Stack Class - 1

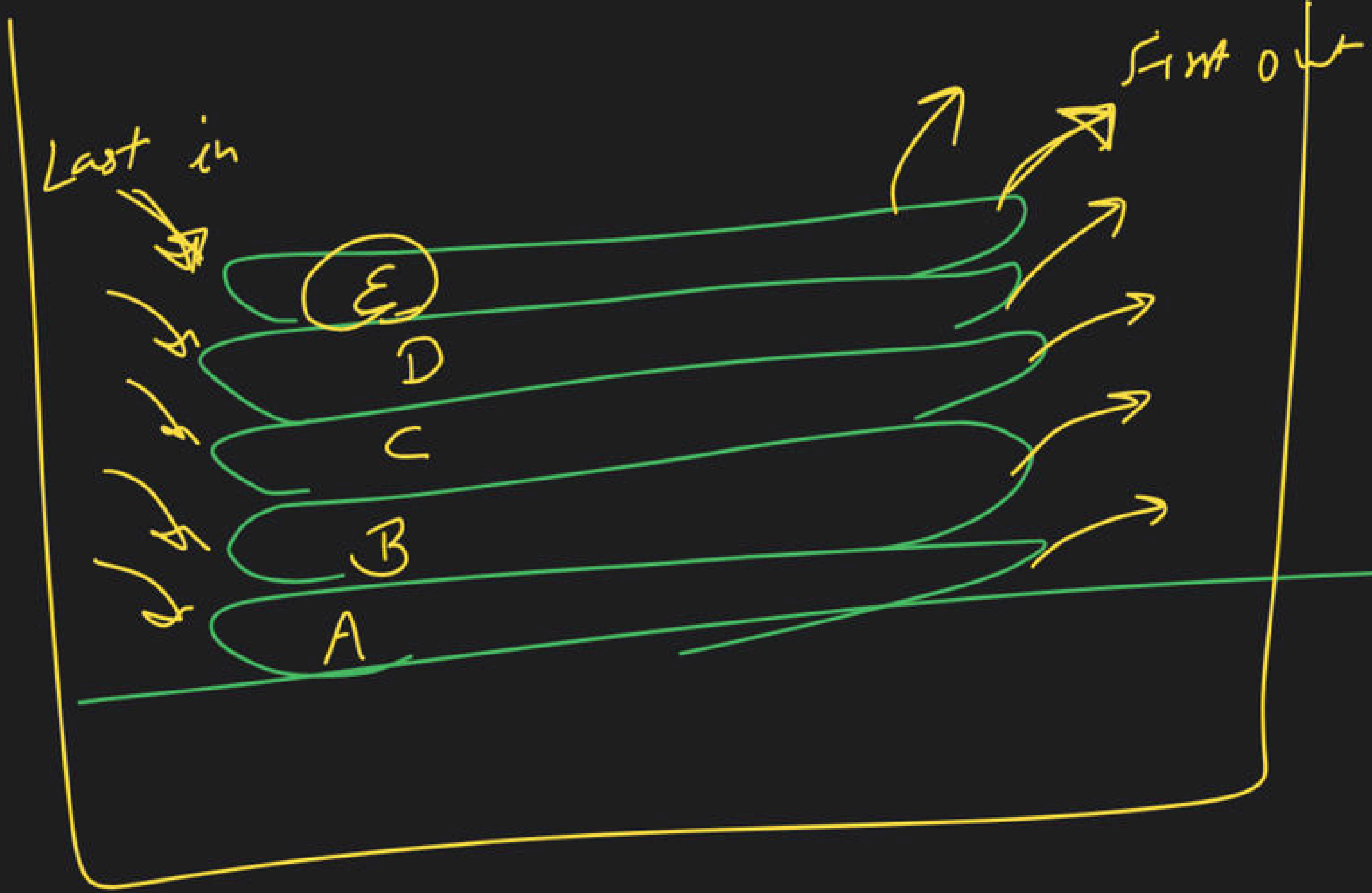
Special class

Love Babbar • July 29, 2024

Stack:

