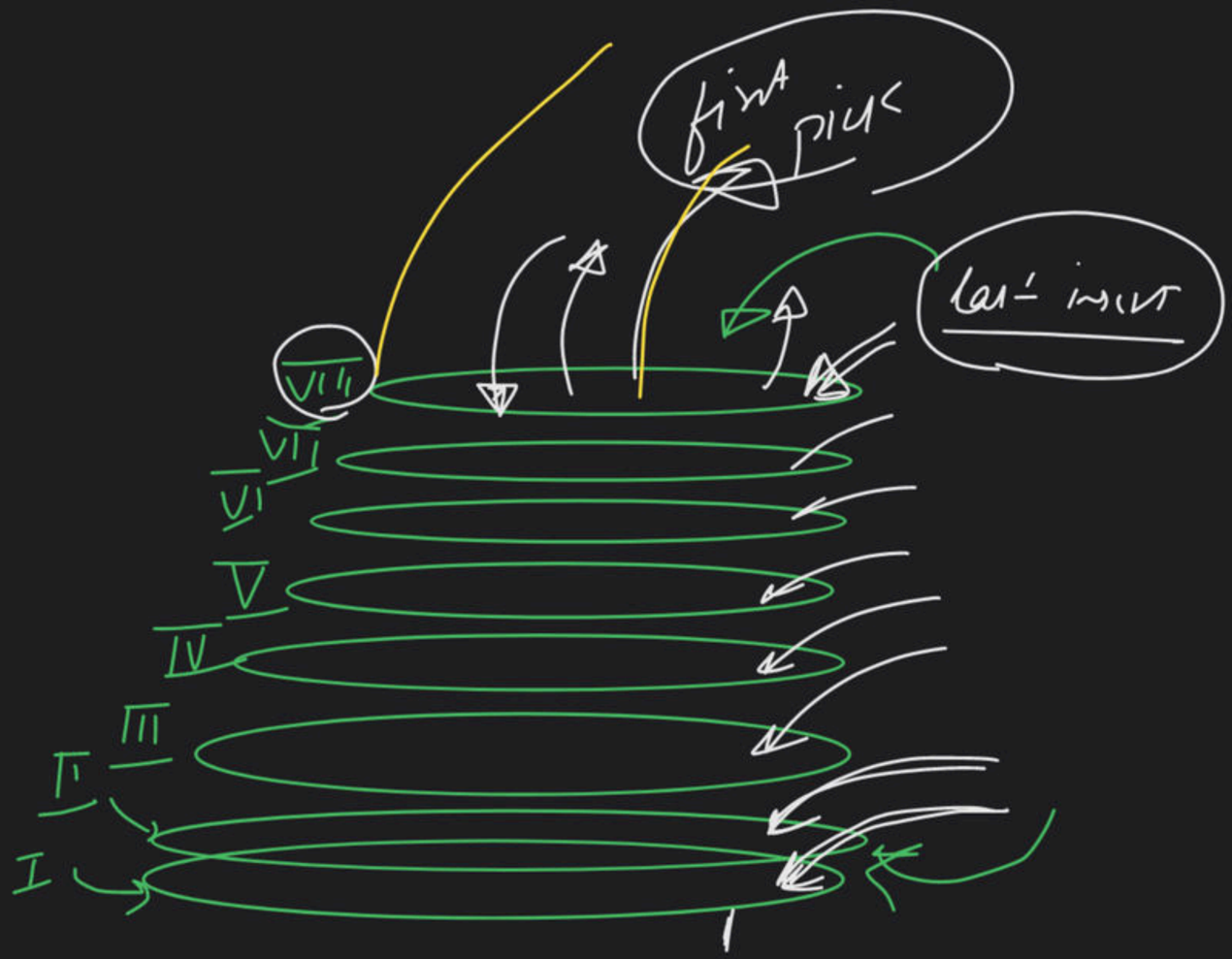
# Stack Class - 1

Special class

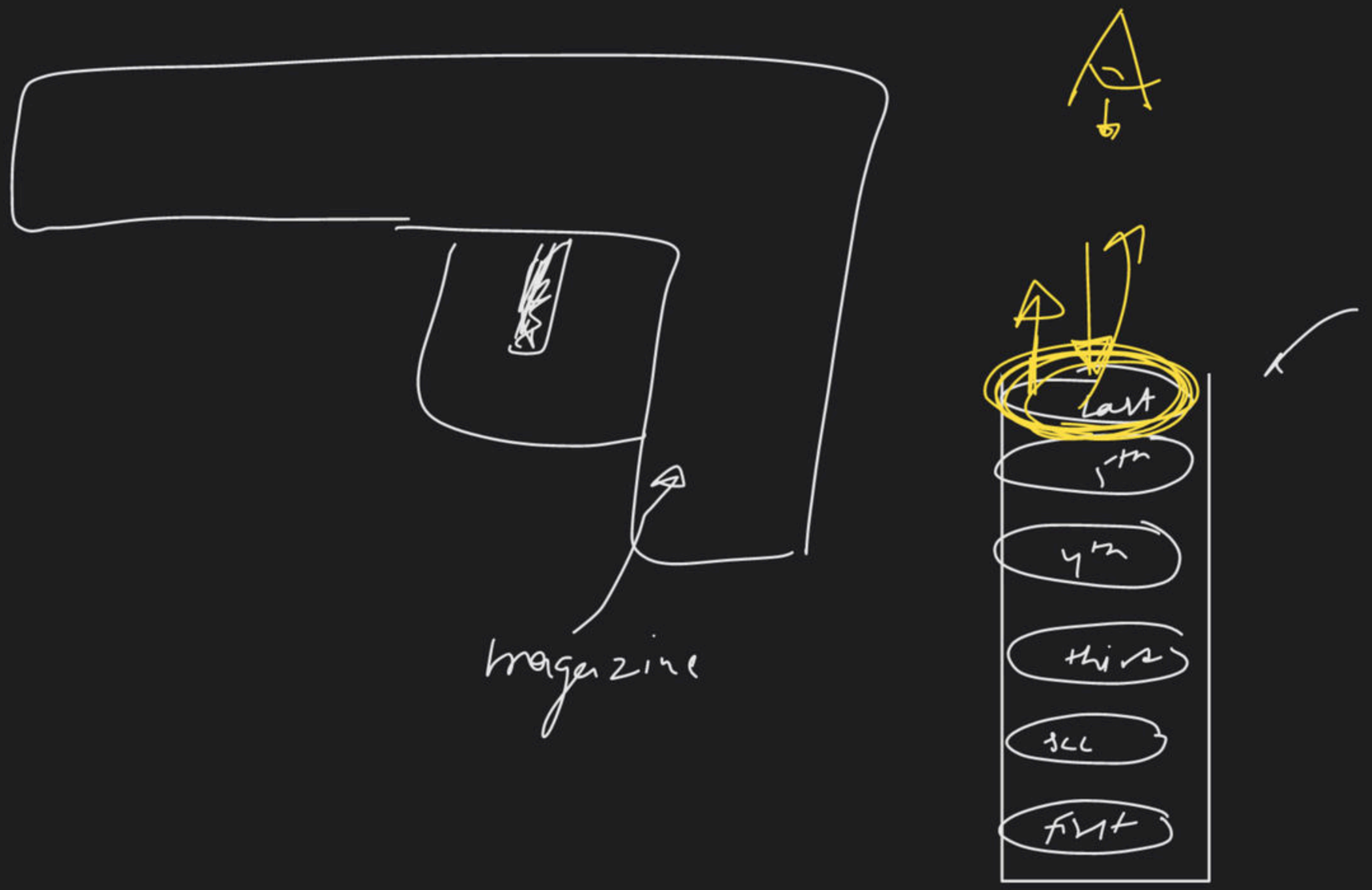
**Love Babbar** • Apr 23, 2023

Stacks → Data Structure → items / data store

