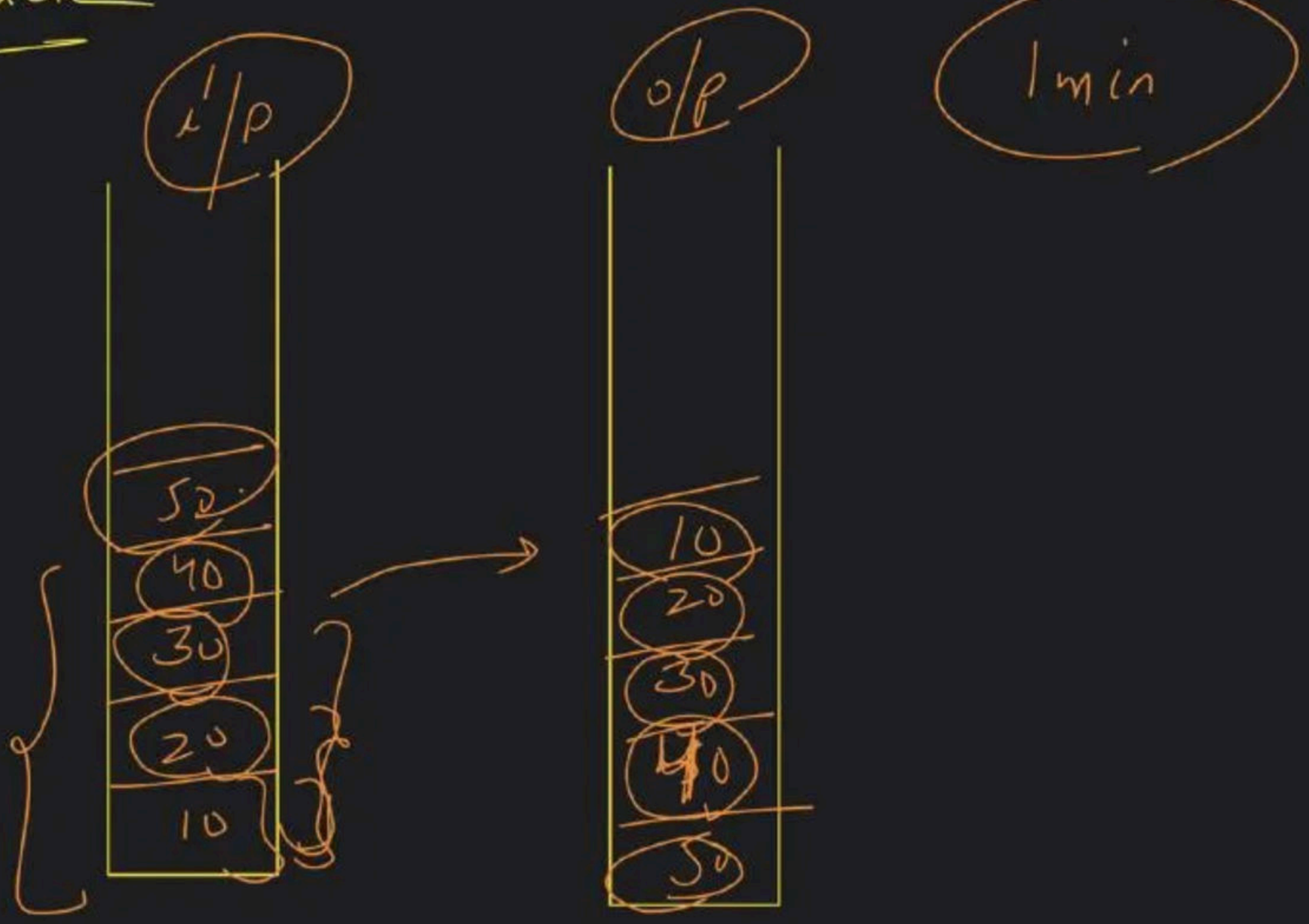
↳ RE

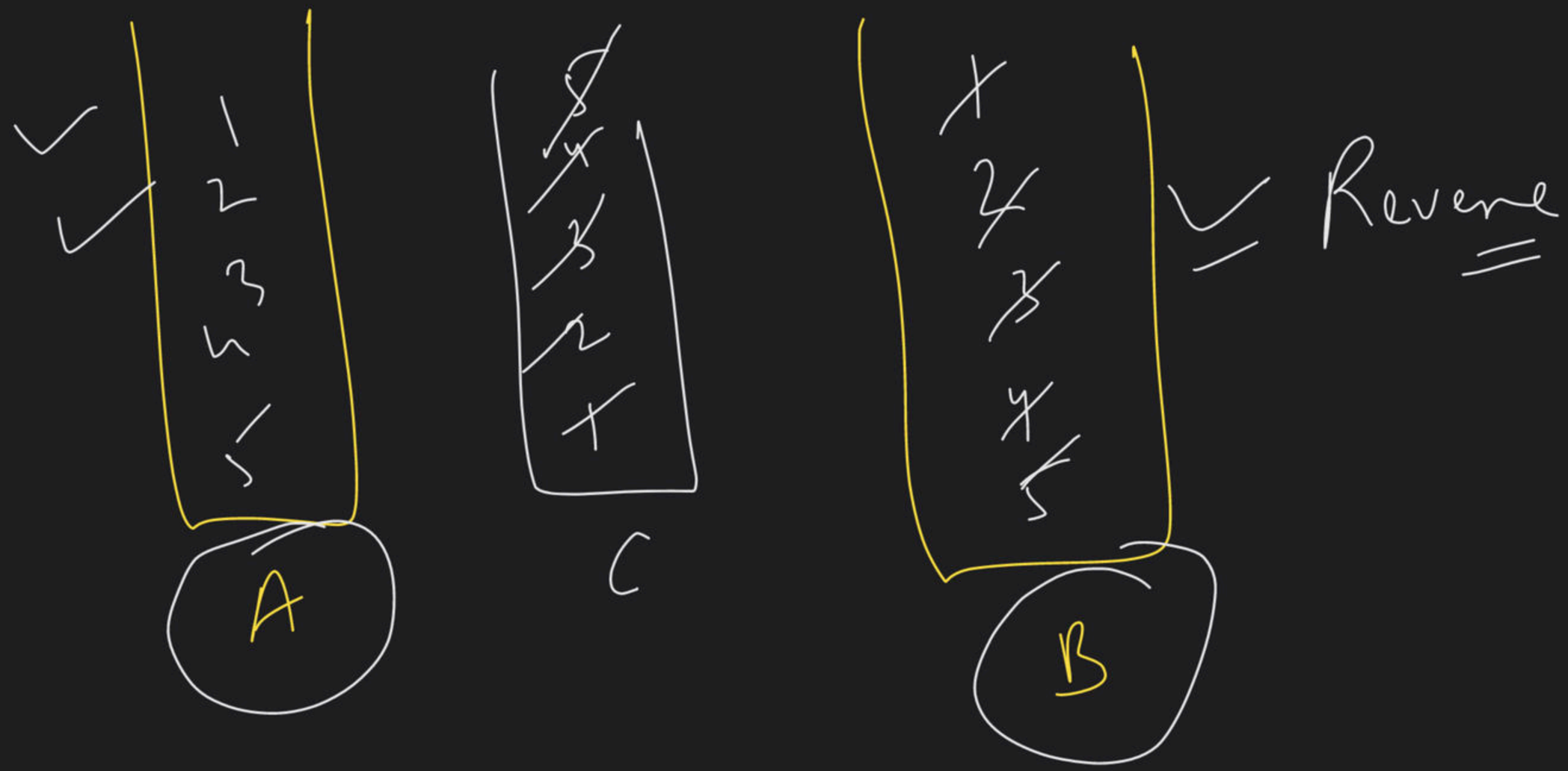
- elemt = INT_MAX

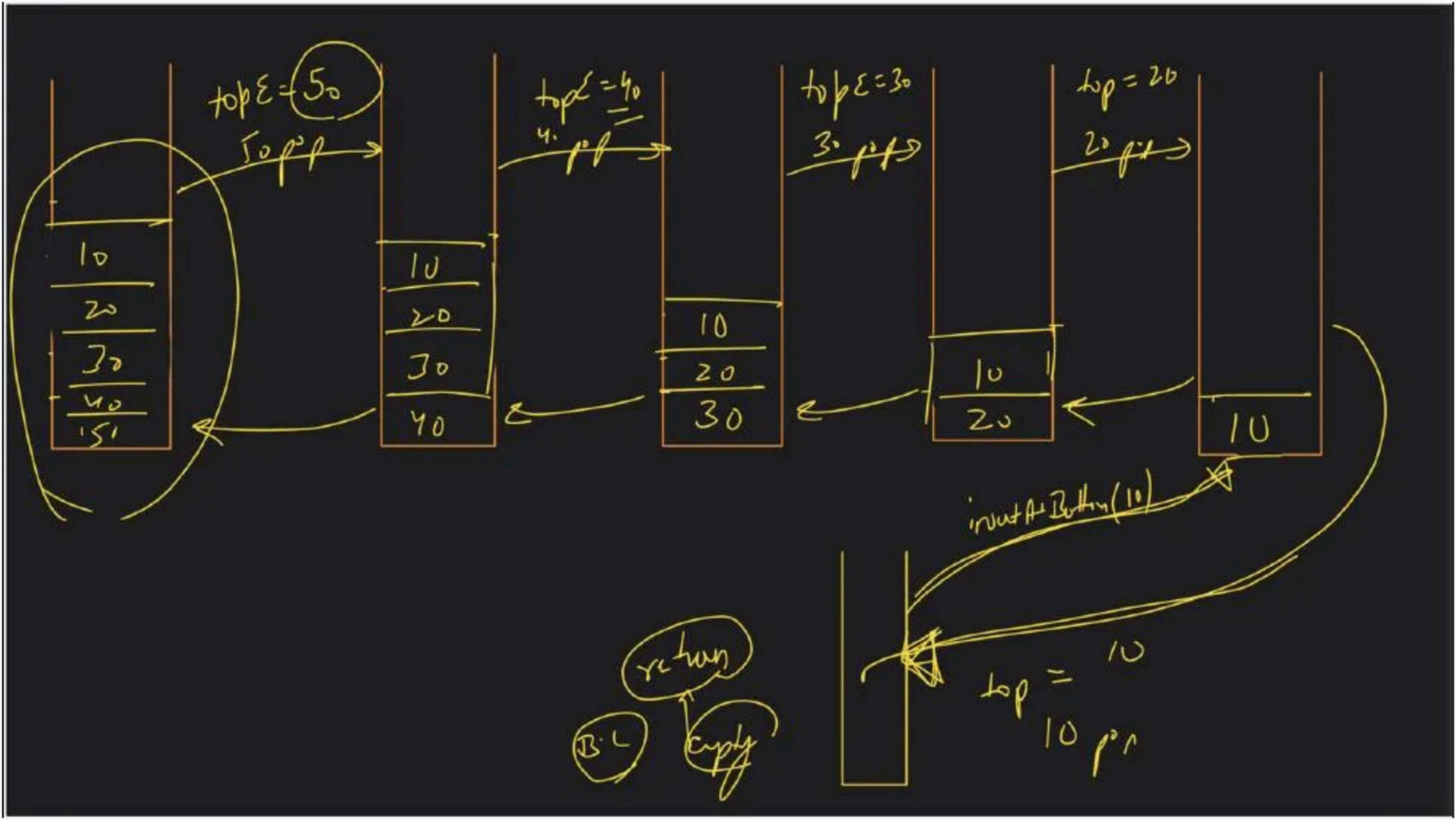

elem1 = INT_MAX
elem2 = 50
50 < INT_MAX
elem1 = 50
elem2 = 40
40 < 50
elem1 = 40
elem1 = 30
30 < 40
elem1 = 30
elem1 = 20
20 < 30
elem1 = 20
elem2 = 10
50
40
30
20
10
T
T
T
T
T
True
empty
P.C -> sorted stack