D.S

LIFO ordering



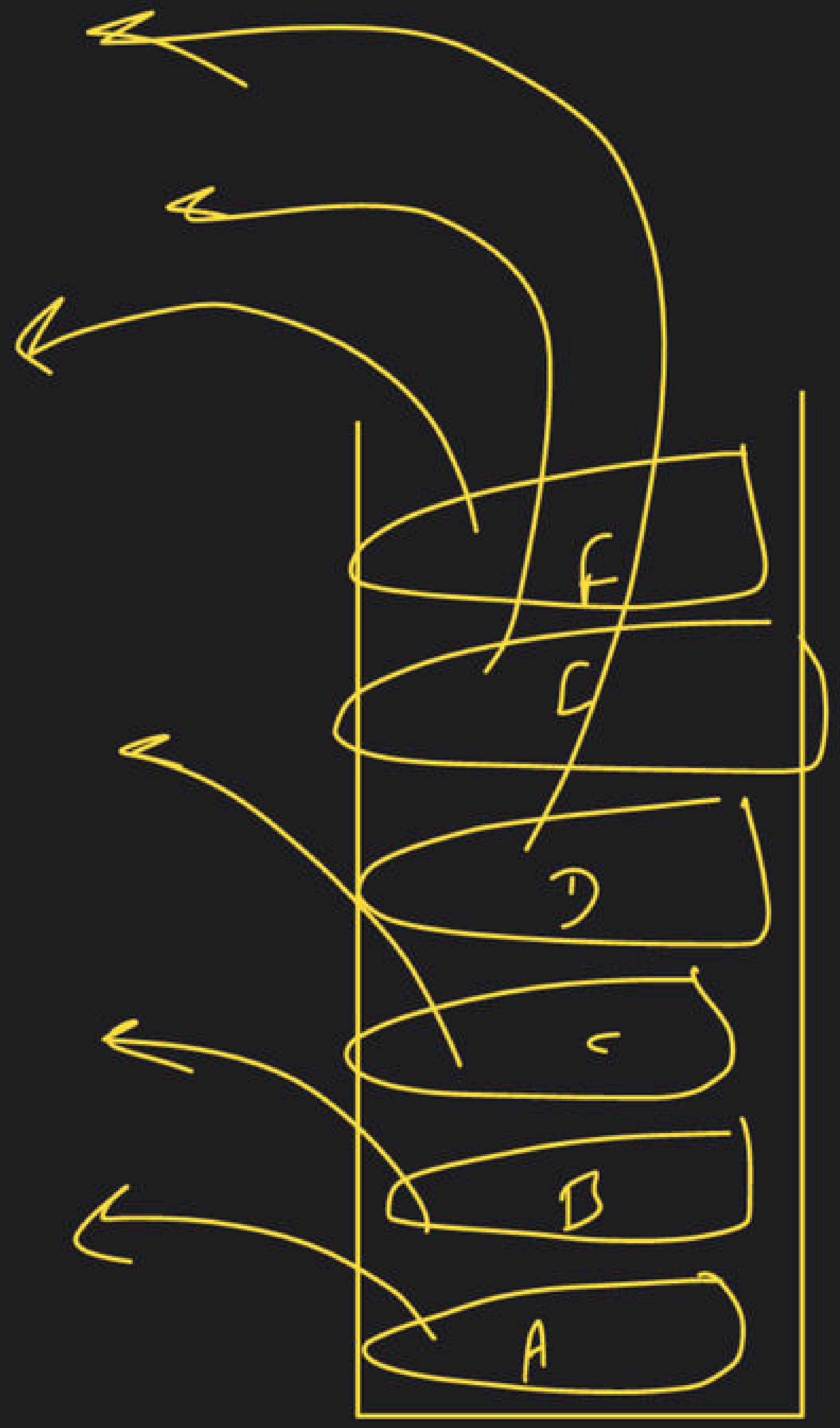
Stack

property

A B C D E

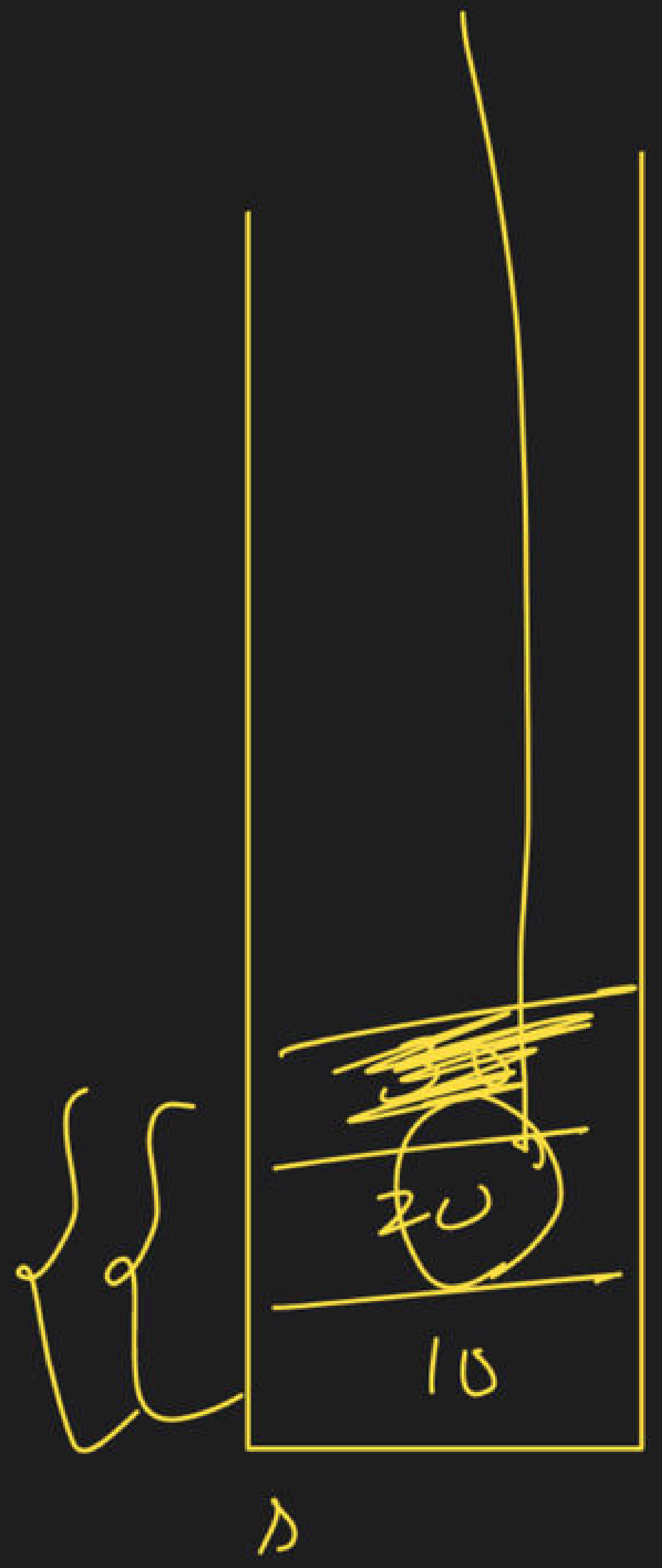
E D C B A

reverse



stl

STL :->



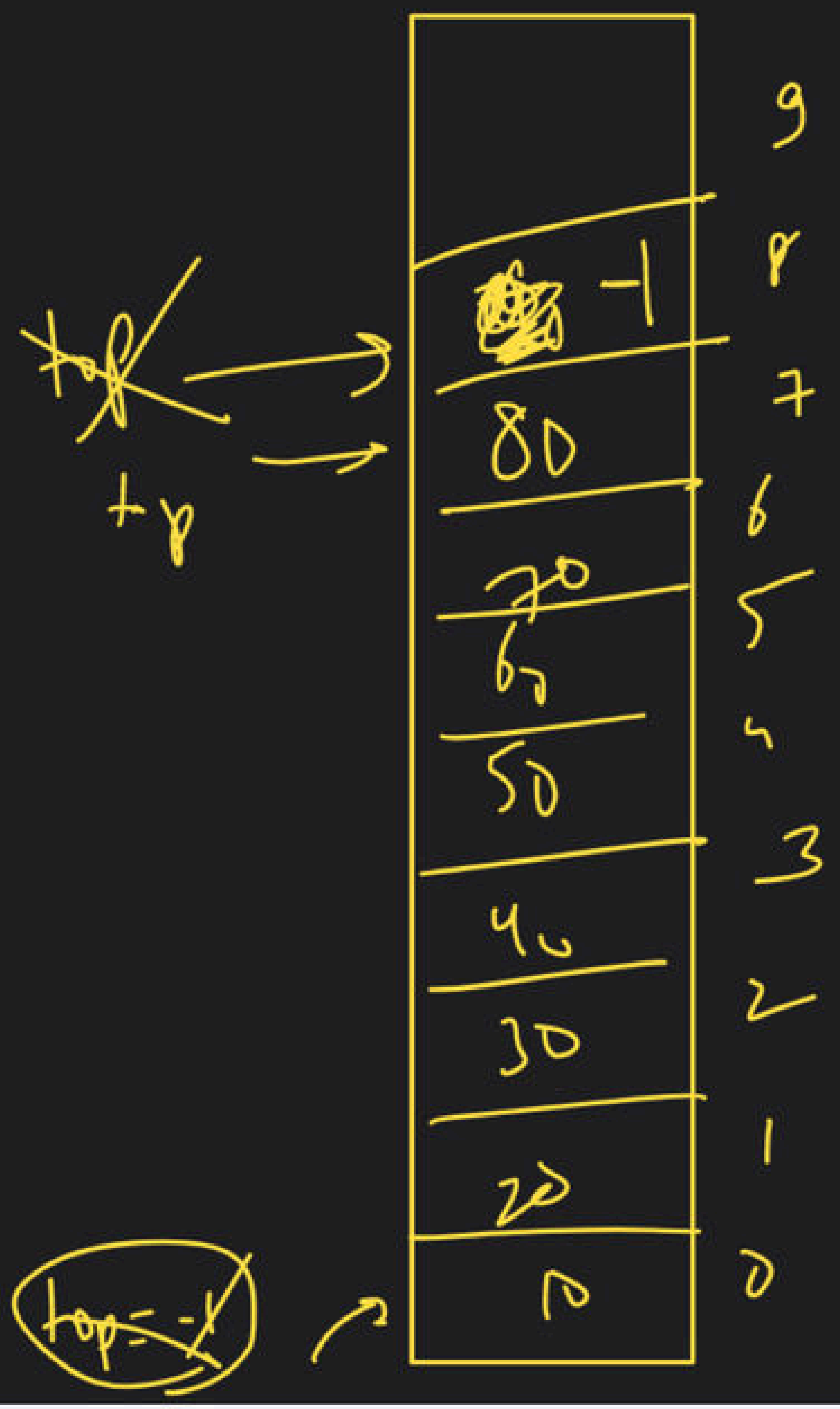
```
s.push(10)
s.push(20)
s.push(30)

s.size()

s.empty() -> F

s.pop()
 s.top() -> 20
```

i/p → stack → max → size → dynamic array

① arr

② size

③ top → index → top element



pop → arr[top] = -1 → ✗

top--

push → top → top+1

→ arr[top] = val

→ top+1

→ arr[top] = value

push → s.push(11)