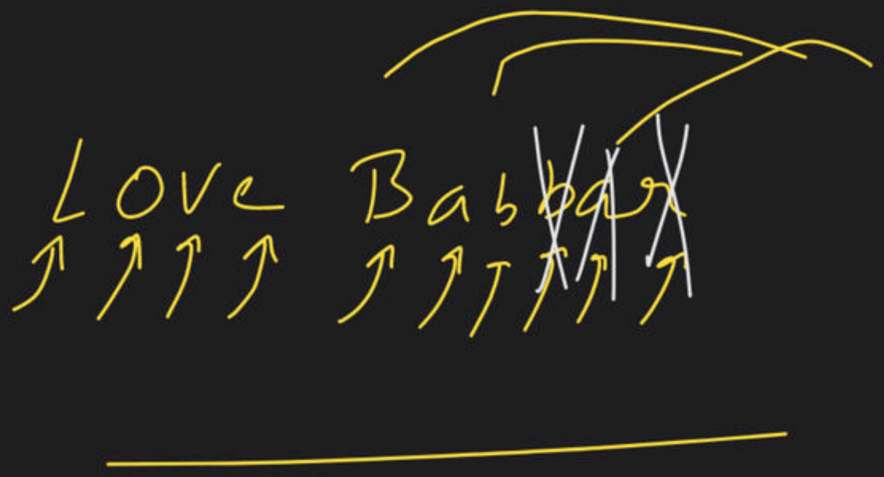



LIFO order

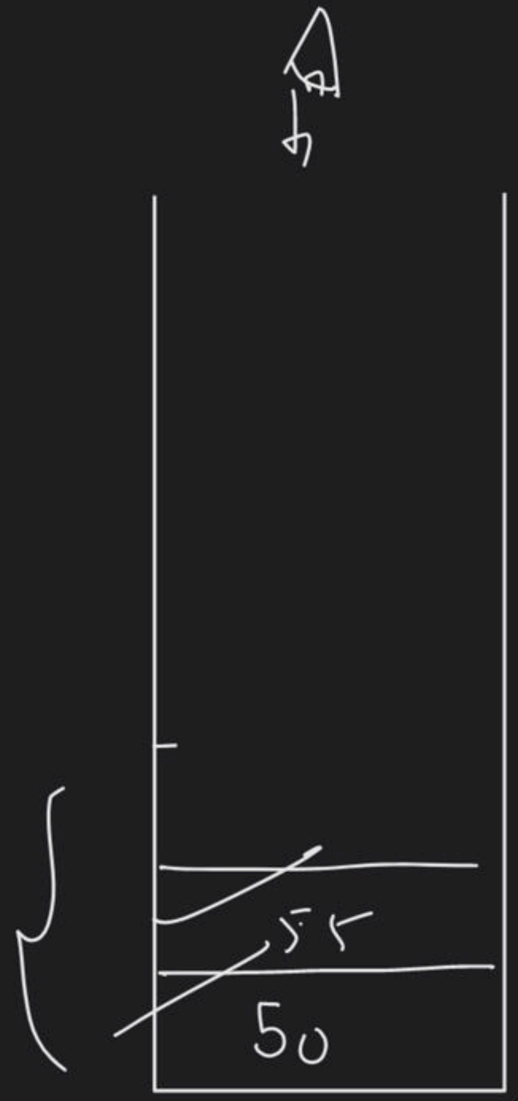
Last In First Out


magazine
last
first
second
third
fourth
fifth

Ctrl + Z