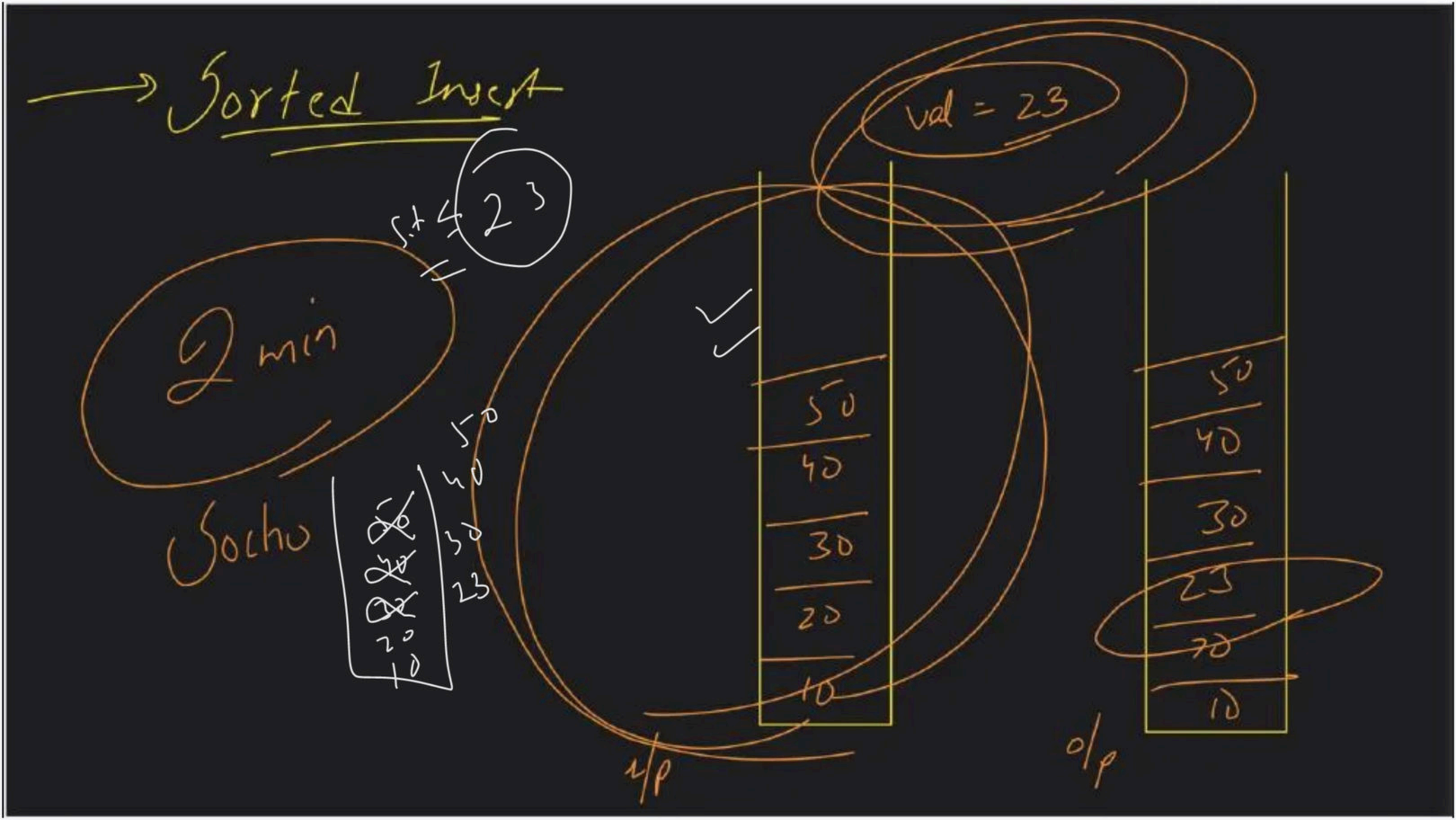

e1=INT_MAX
e2
10
e1 = 10
e2 = 30
40 < INT_MAX
30 < 40
e1 = 30
e2 = 20
20 < 30
e1 = 20
e2 = 10
40
30
20
10
20
10
20
10
10
T
F
F
F
T
10 < 20
empty

Reverse a stack

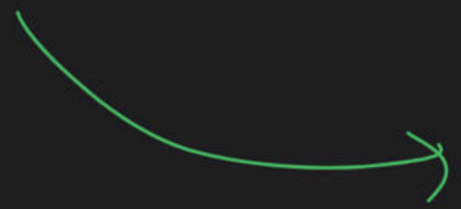



1
2
3
4
5
A
C
B
Reverse


top = 50
top = 40
top = 30
top = 20
10
20
30
40
50
10
20
30
40
10
20
30
10
20
10
insertAtBottom(10)
top = 10
return
B.C
Empty