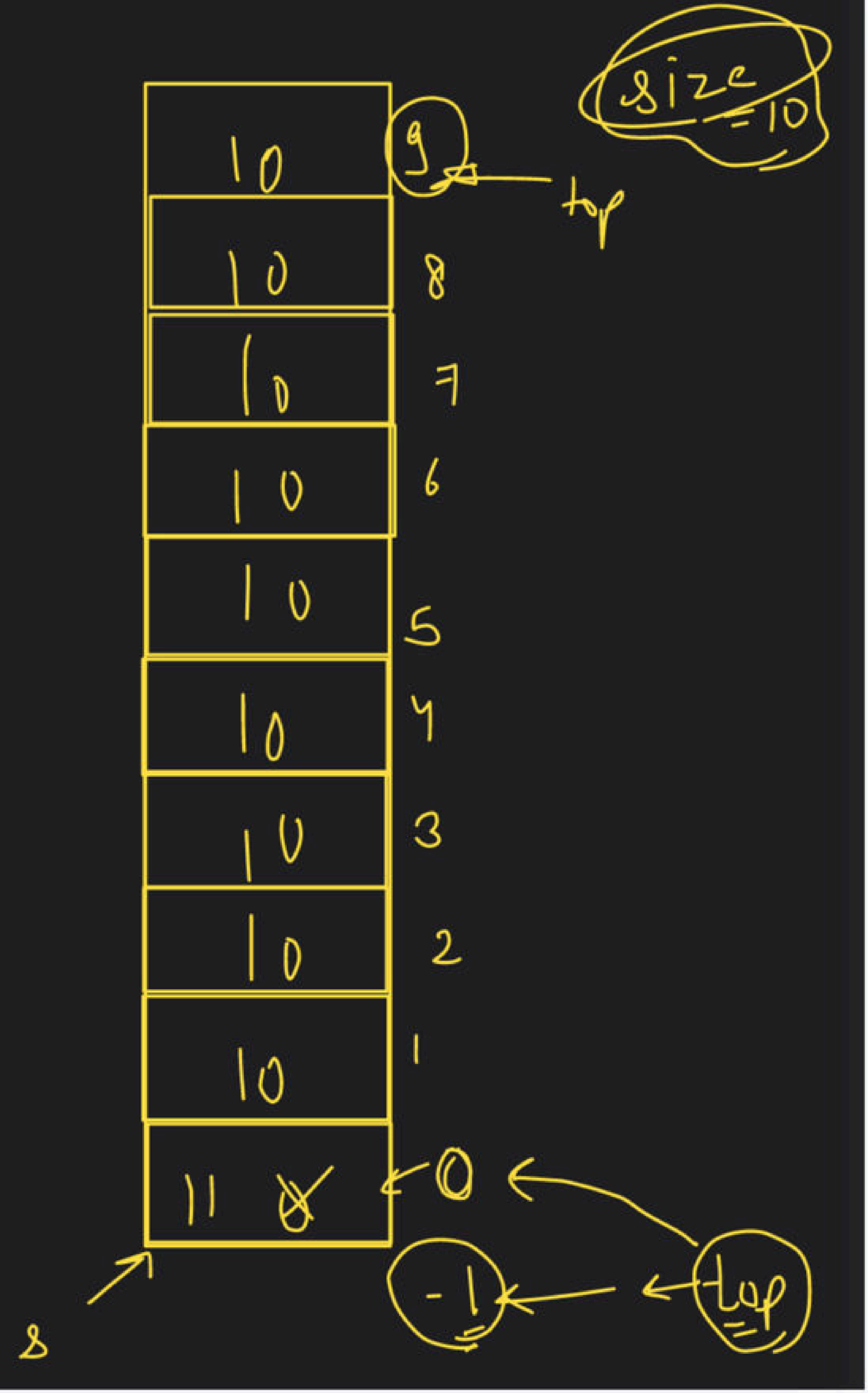
→ top++

→ arr[top] = value

s.push(50)

↳ if (top = = size - 1)

↳ Stack Overflow



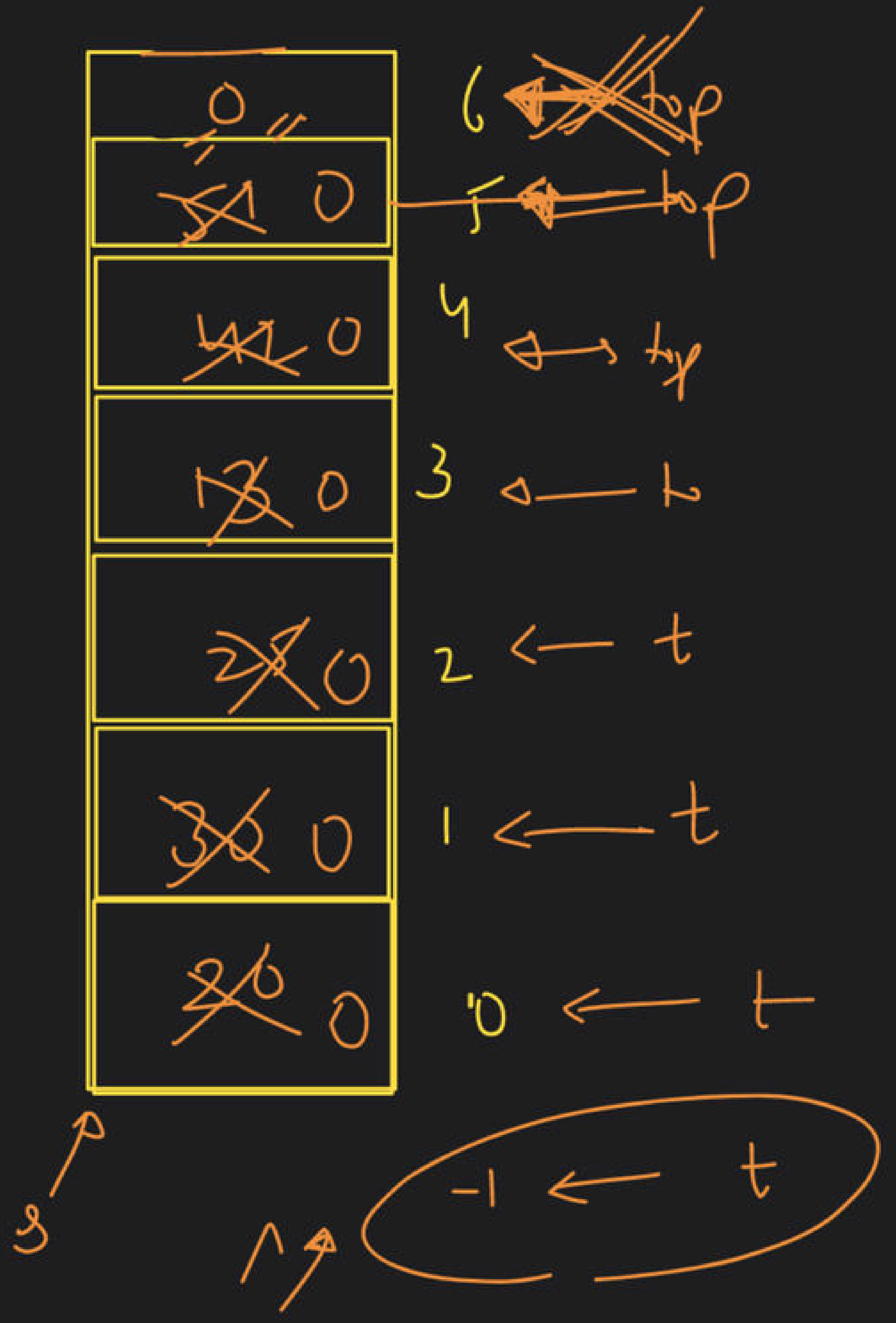
s.pop()

arr[top] = 0;

top --;