Stack → D.S → LIFO




stack s;

s.size()

no of elements

Peek

s.top()

55

insertion

s.push(50);

s.push(55);

s.push(420);

isEmpty

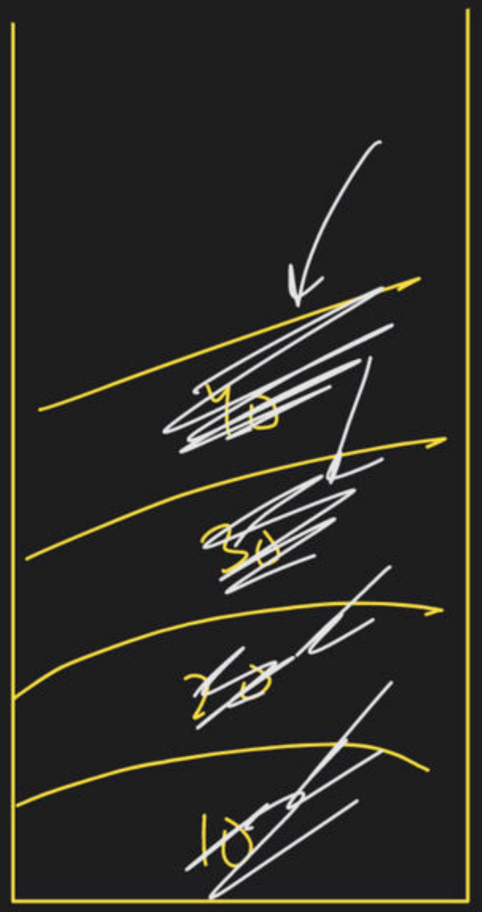
s.empty()