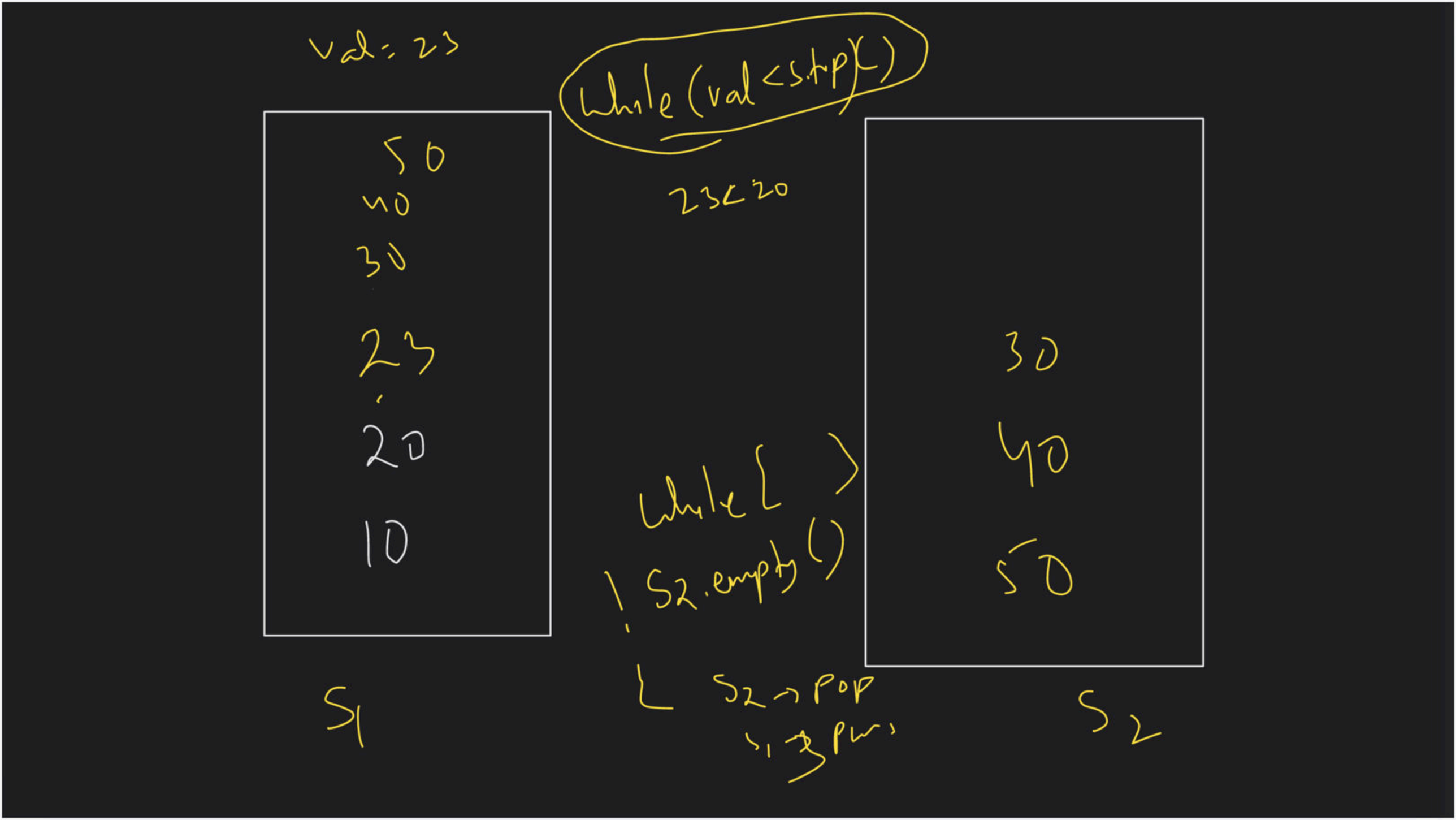

Sorted Insert
val = 23
sol = 23
2 min
Socho
50
40
30
20
10
23
i/p
o/p

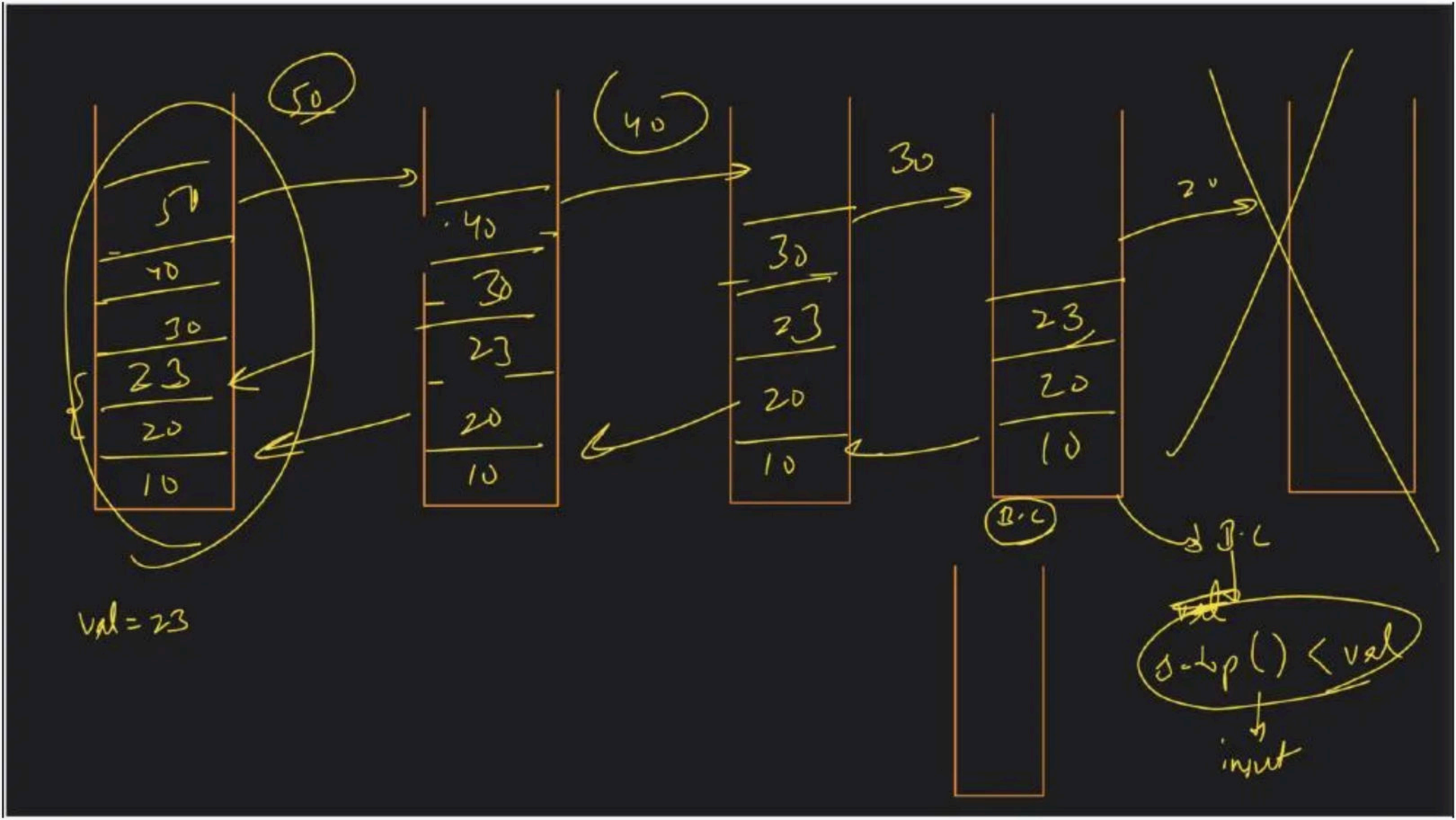
Sorted Insert



Header File

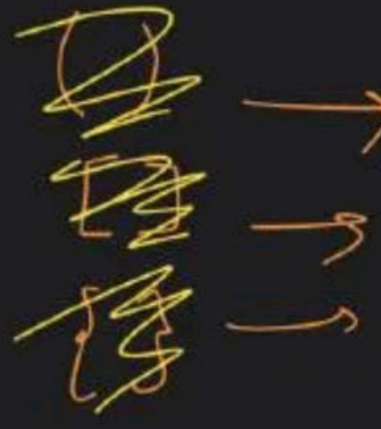
↳ Peanut Mazzetta

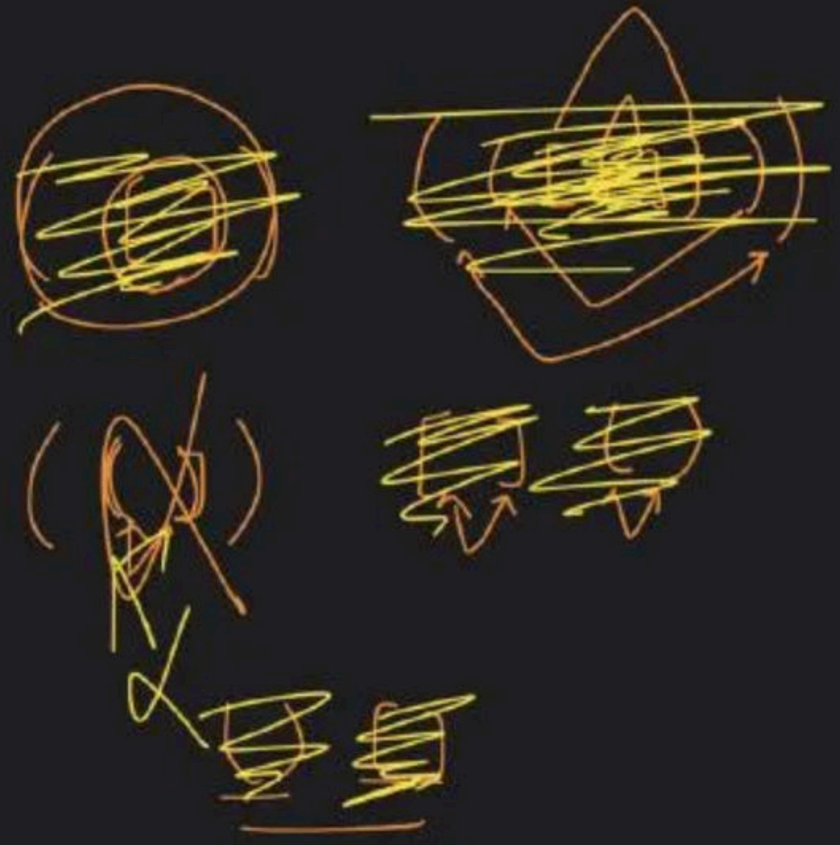

val: 23
while (val < s.top())
50
40
30
23
20
10
23 < 20
30
40
50
while (!S2.empty())
S2 -> pop
S1
S2

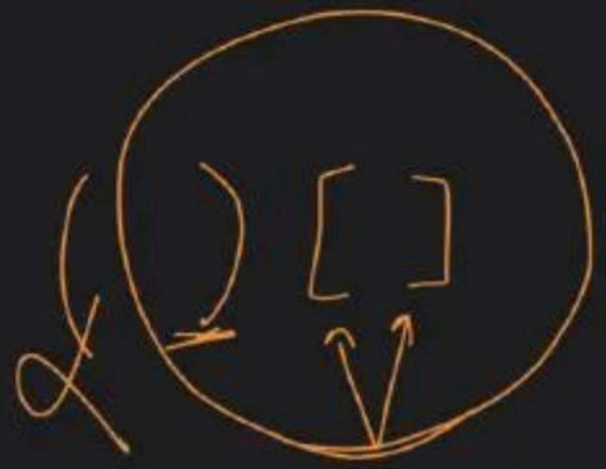

50
40
30
20
10
val = 23
s.top() < val
input

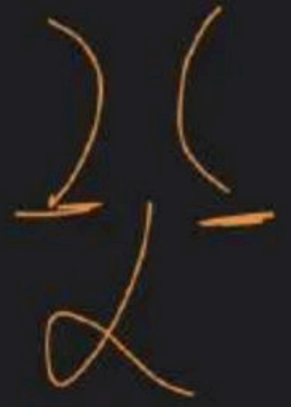
Valid Parenthesis

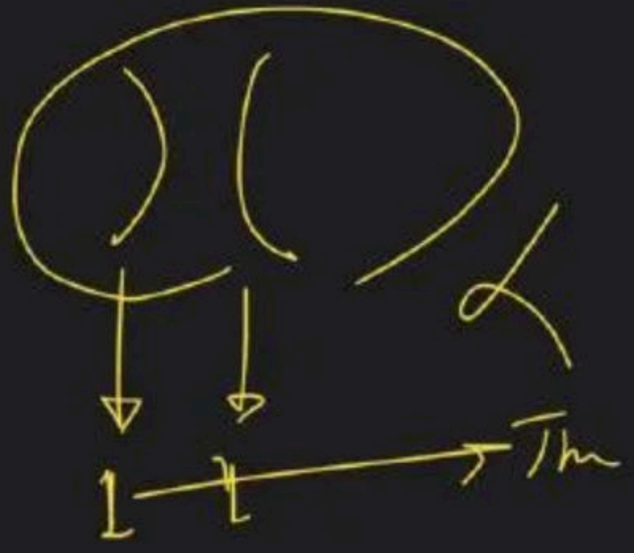
2 min Break



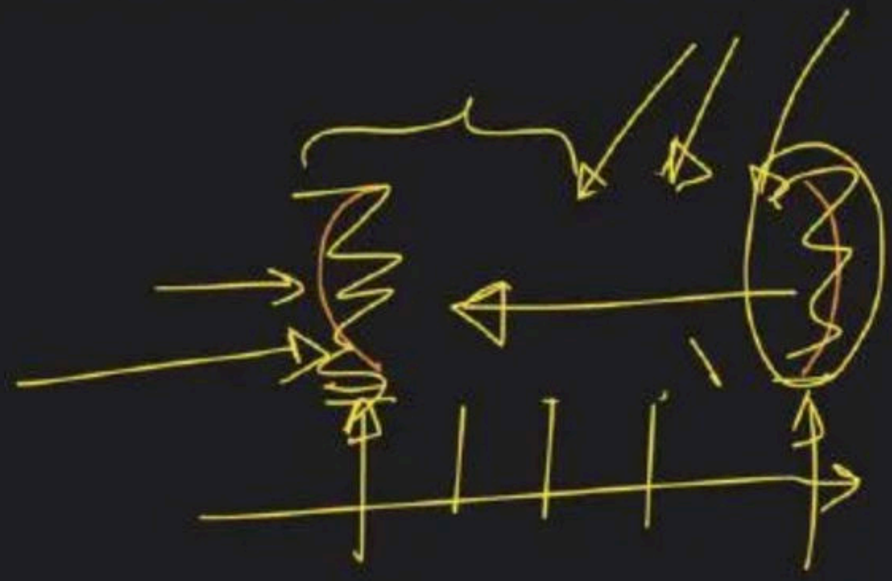


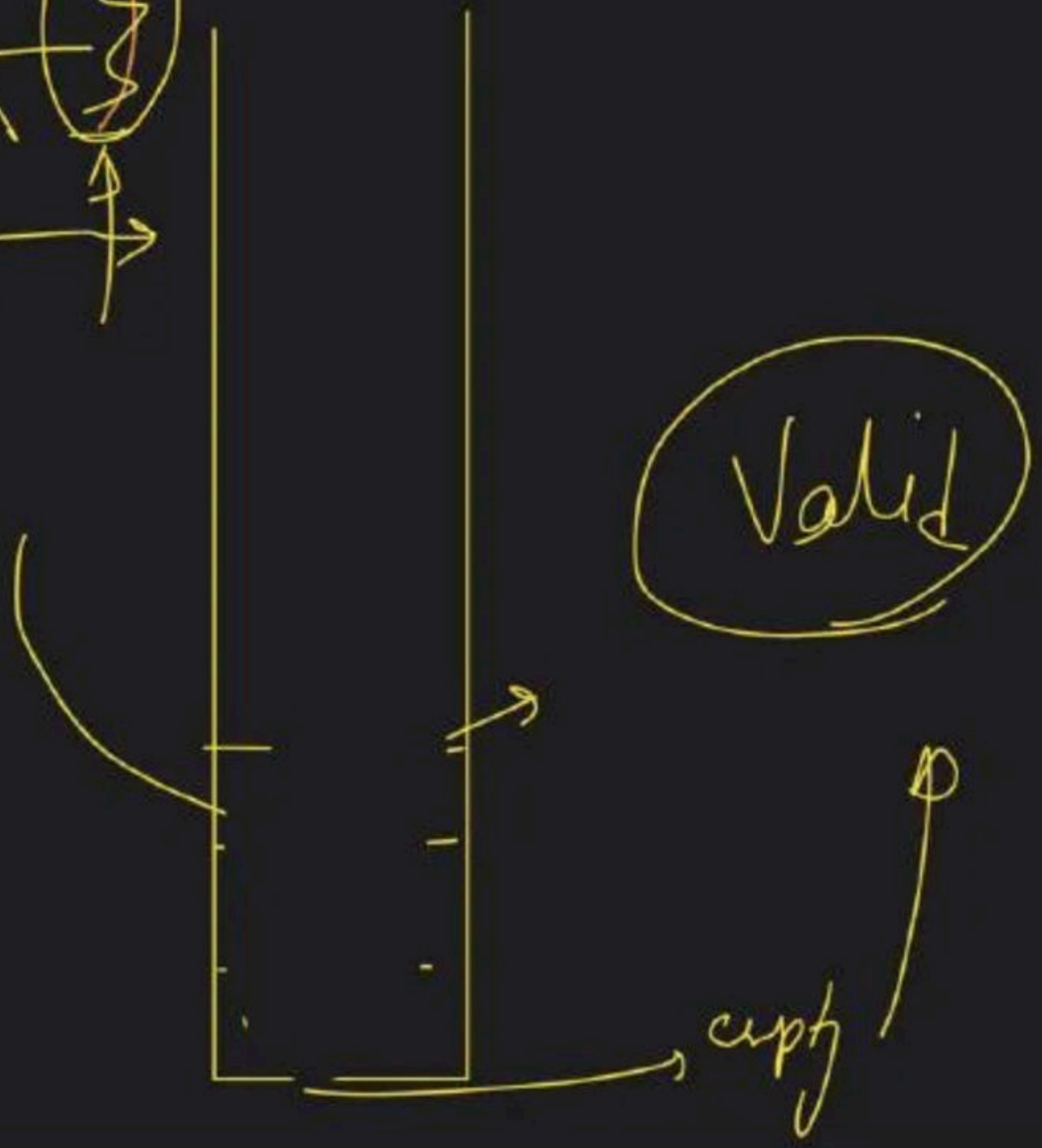





1
Tm

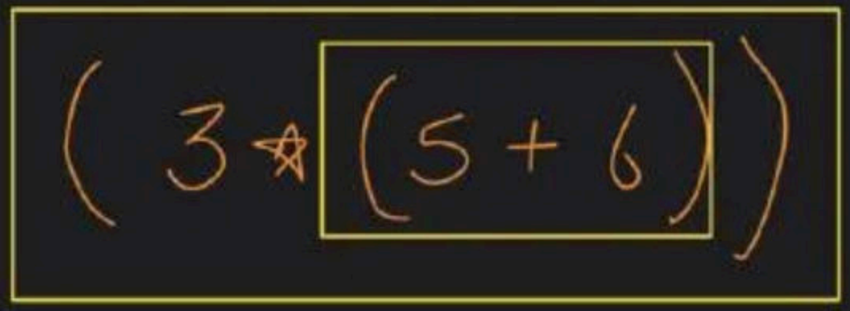
How ?