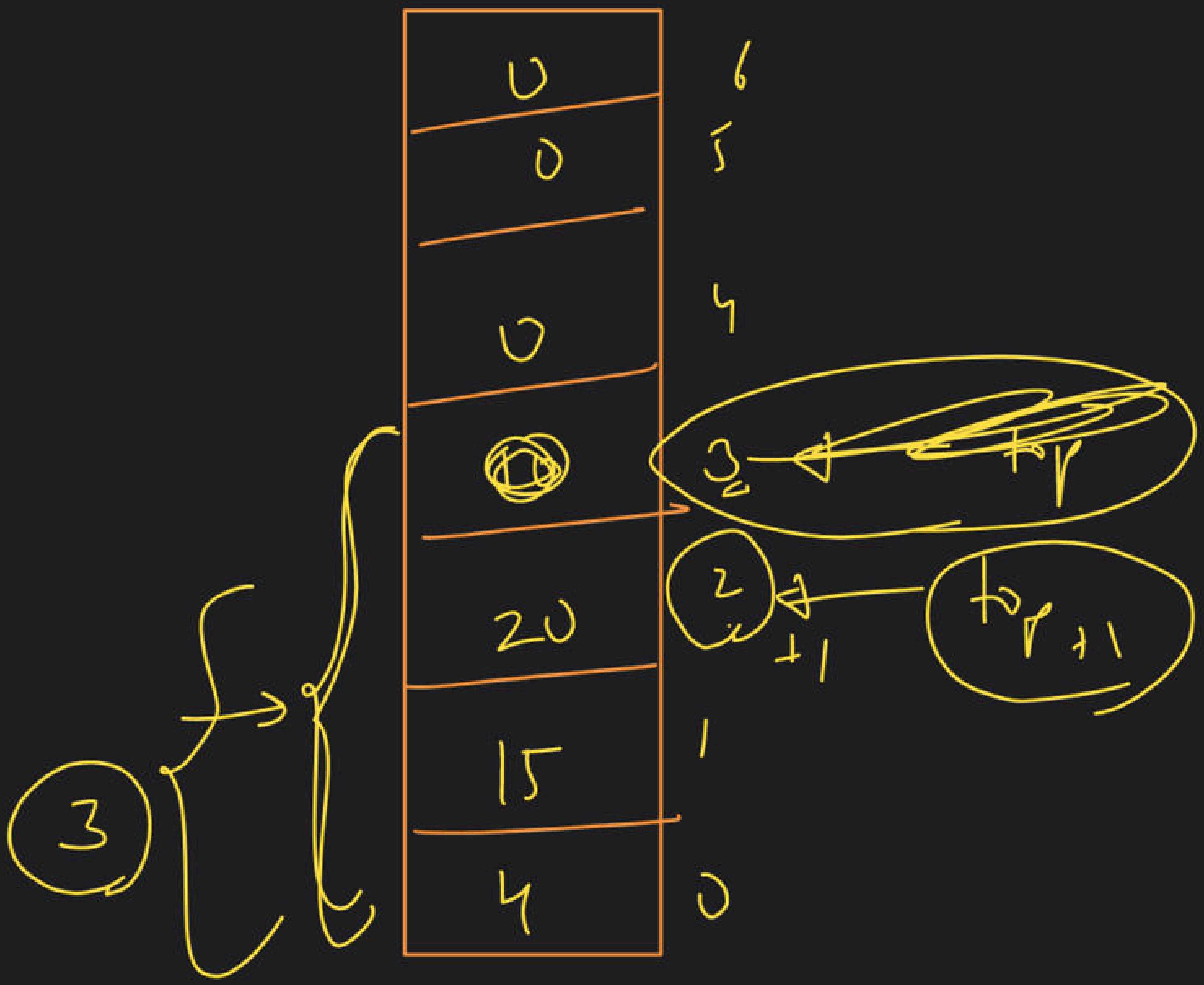
if (top == -1)

S underflow

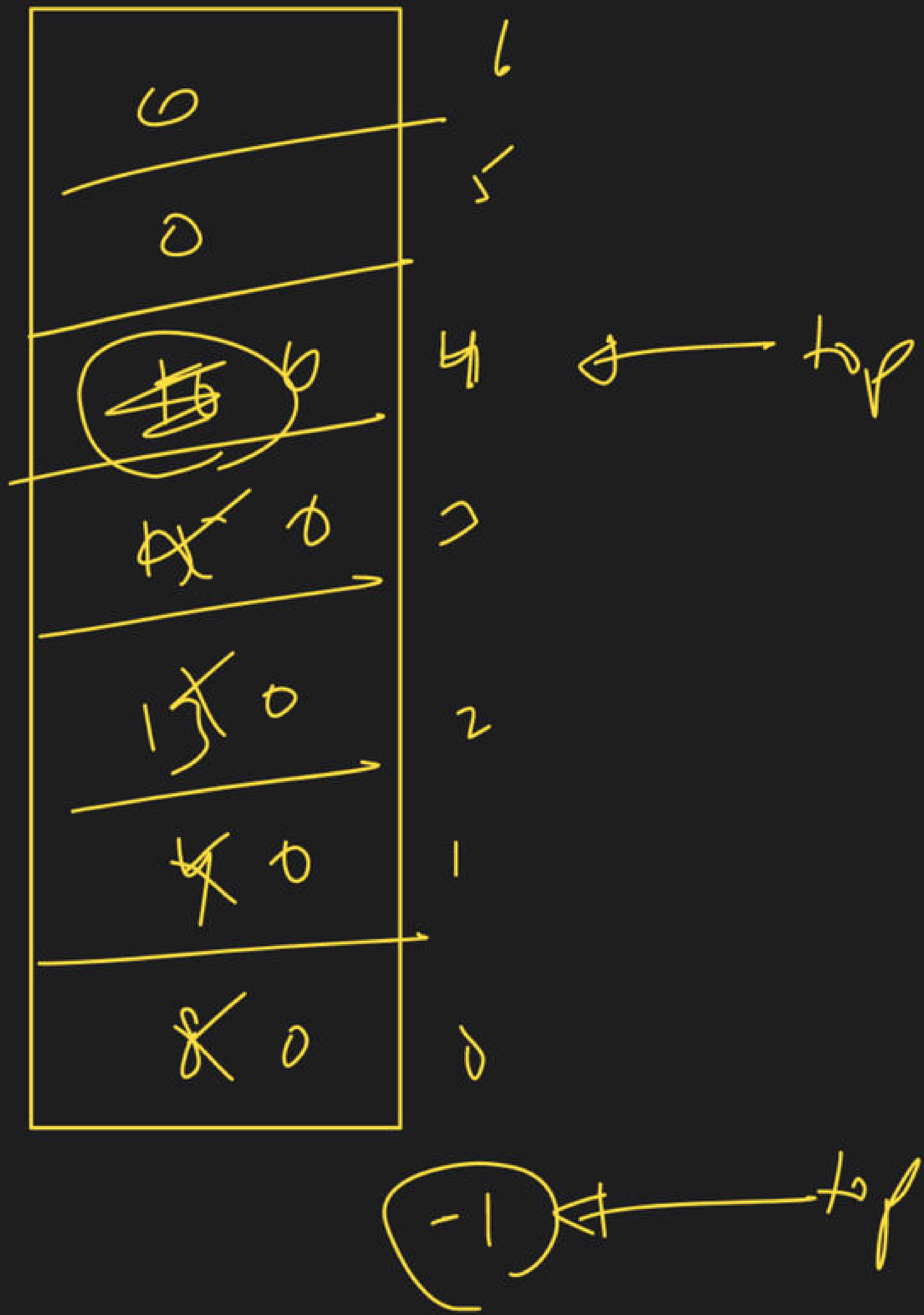


getSize()