T F

removal

s.pop()

10 20 30 40

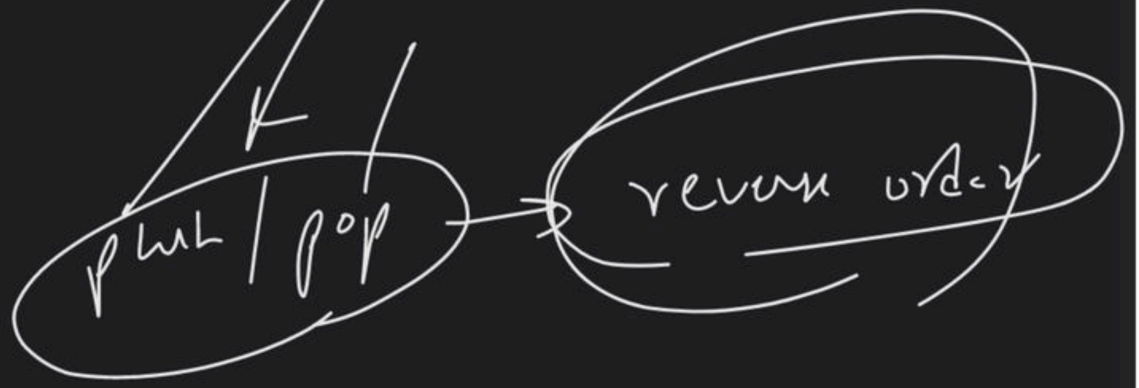


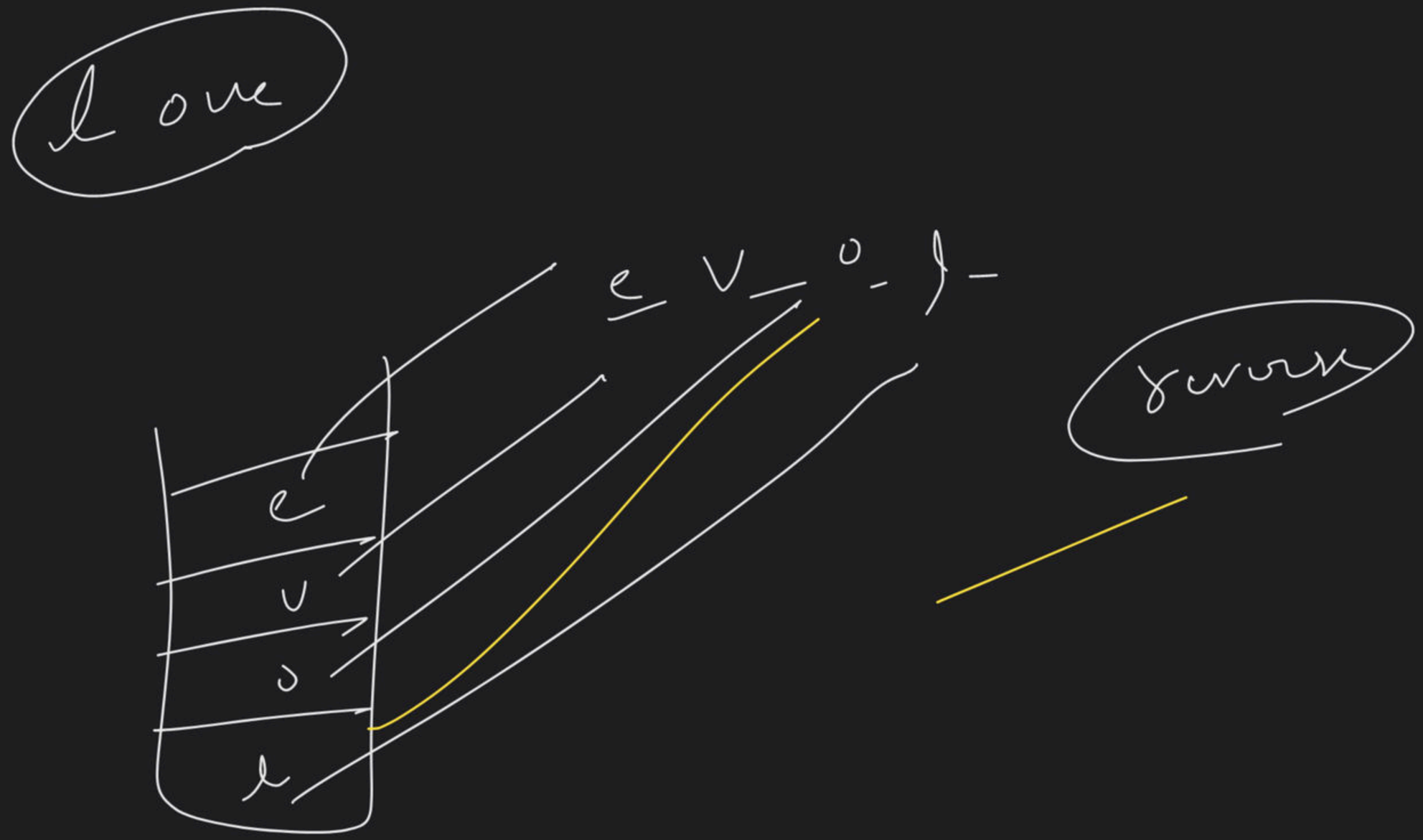

40 30 20 10

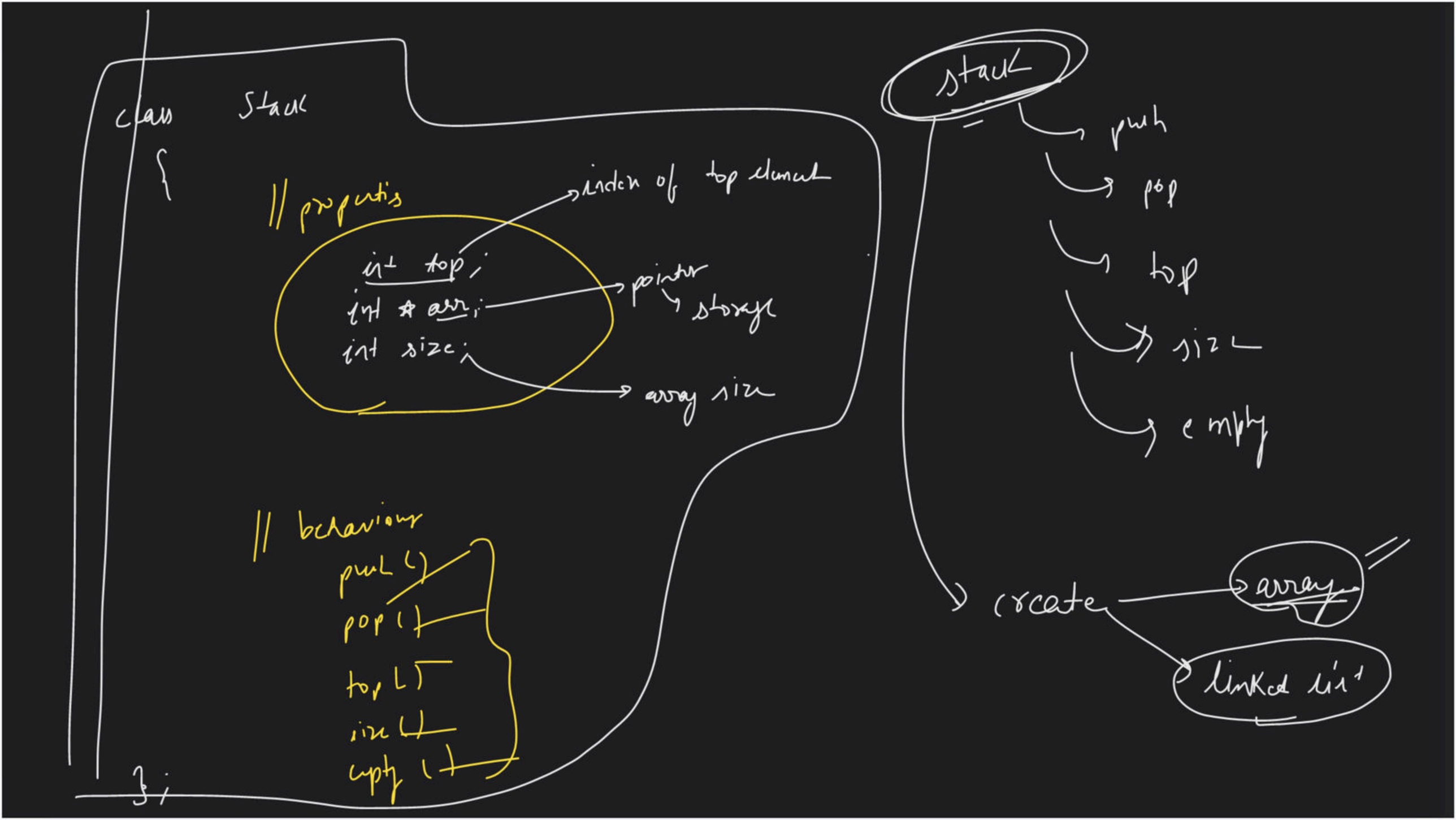
Removal

stack