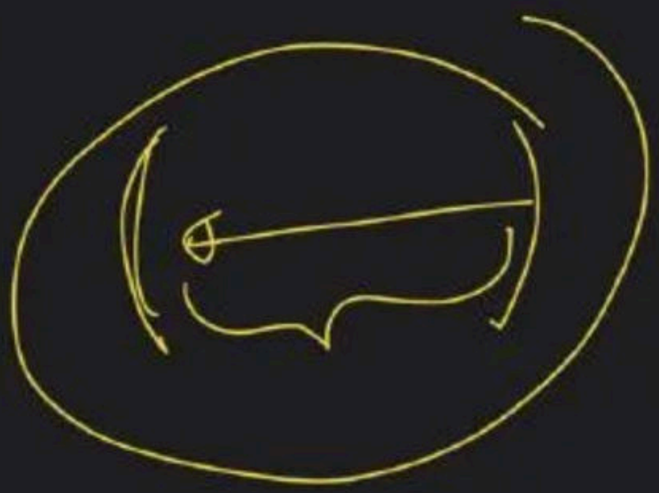
Check Redundant Brackets

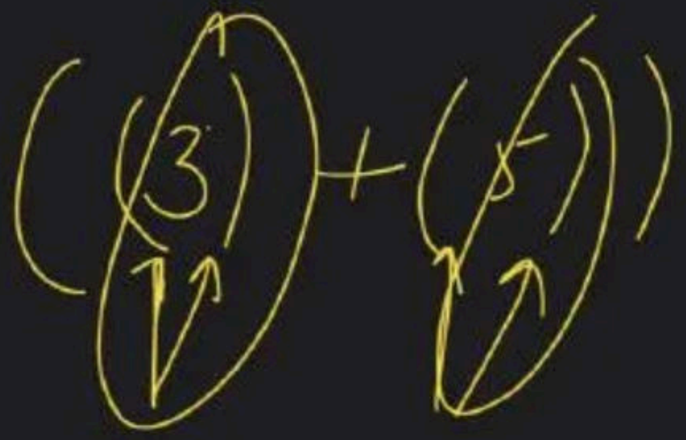
i/p →



$(3 * (5 + 6))$

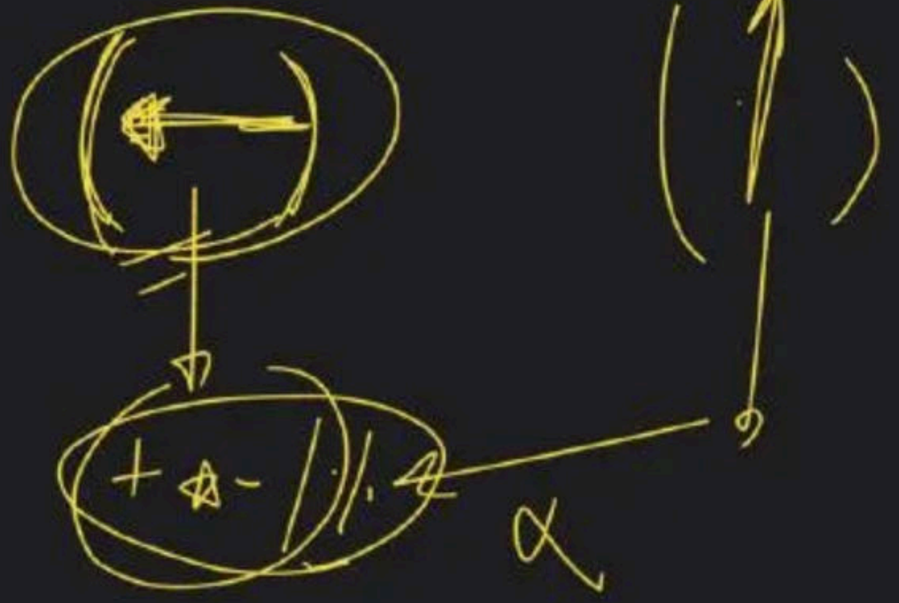
3





2 pair
↳ of Rcv

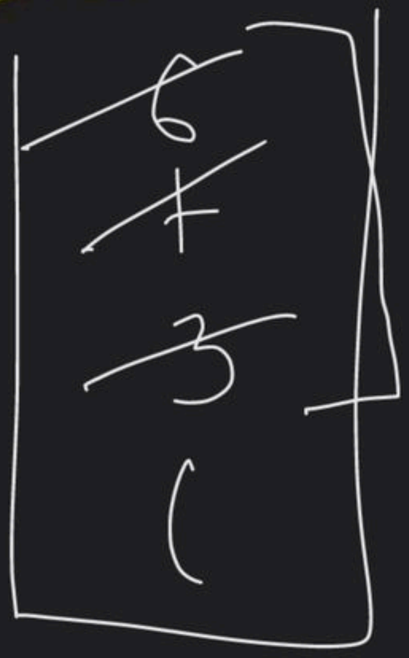
(3 + 5)



$\left(\left(\frac{1}{3}\right) + \left(\frac{1}{5}\right)\right)$

$(3$

$(3+6$



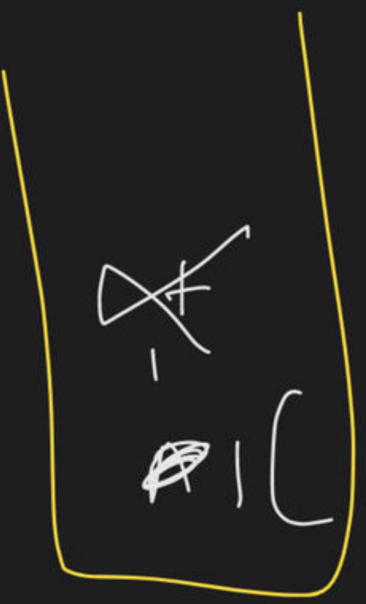
Time

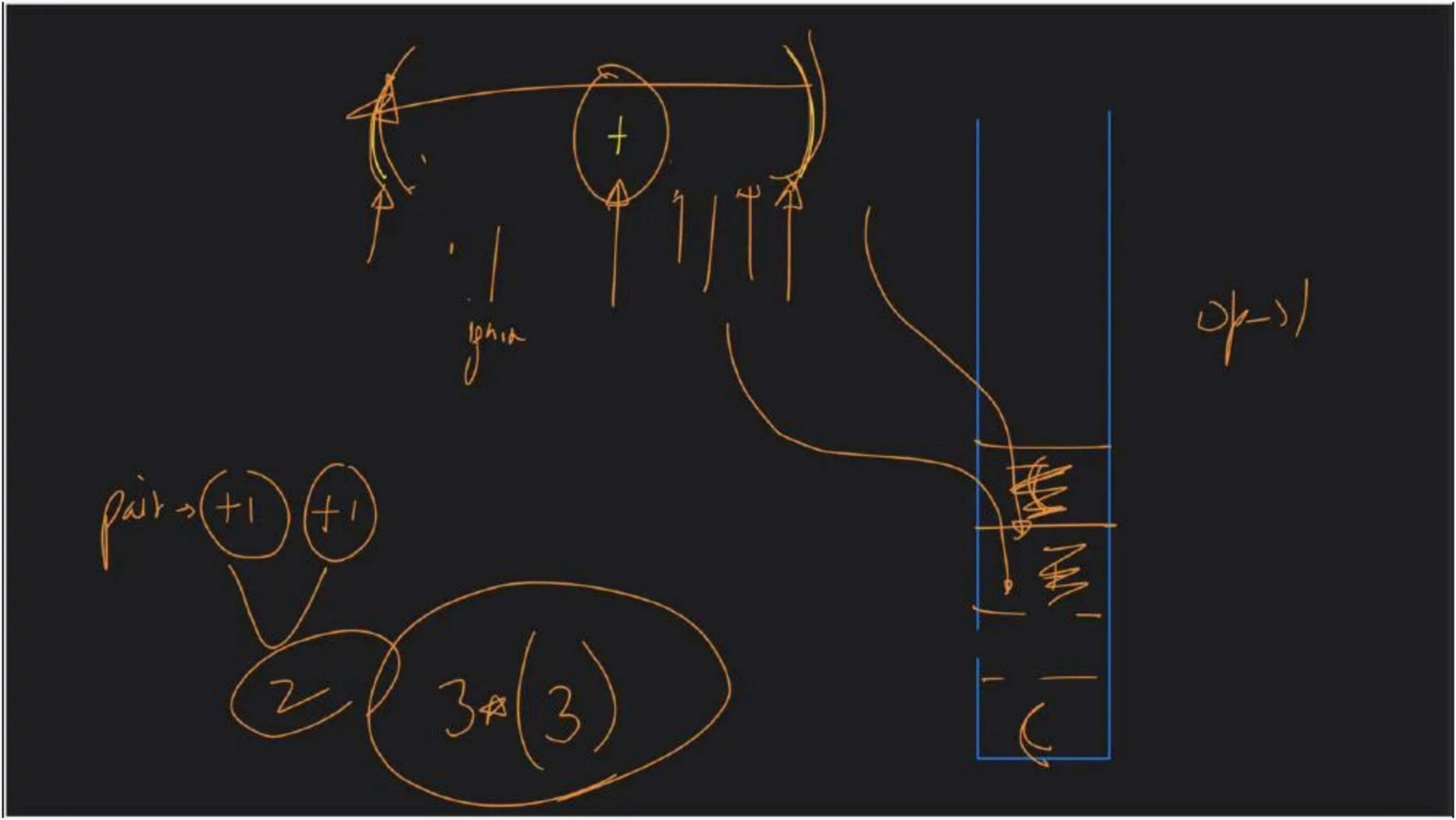
(A + (B + C))

if (ch == ')')

{

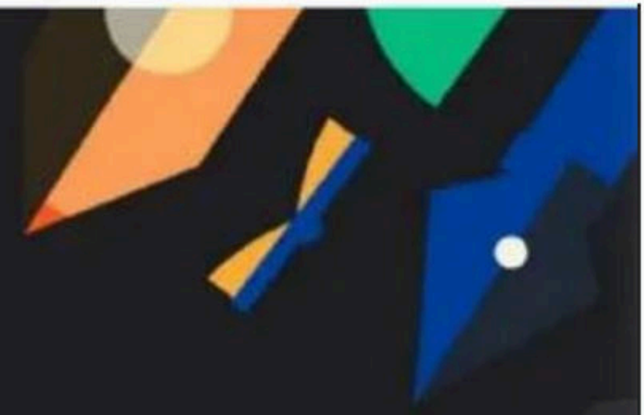
checking




+
pair → +1 +1
2
3*(3)