return top + 1

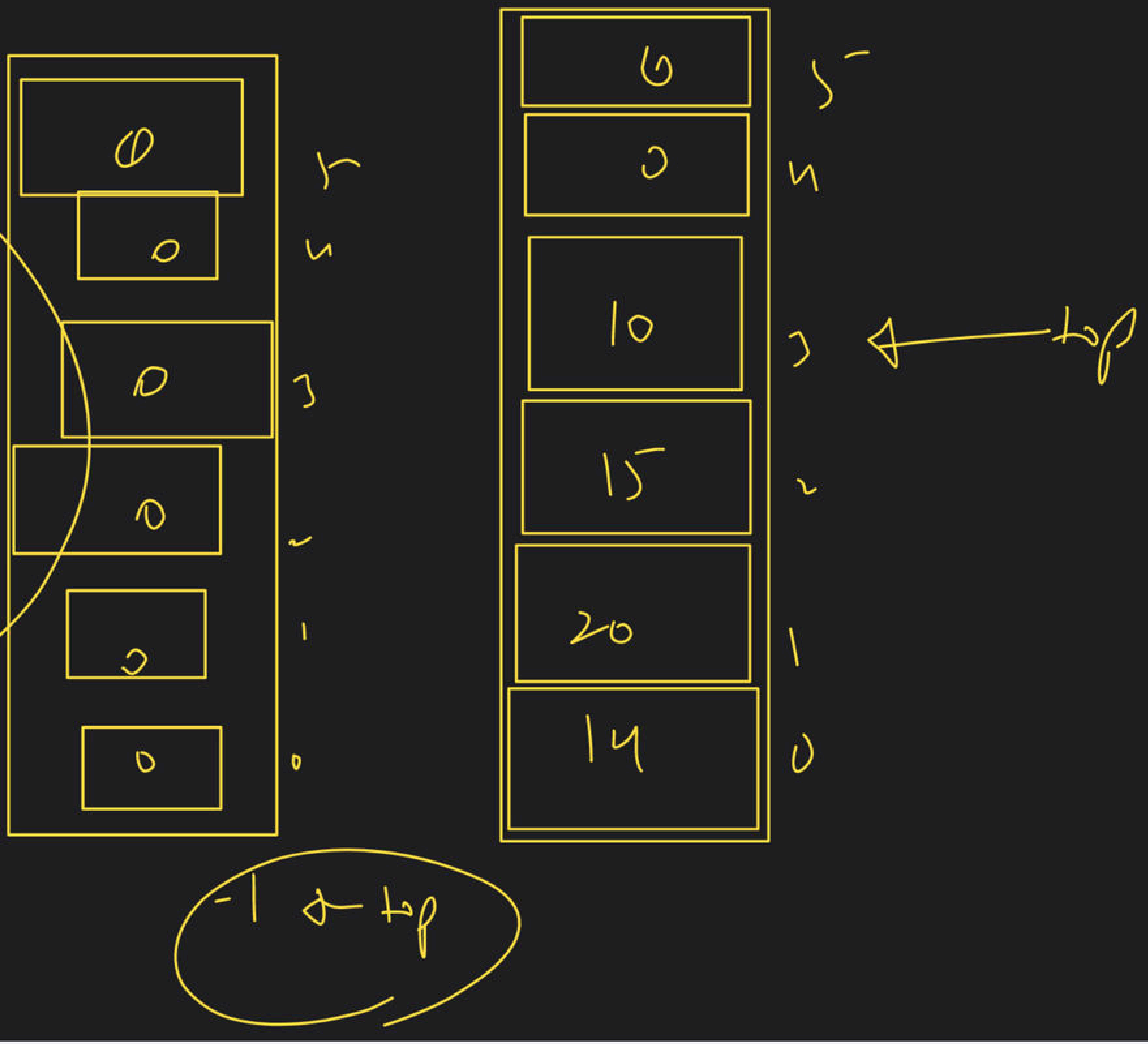


```
getTop()
  ↳ return arr[top];
  ↳ if (top == -1)
      {
      ↳ return -1
```



isEmpty

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



1 stall

2 stacks

→ Implement 2 Stacks in an Array

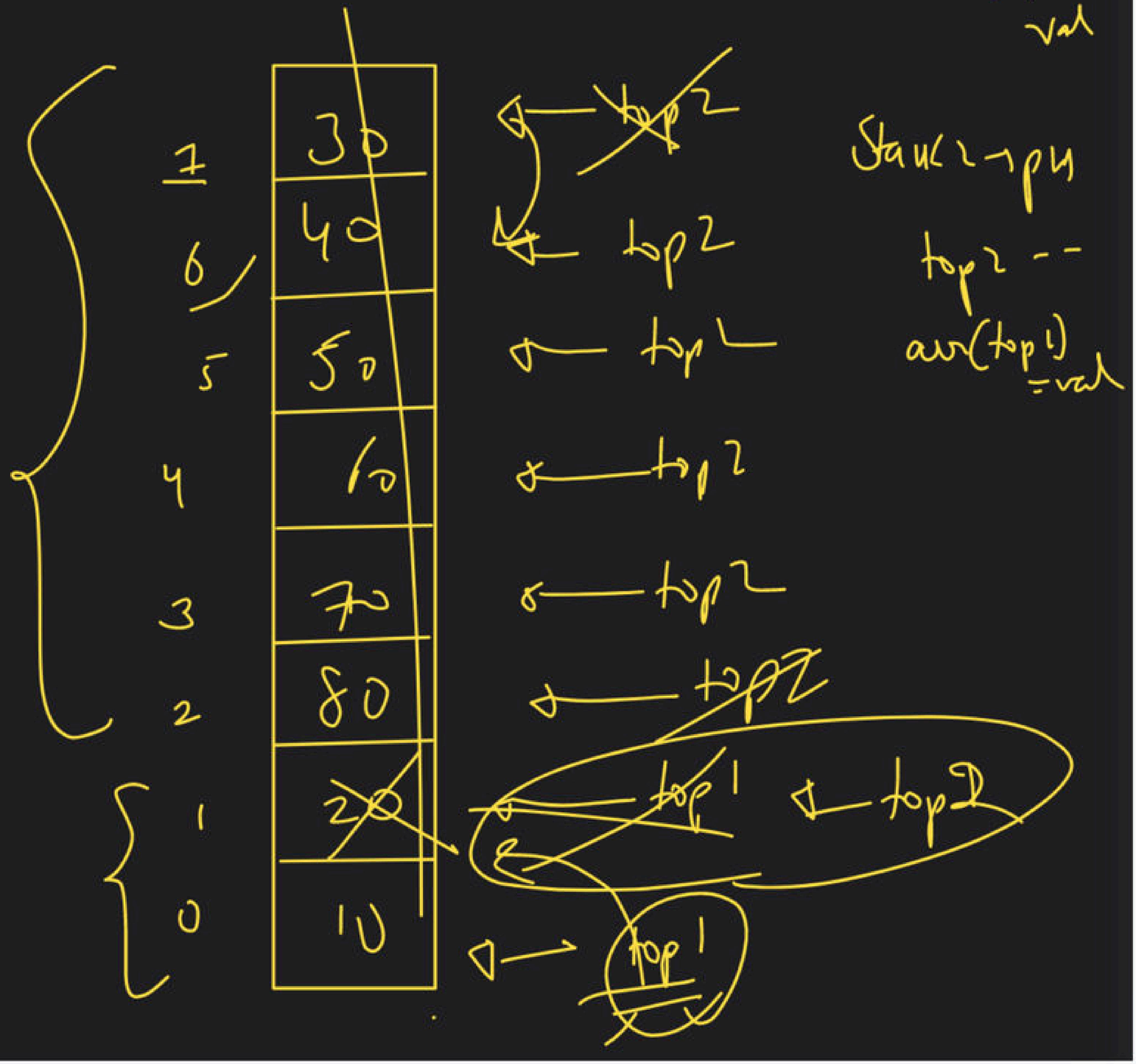
Stack 1 → push
↳ top1++
↳ arr(top1) = val