Kal

8:30 pm



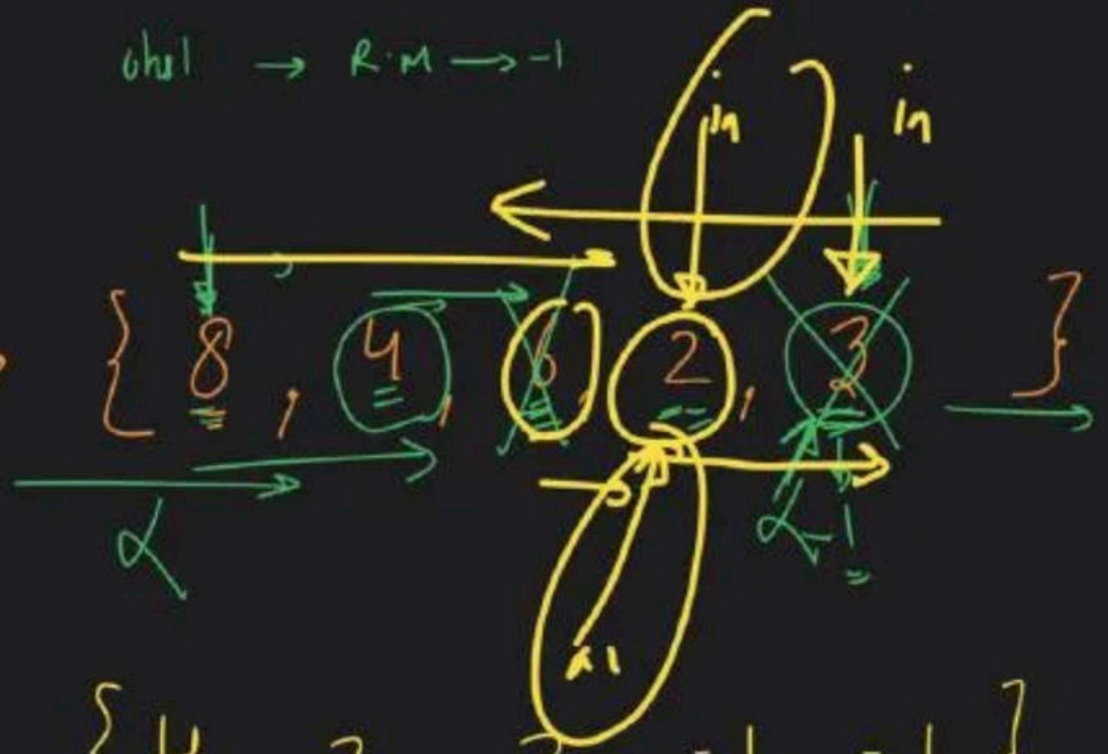
Stack Class - 3

Special class

Love Babbar • Aug 1, 2024

→ Next smaller element

i/p → arr → { 8, 4, 6, 2, 3 }



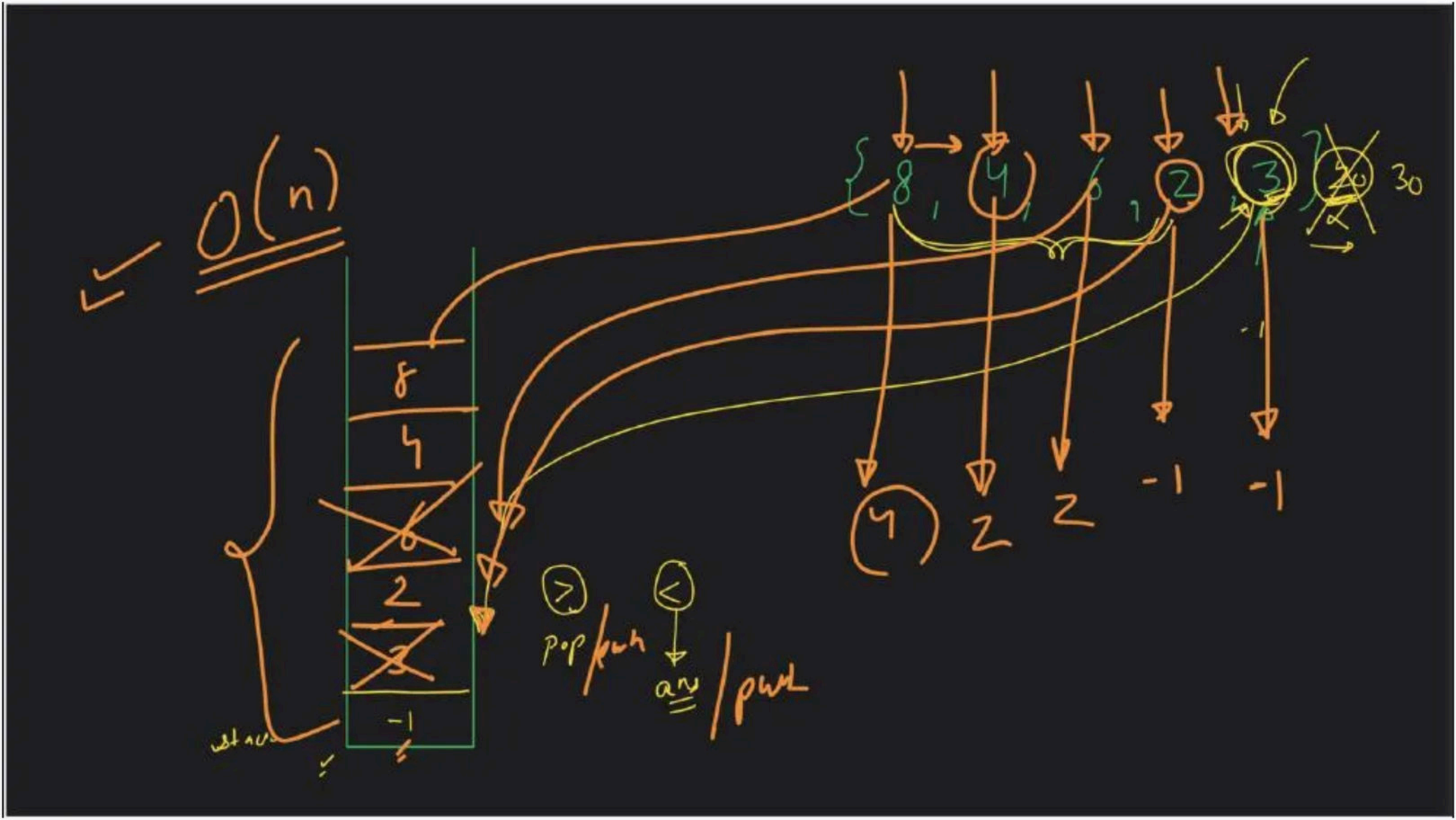
o/p → { 4, 2, 2, -1, -1 }

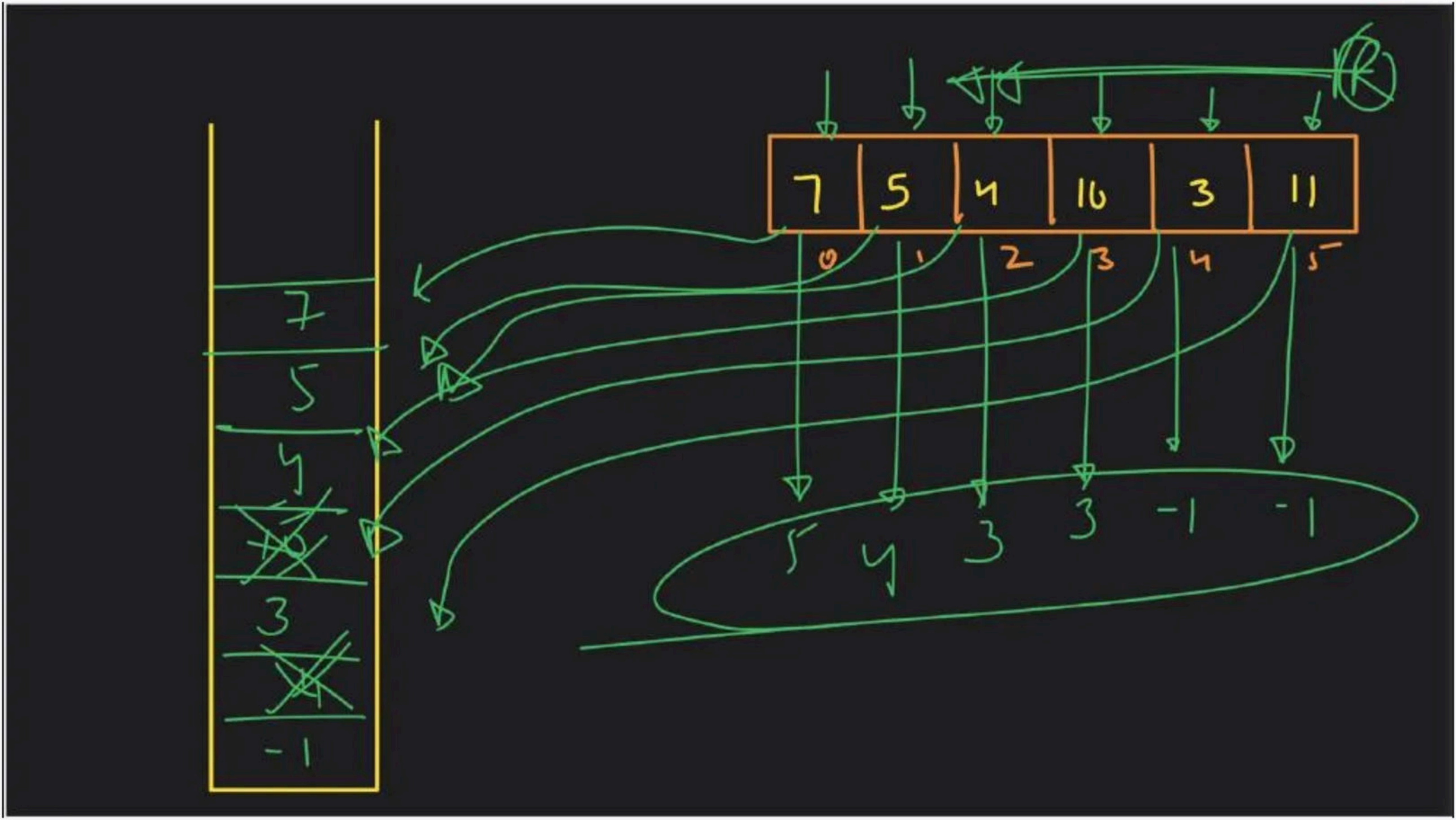
Approach

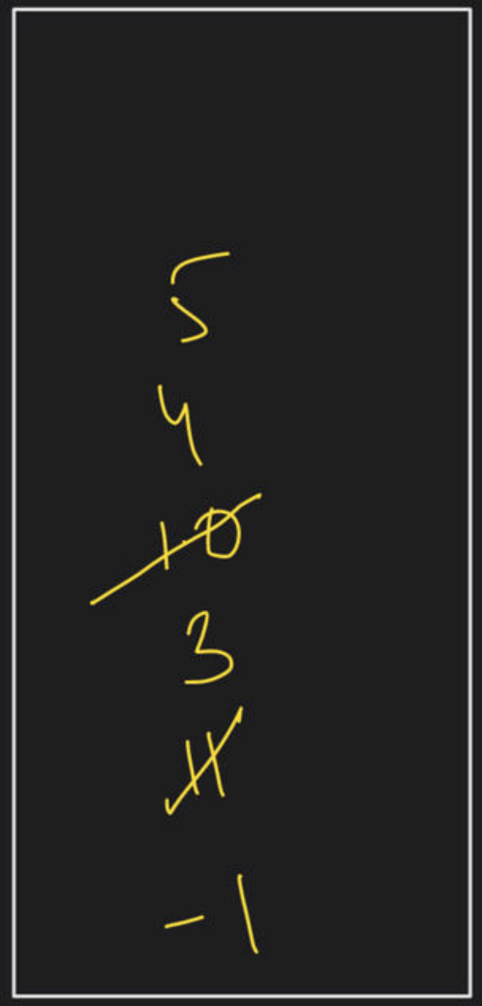
B·F → 2 Loop → O(n²)