#1 → array /2 parts

#2 top1, top2

top1 = = top2

↳ S. is Full
↳ Overflow



Stack 2 → push

top2 - -

arr(top1) = val

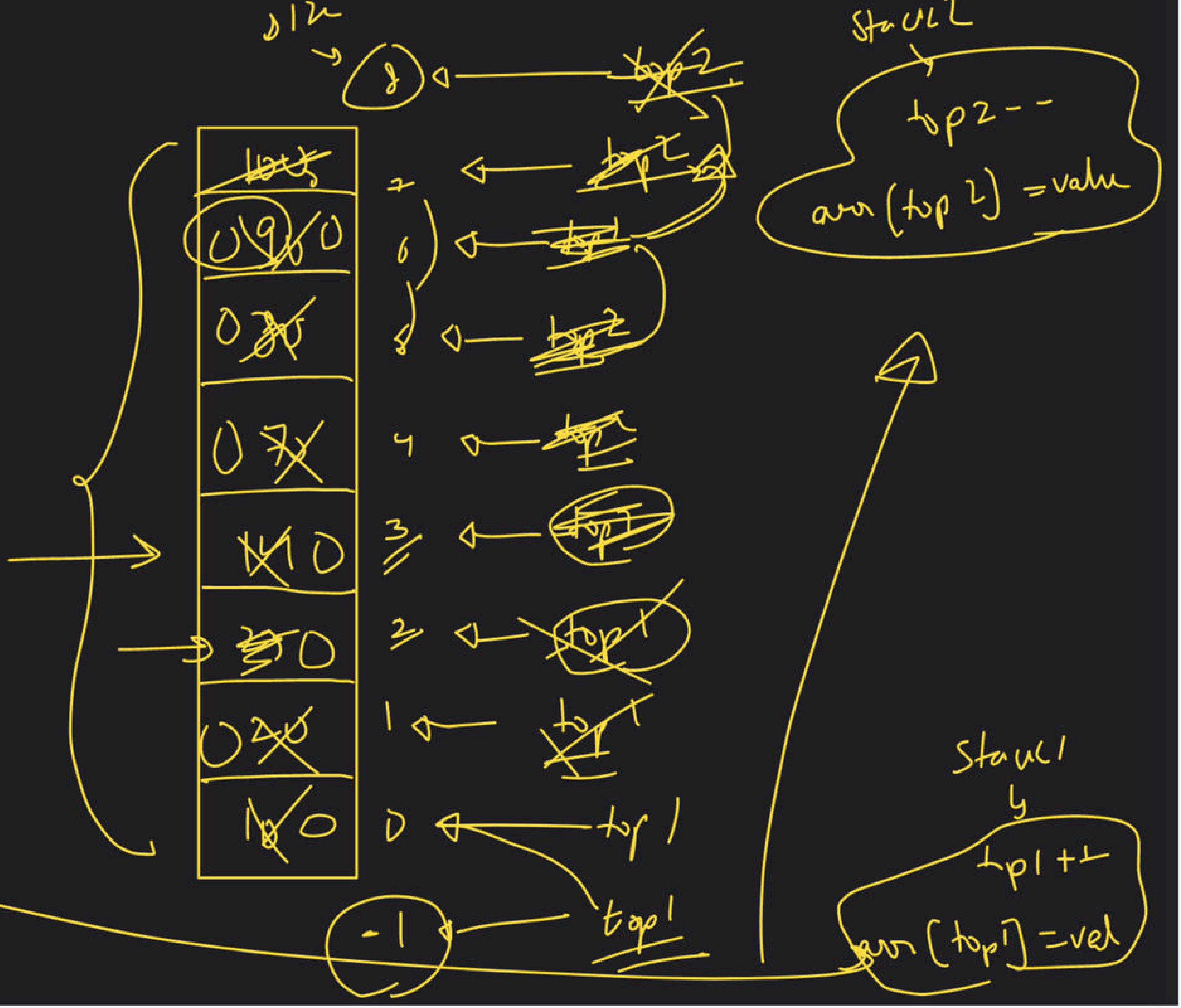
top2 - top1 == 1

→ Stack is full

→ Overflow

else

→ insert

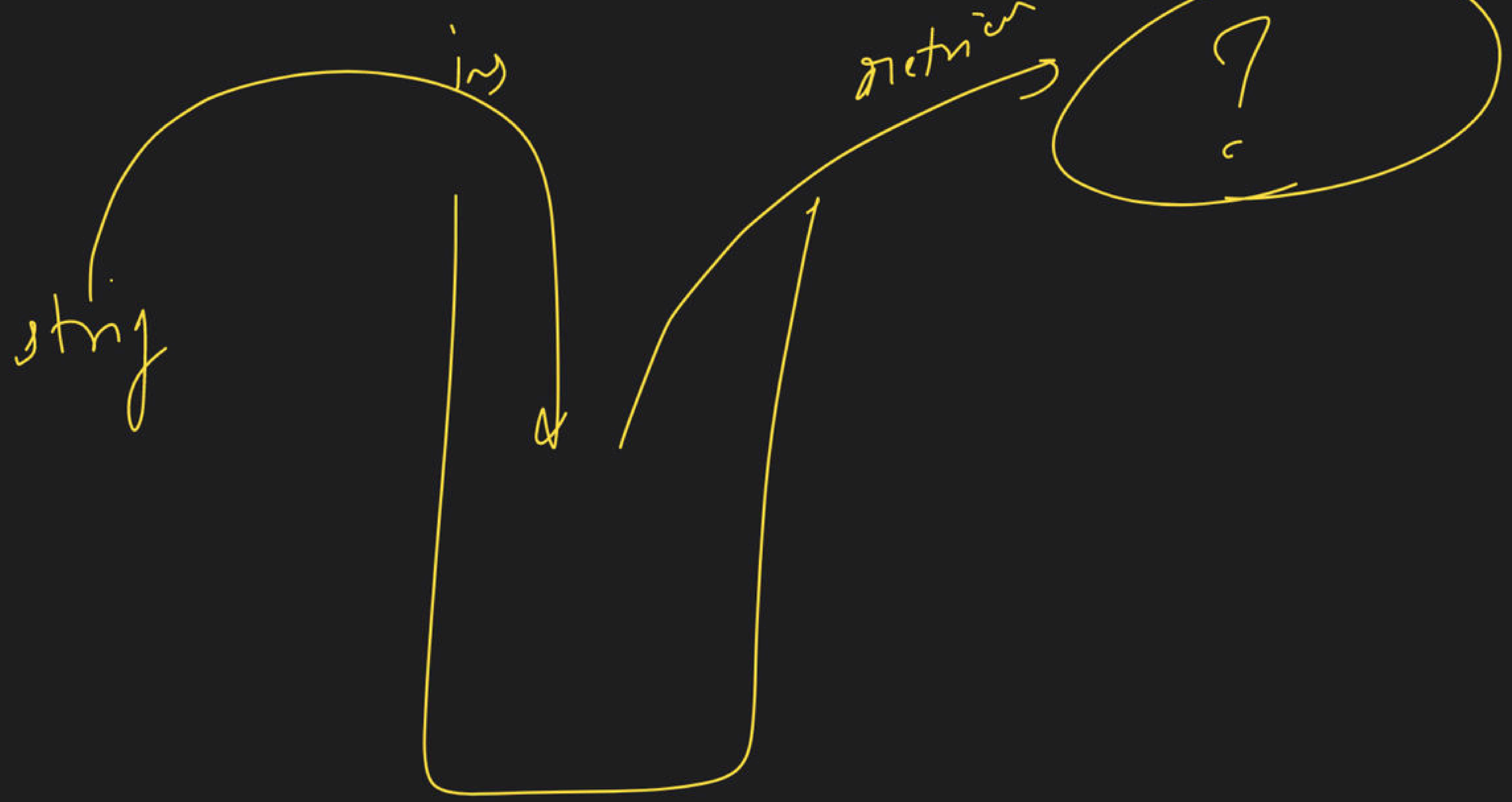


n stacks in an array

2 min Break

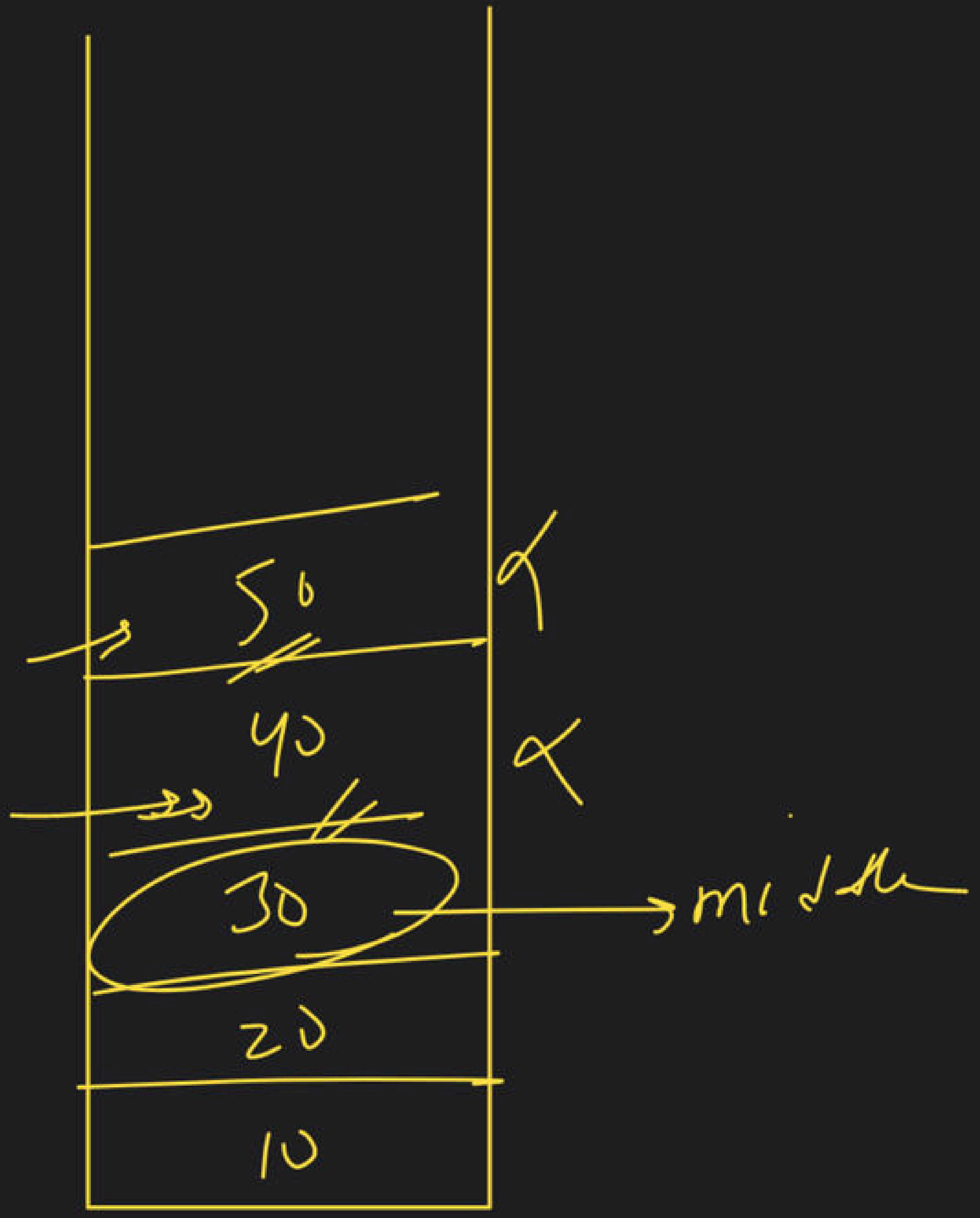
Bye

→ Property → Reverse



→ Pattern

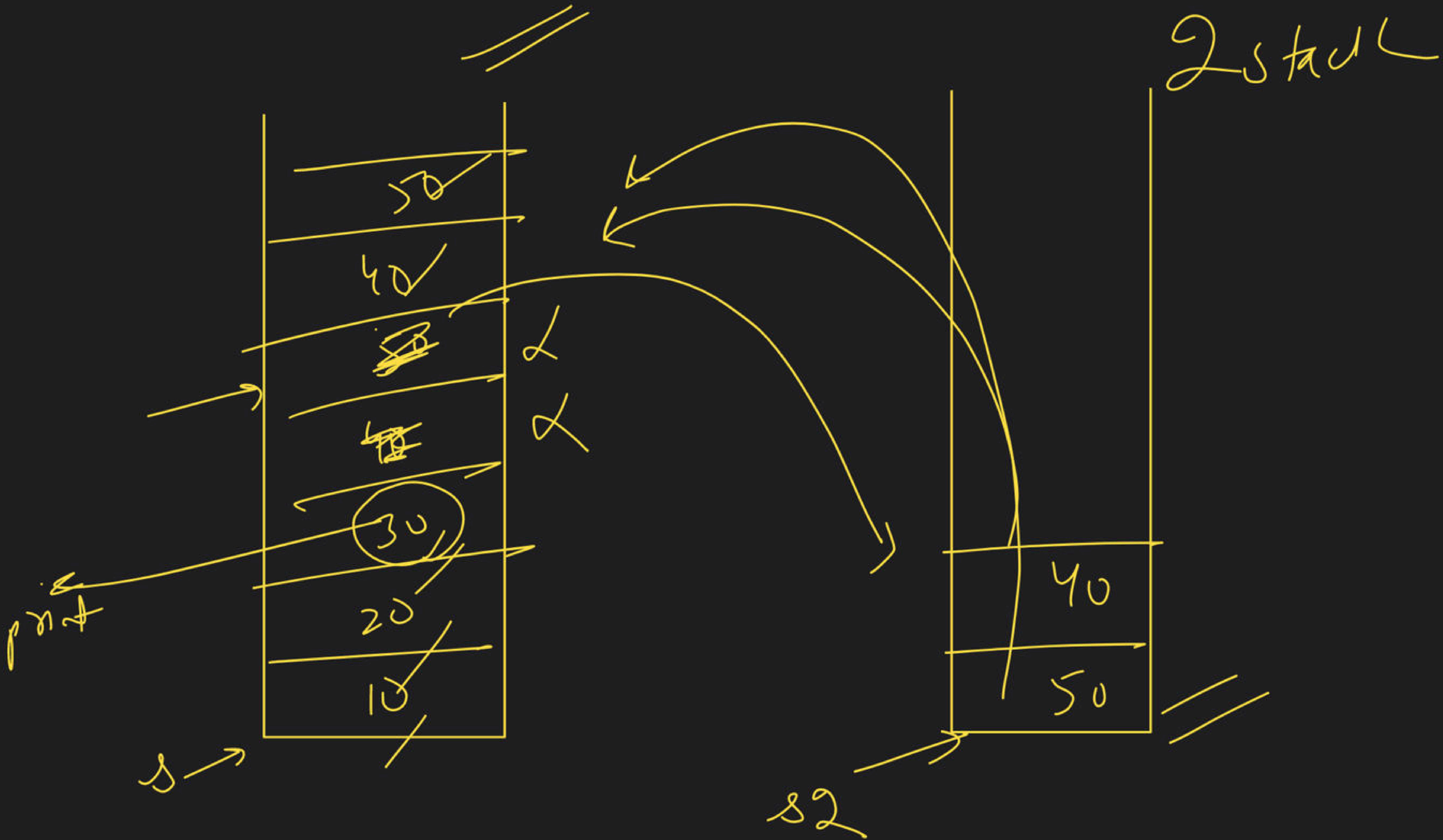
→ PrintMiddle

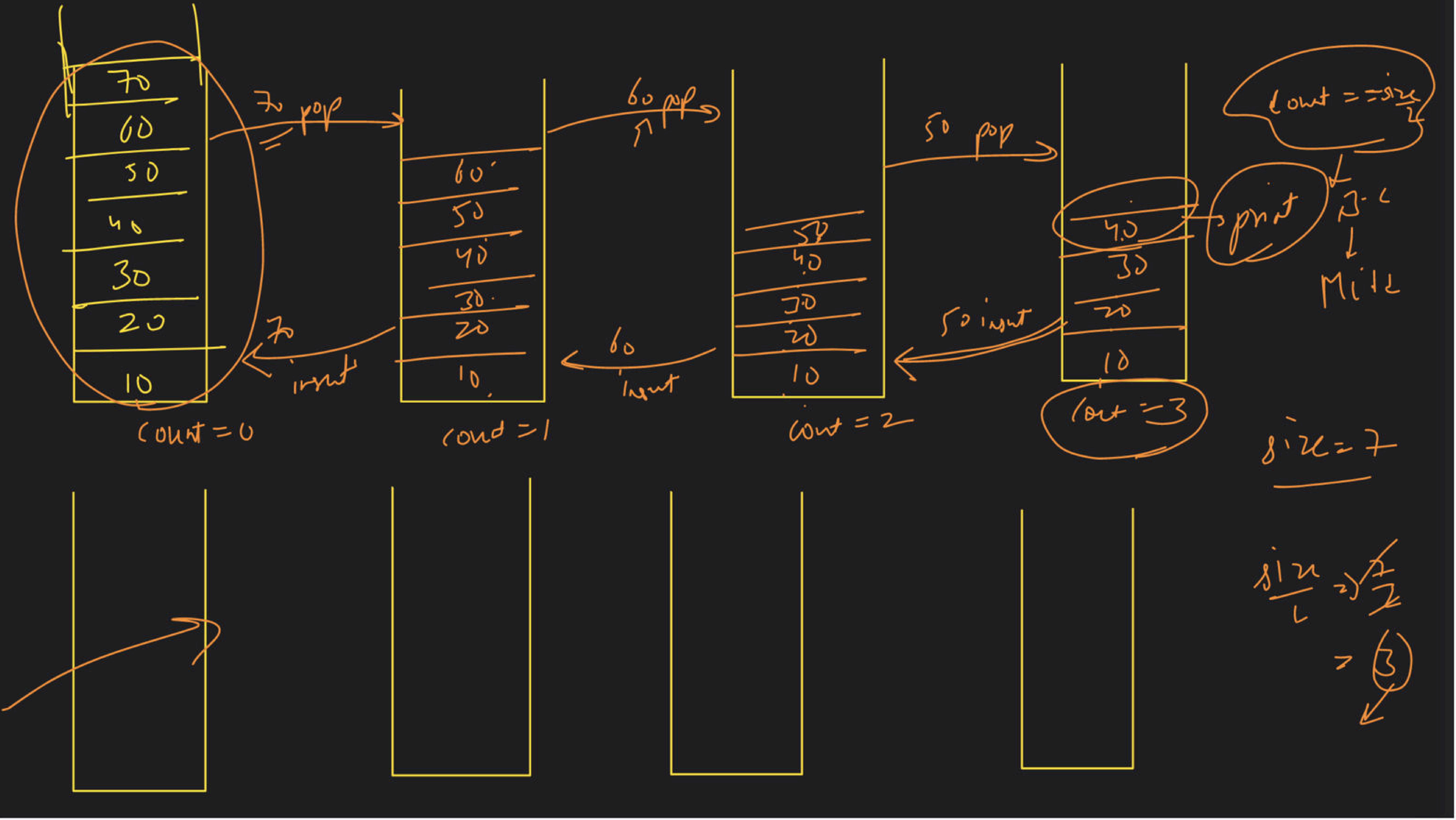


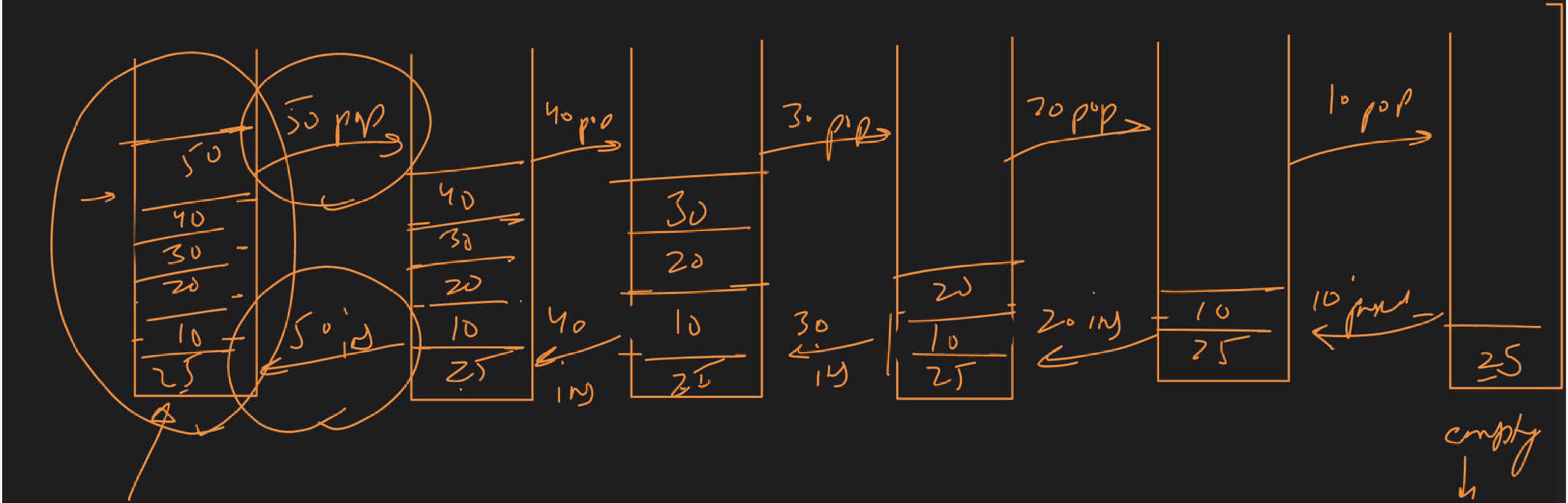
size → 5

$\frac{size}{2} \to \frac{5}{2} \to 2$

α


2 stack
50
40
30
20
10
print
s
40
50
s2


70
60
50
40
30
20
10
count=0
70 pop
70 insert
60
50
40
30
20
10
cond=1
60 pop
60 insert
50
40
30
20
10
count=2