O(n) → single loop

Trick


O(n)
8
4
2
-1
pop/push
any/push


7
5
4
16
3
11
0
1
2
3
4
5
7
5
4
3
-1
3
-1
5
4
3
3
-1
-1



7	5	4	10	3	11
0	1	2	3	4	5

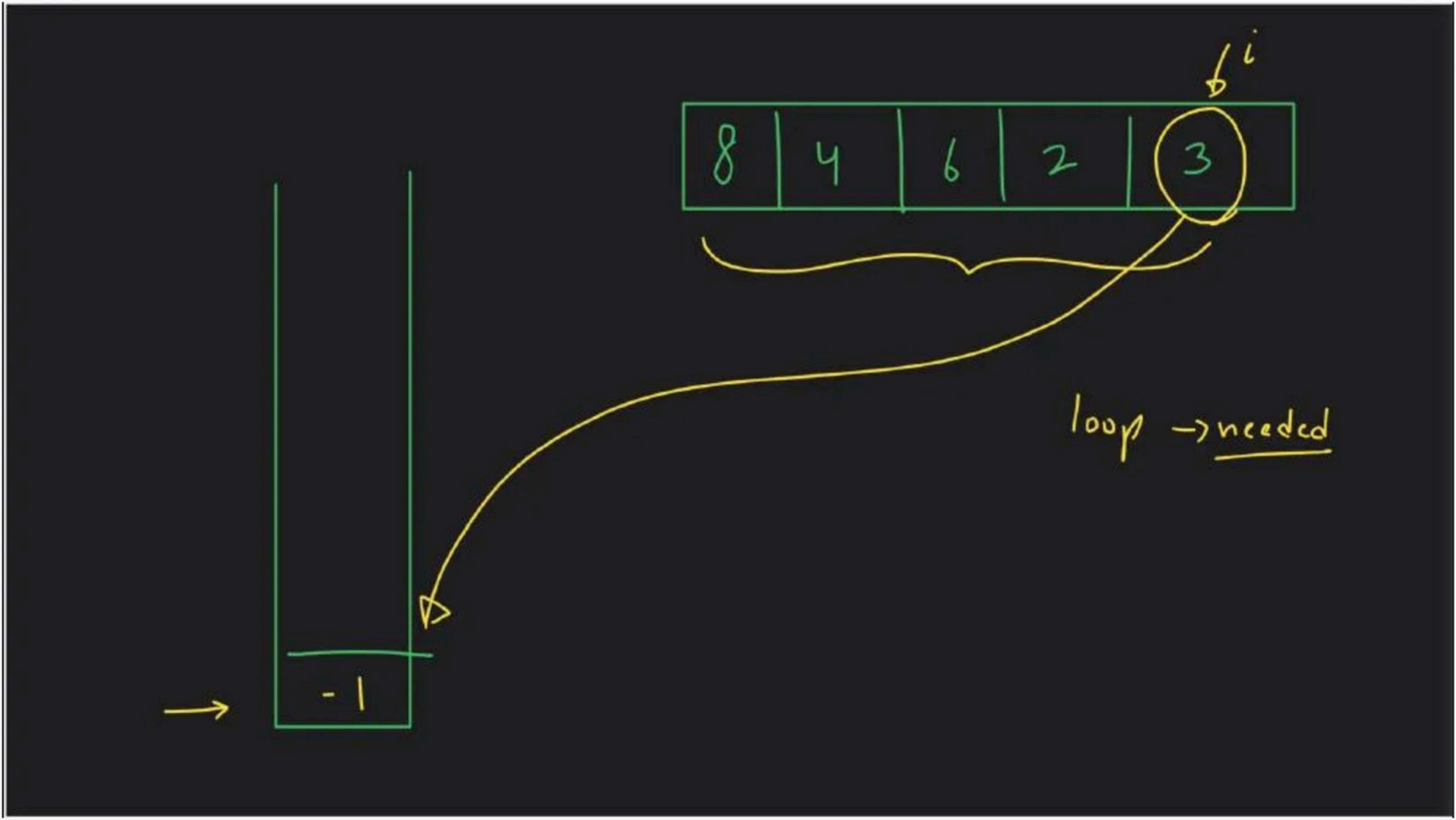
5 4 3 3 -1 -1

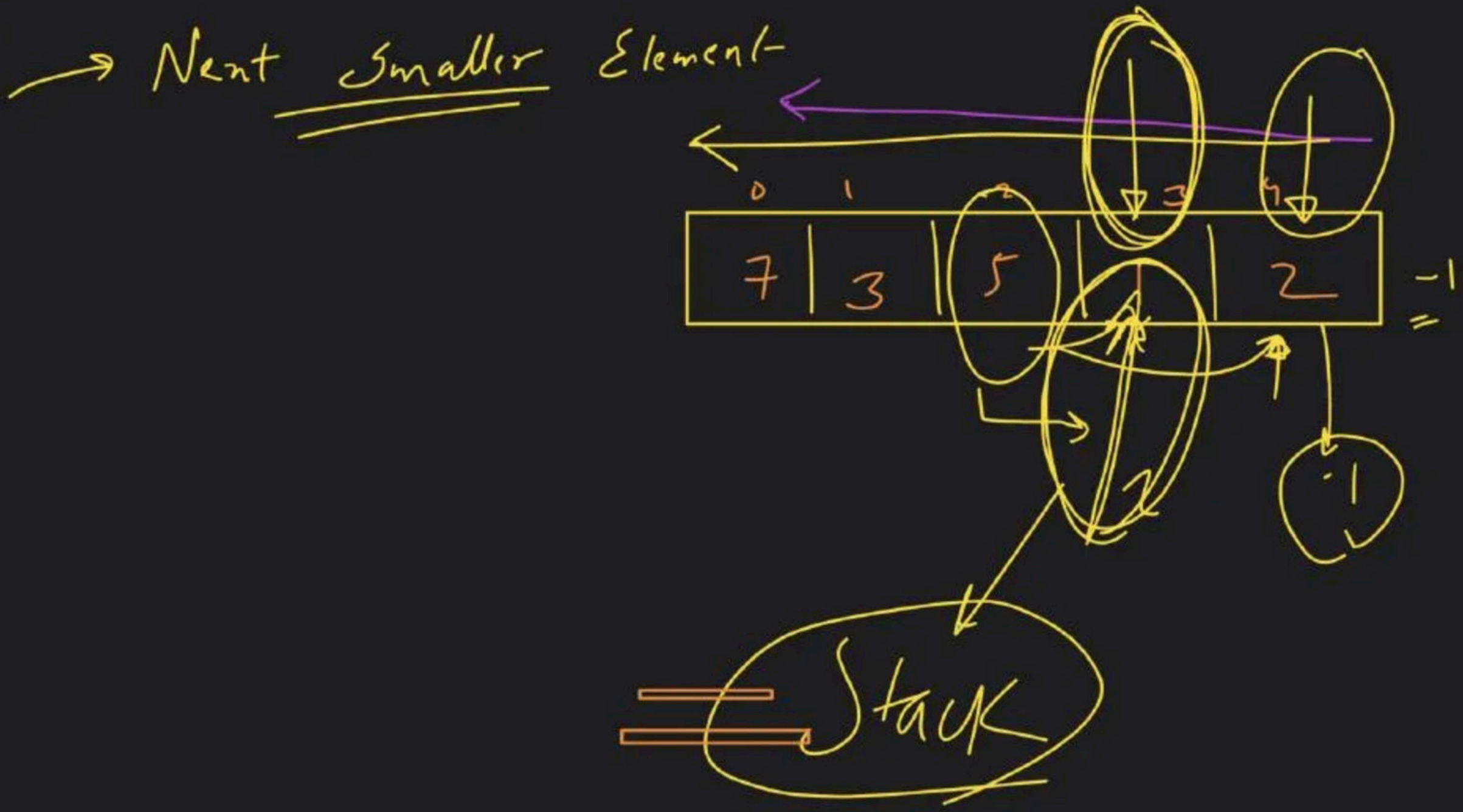
① stop >= elemt
↳ P.OP ⇒

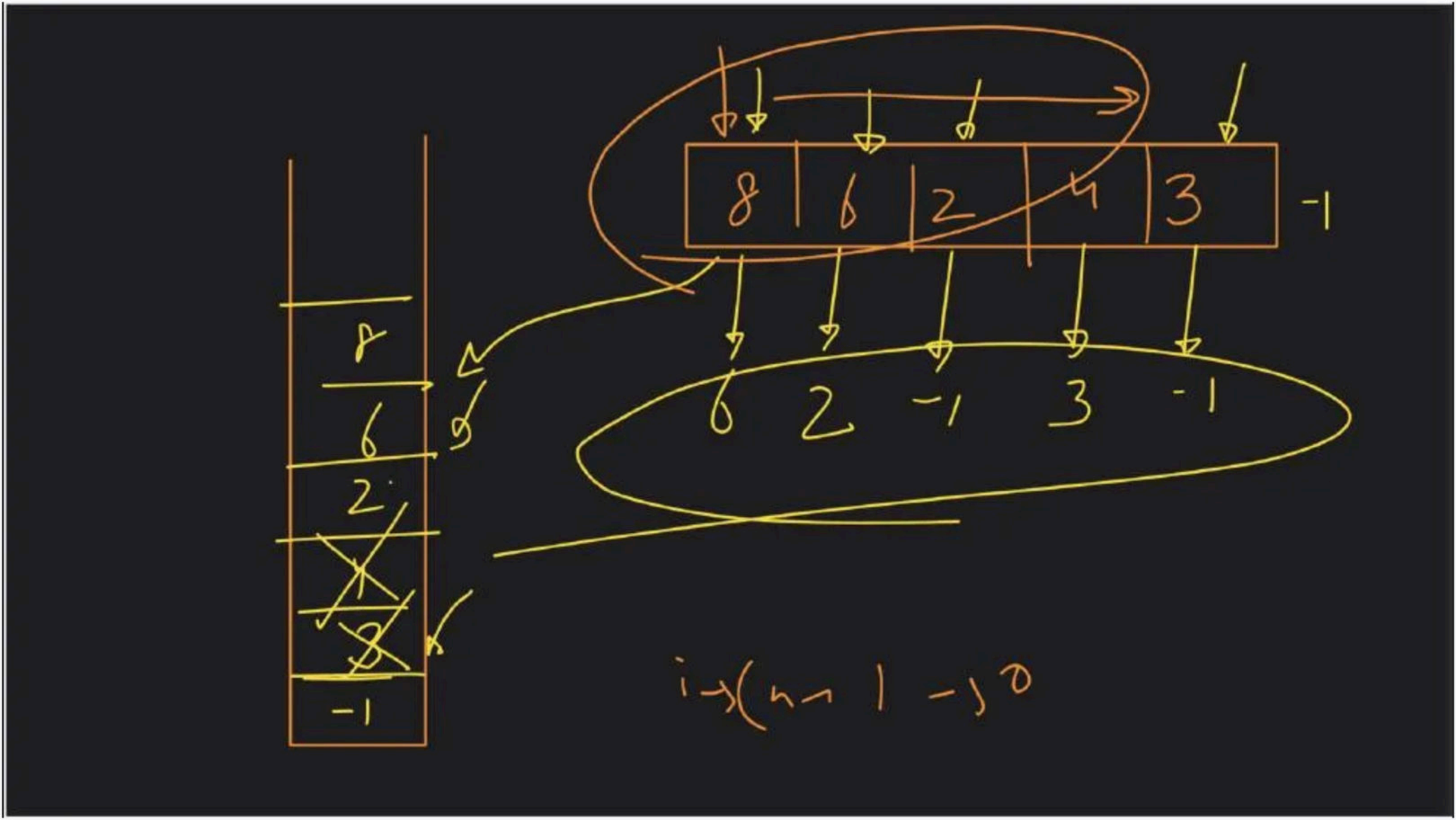
② i am update -
st. Push

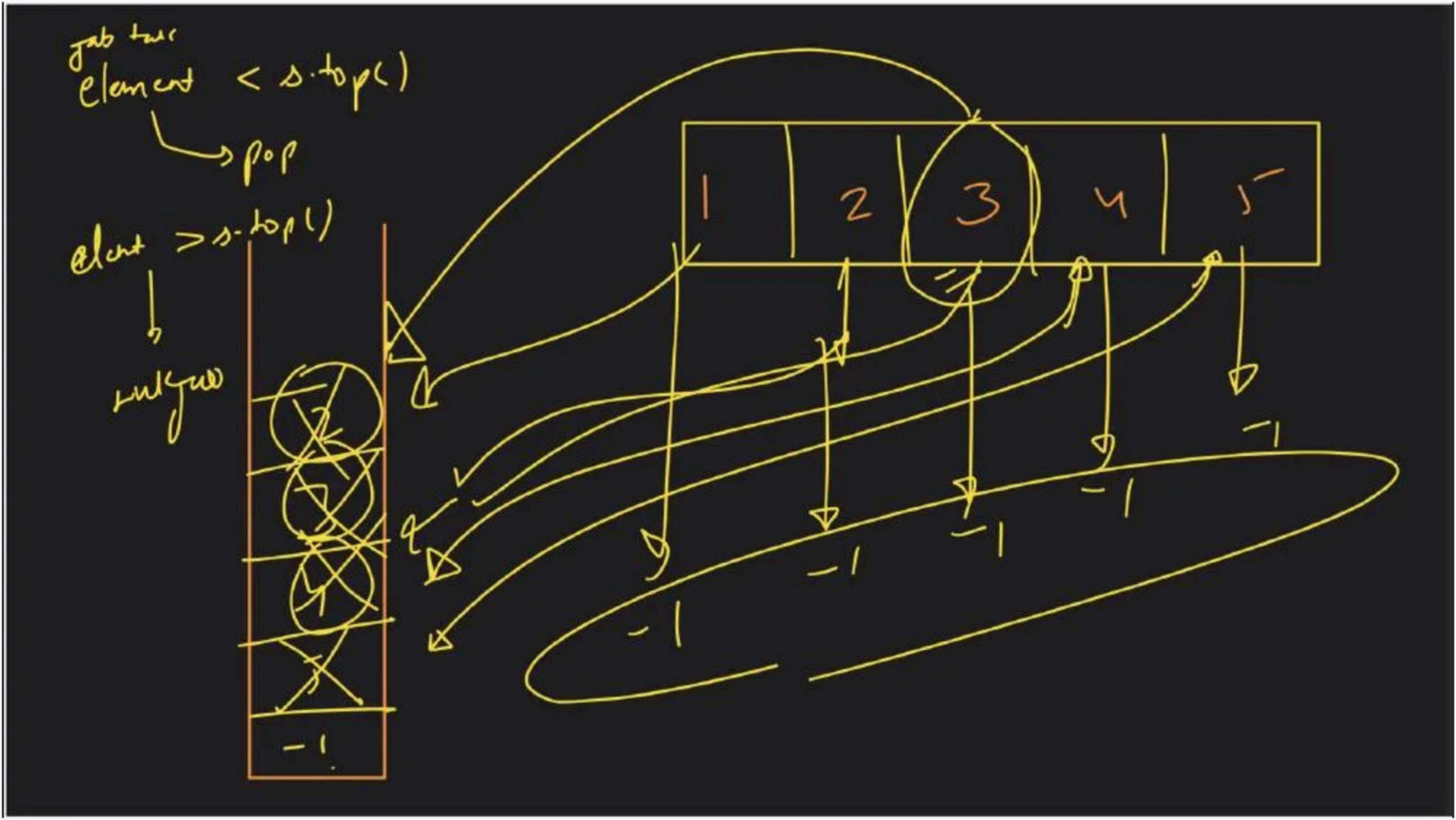
7	5	4	3	2	1
0	1	2	3	4	5

No. of operations

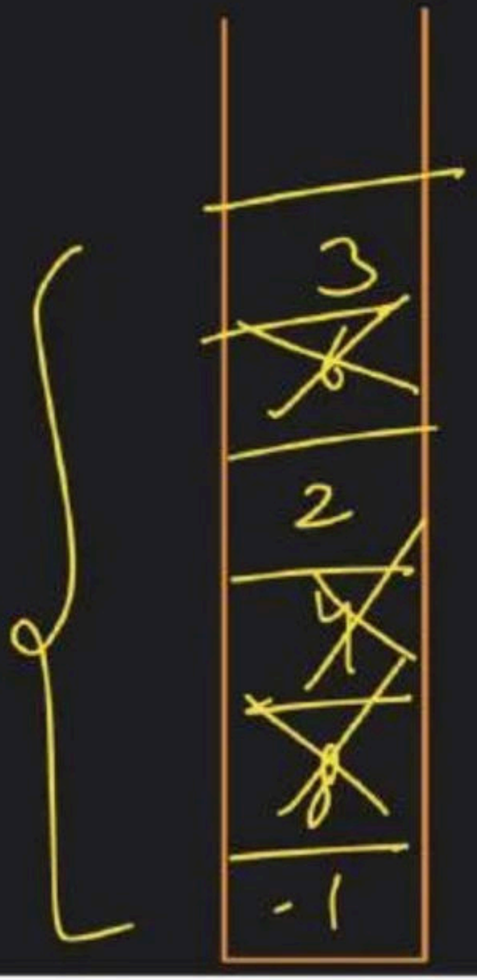

i
8
4
6
2
3
-1
loop → needed

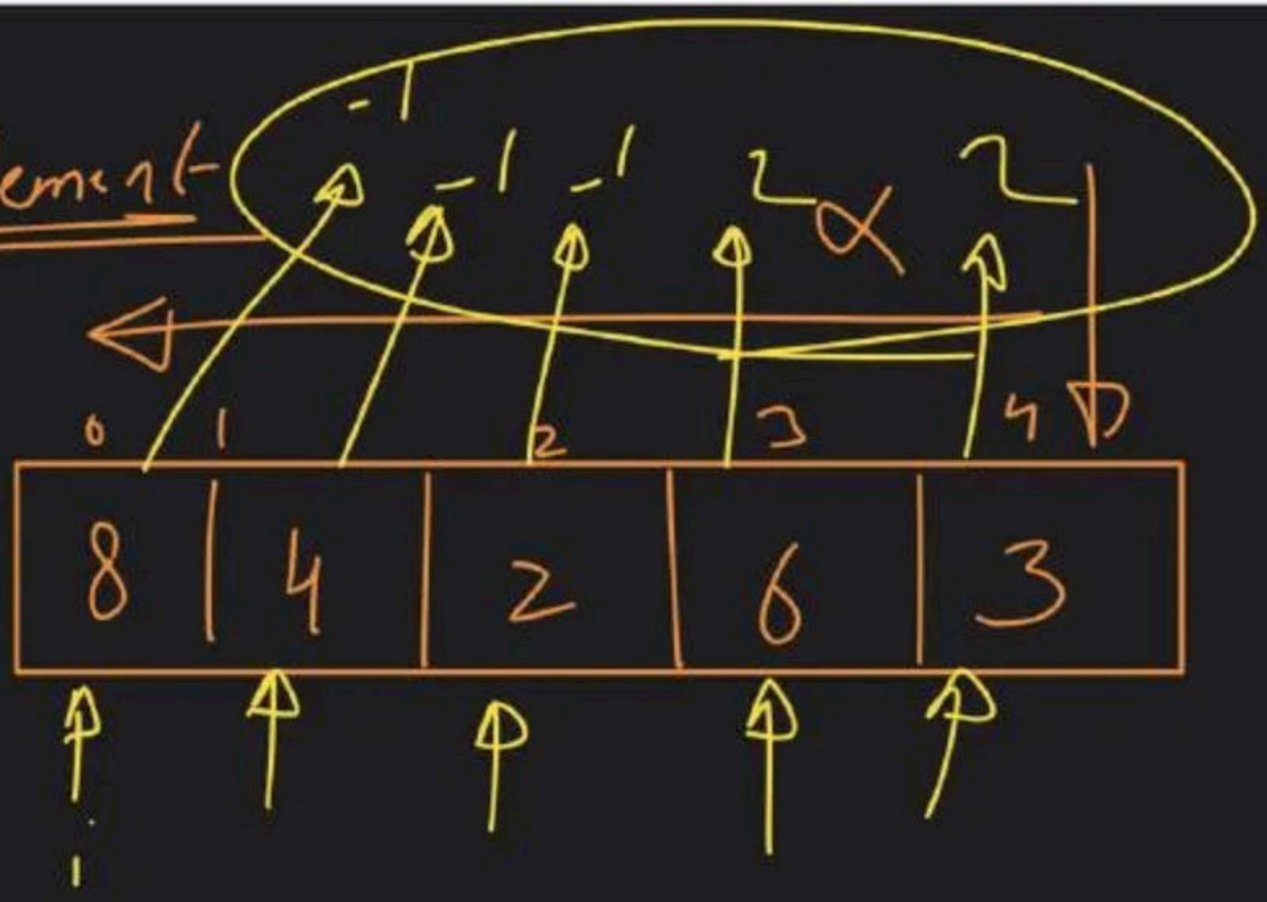