50 pop
50 insert
40
30
20
10
count=3
print
count==size/2
T.C
MLE
size=7
size/2 => 7/2
=> 3



empty

↓

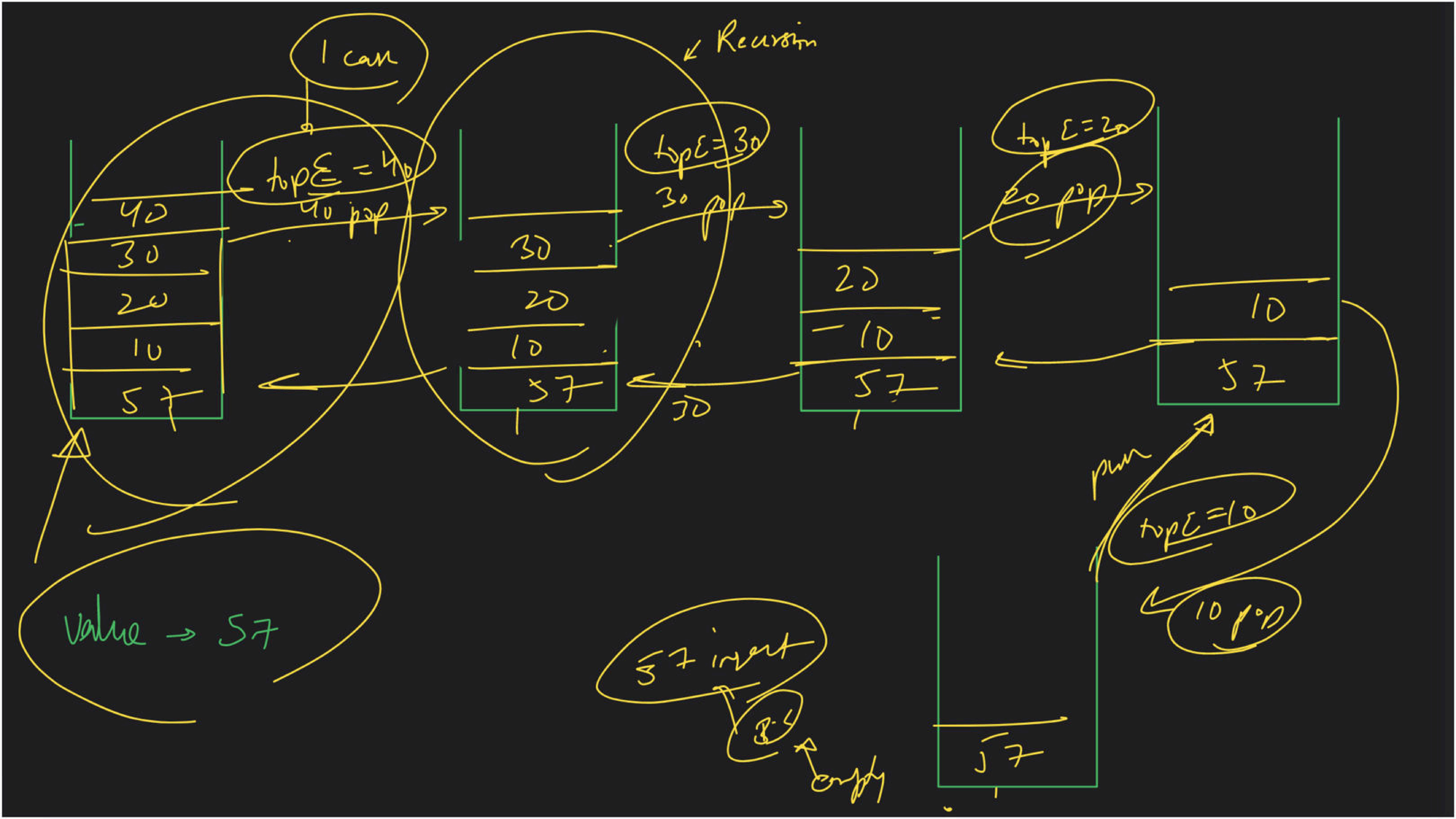
B.C

↓

Insert value

insertAtBottom (value)

value → 25


I can
Recursion
topE = 40
40 pop
40
30
20
10
57
topE=30
30 pop
30
20
10
57
30
topE=20
20 pop
20
10
57
10
57
push
topE=10
10 pop
value → 57
57 insert
B.C
empty
57