Next Smaller Element-
0
1
2
3
4
7
3
5
2
-1
-1
Stack

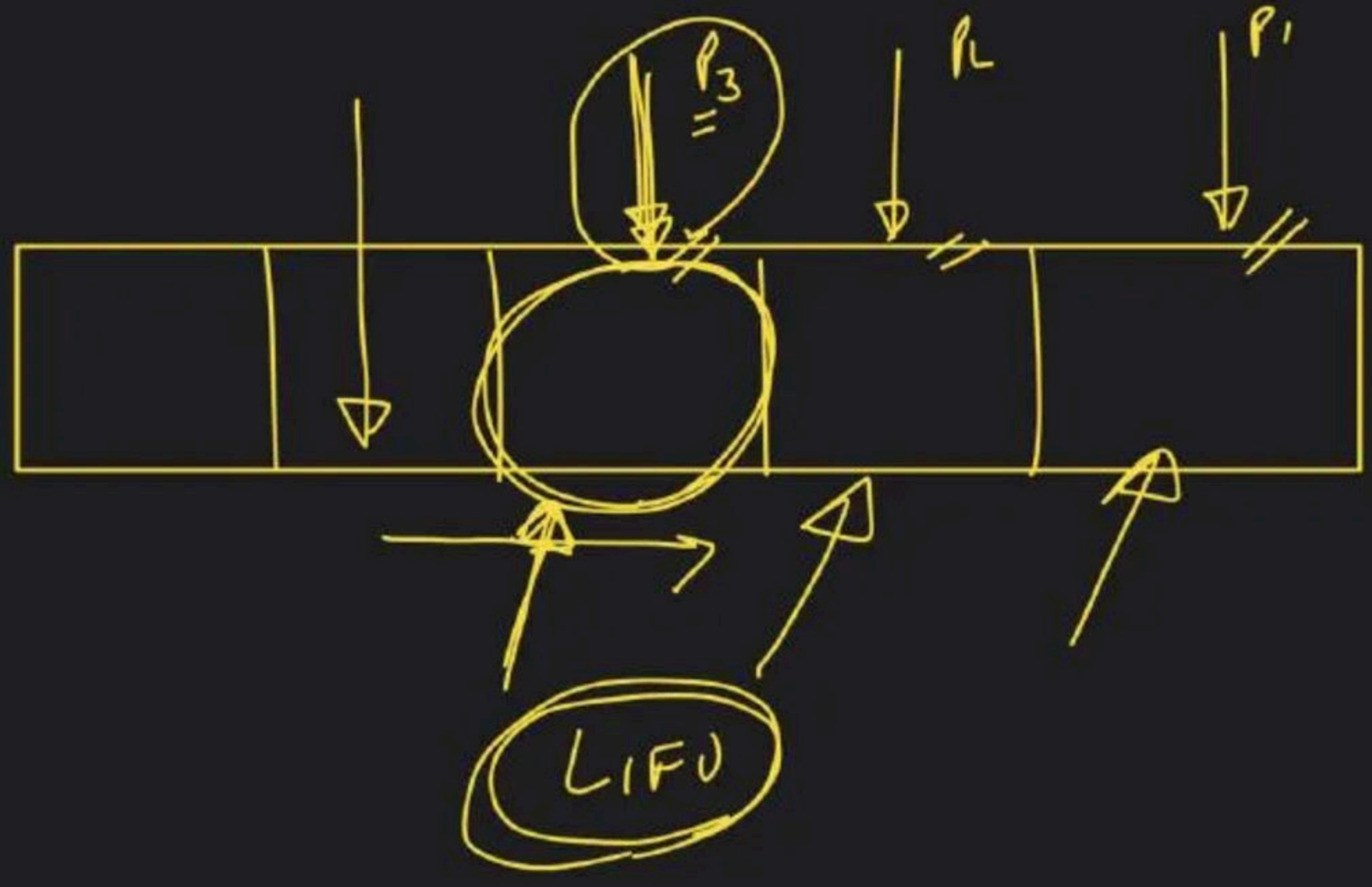

8
6
2
4
3
-1
8
6
2
-1
3
-1
8
6
2
-1
i->(n-1->0


jab tak
element < s.top()
pop
element > s.top()
1
2
3
4
5
-1
-1
-1
-1
-1
-1

→ Prev Smaller Element

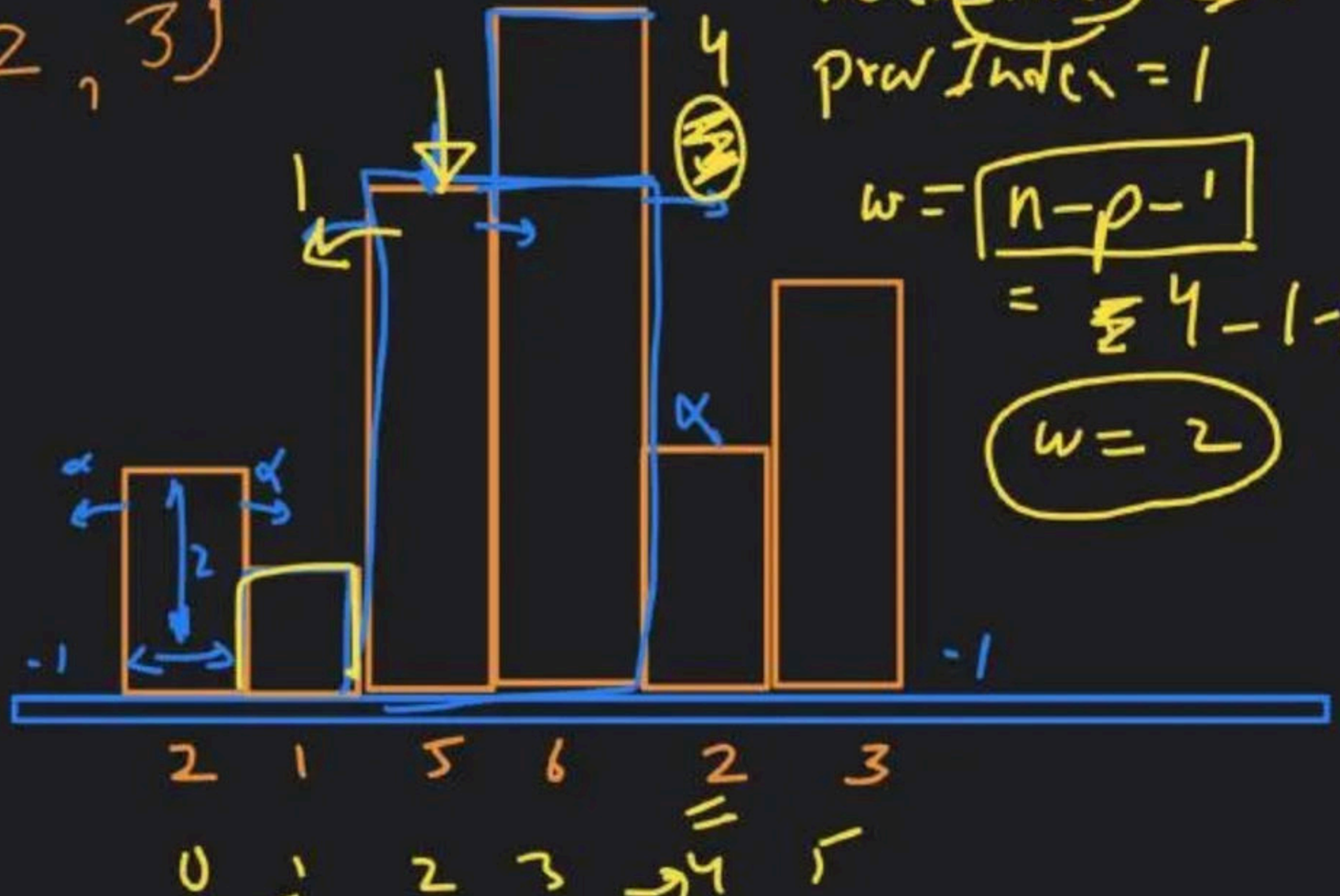





P3
PL
P1
LIFU

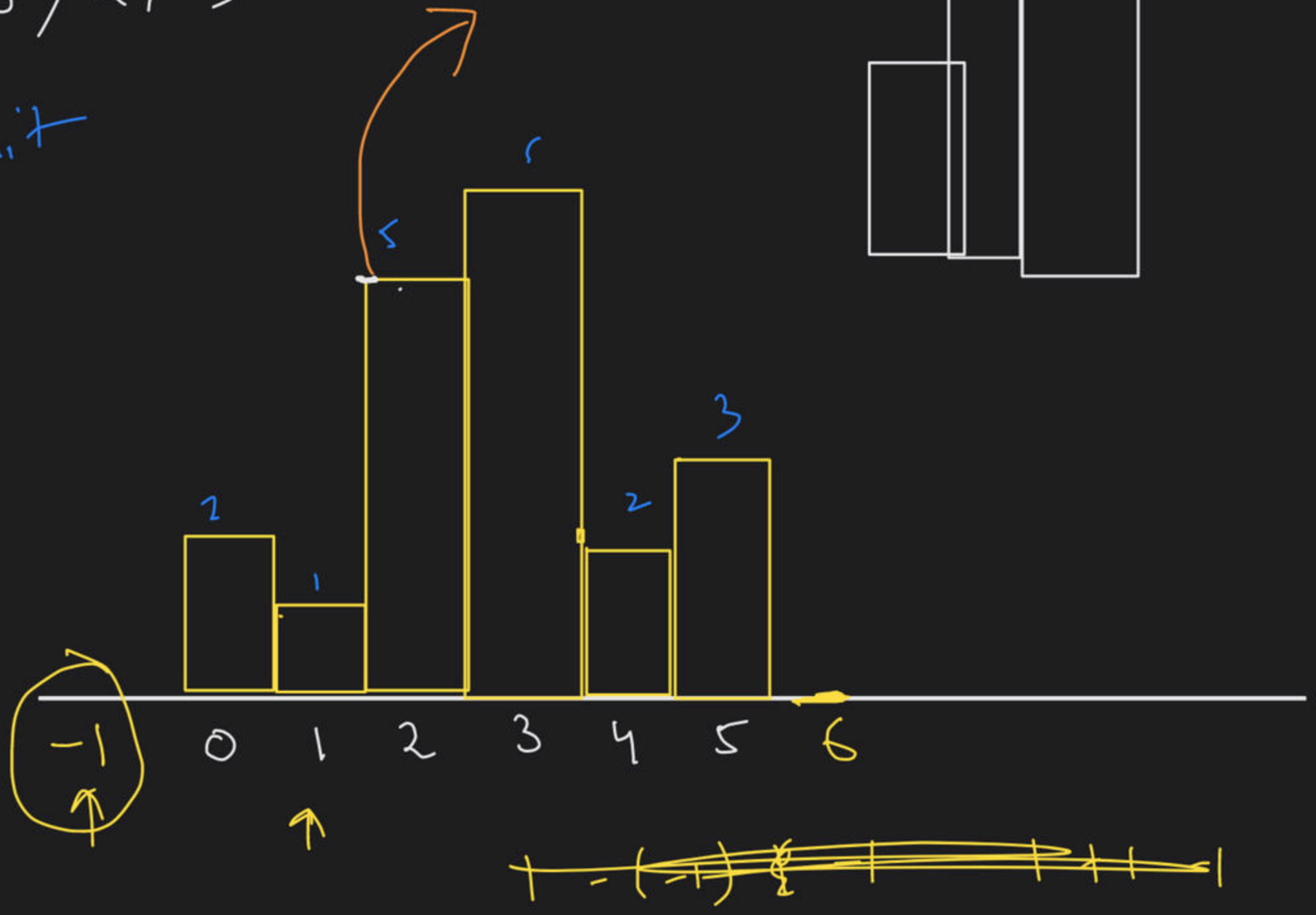
→ Largest Rectangular area in Histogram

ip-> {2, 1, 5, 6, 2, 3}



2, 1, 5, 6, 2, 3

↔ width -> 1 unit



	2	1	5	6	2	3
	0	1	2	3	4	5
next =	1	6	4	4	6	6
prev =	-1	-1	1	2	1	4

(0) ⇒ w = 1 − (−1) − 1 ⇒ 1 + 1 − 1 = 1

h = 2

A = 2

(1) ⇒ w = (6) − (−1) − 1

= 6 + 1 − 1 = 6

h = 1 ⇒

A = 6

(2) $w = 4 - 1 - 1 = 4 - 2 = 2$

$n = 5 \Rightarrow A = 10$ Ans

(3) $w = 4 - 2 - 1 = 4 - 3 = 1$

$n = 6 \Rightarrow A = 6$

(4) $w = 6 - 1 - 1 = 6 - 2 = 4$

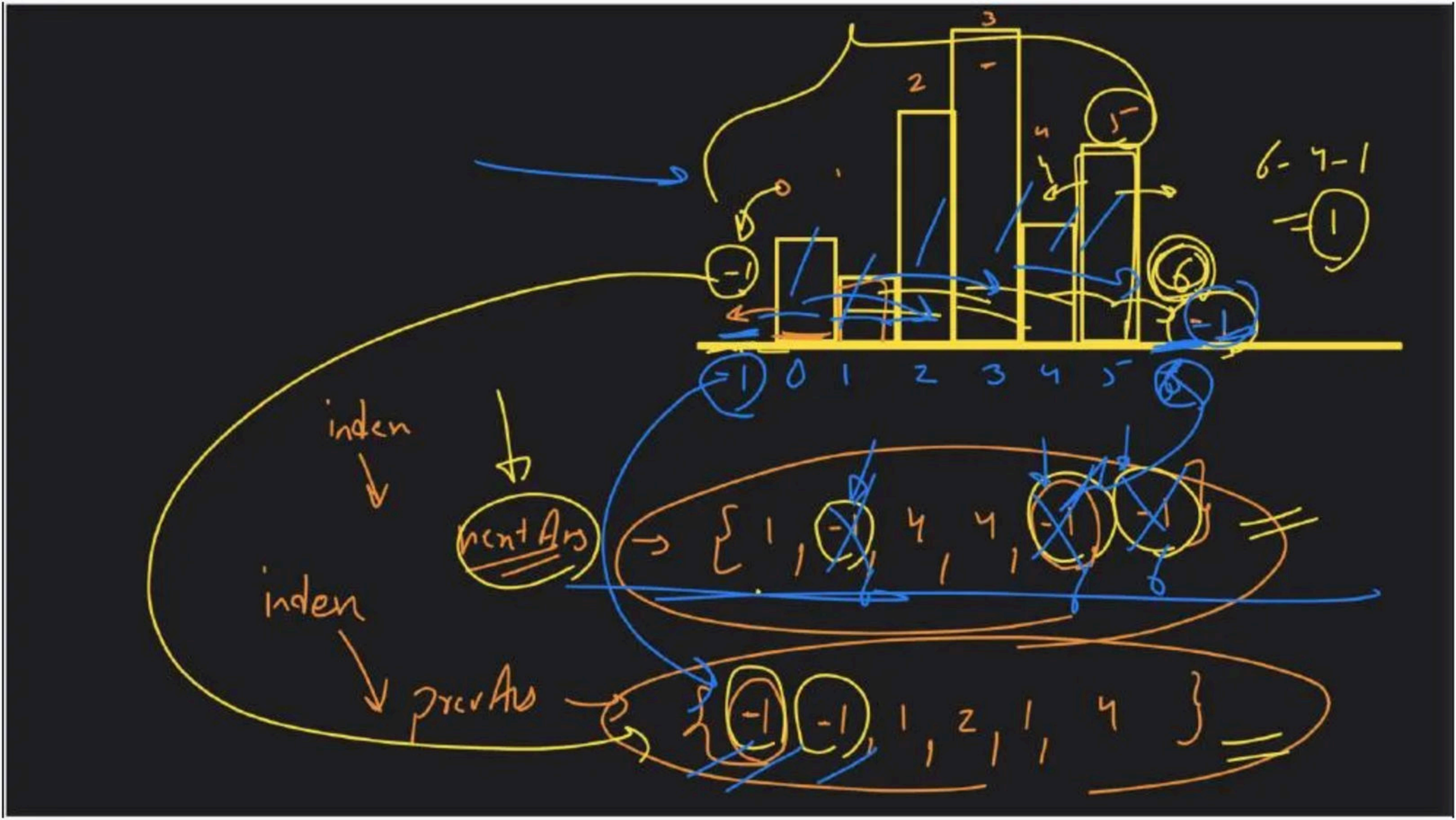
$n = 2 \Rightarrow A = 8$

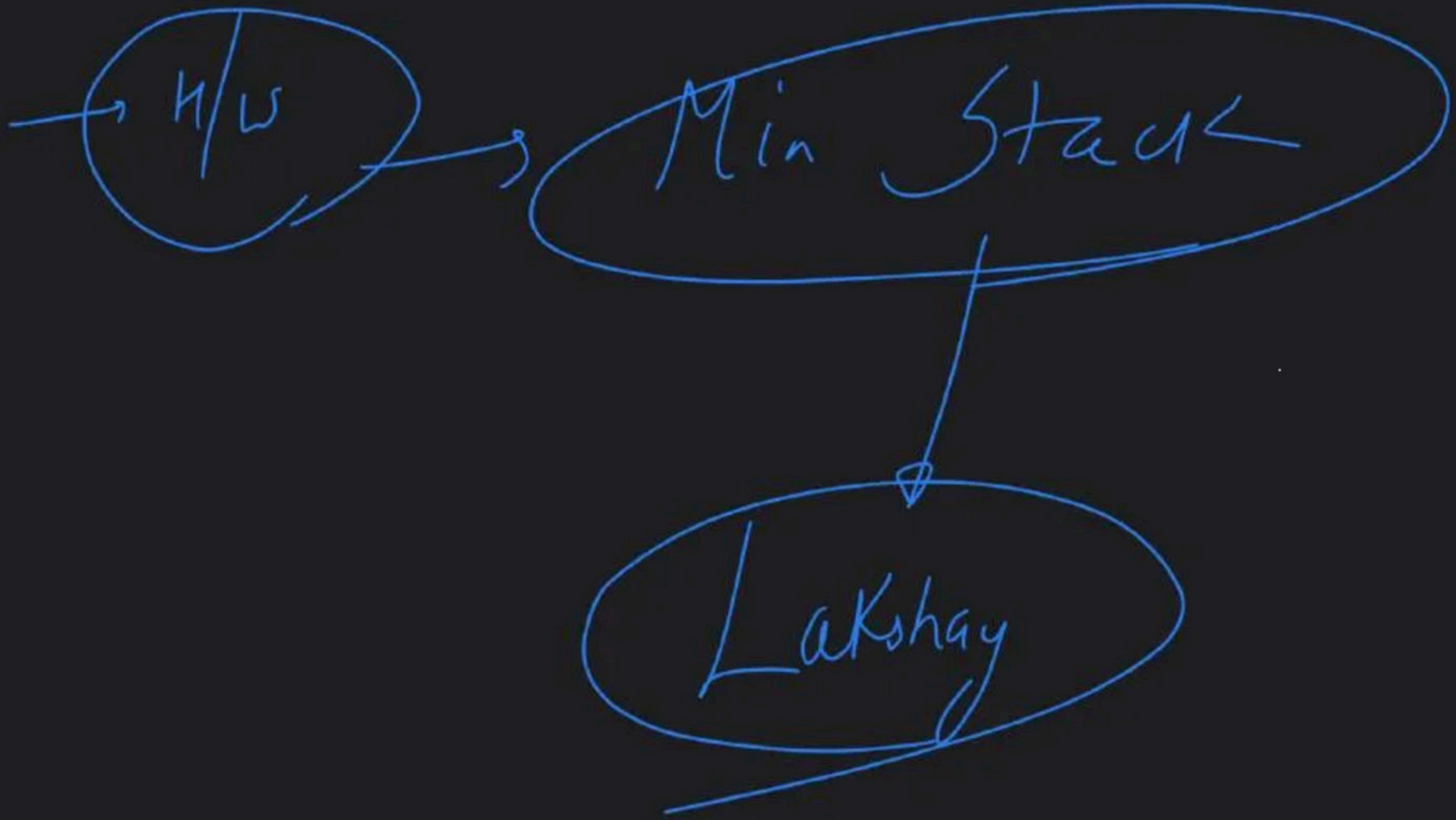
6

$W = -4 - 1 = 6 - 5 = 1$

$H = 3$

$A = 3$


3
2
4
5
6-4-1
-1
6
-1
-1
0 1 2 3 4 5
inden
nextAns
{ 1 , -1 , 4 , 4 , -1 , -1 }
inden
prevAns
{ -1 , -1 , 1 , 2 , 1 , 4 }


H/W
Min Stack
Lakshay