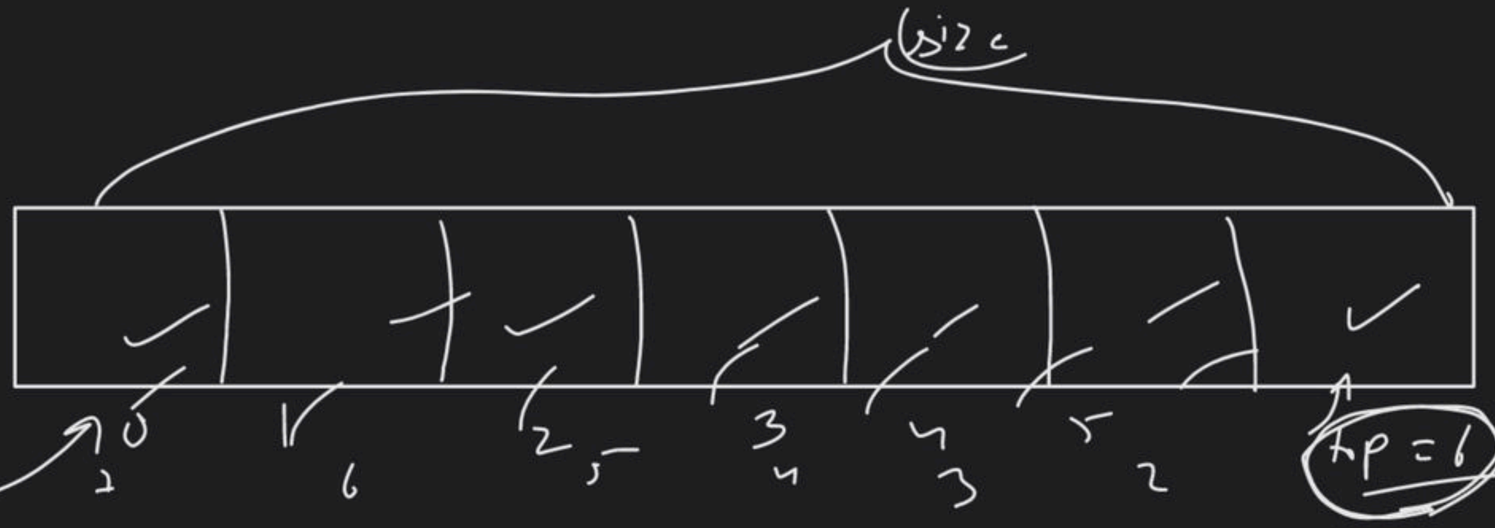
↳ nature




e
v
o
l


class Stack
{
// properties
int top;
int * arr;
int size;
index of top element
pointer
storage
array size
// behaviours
push ()
pop ()
top ()
size ()
empty ()
};
stack
push
pop
top
size
empty
create
array
linked list

Push




1 min
Break

size - top > 1

T

F

size - top
= 7 - 6 = 1

7 - 6 = 1

size = 7

size - top
= 7 - (-1) = 8
=

Push

Space available

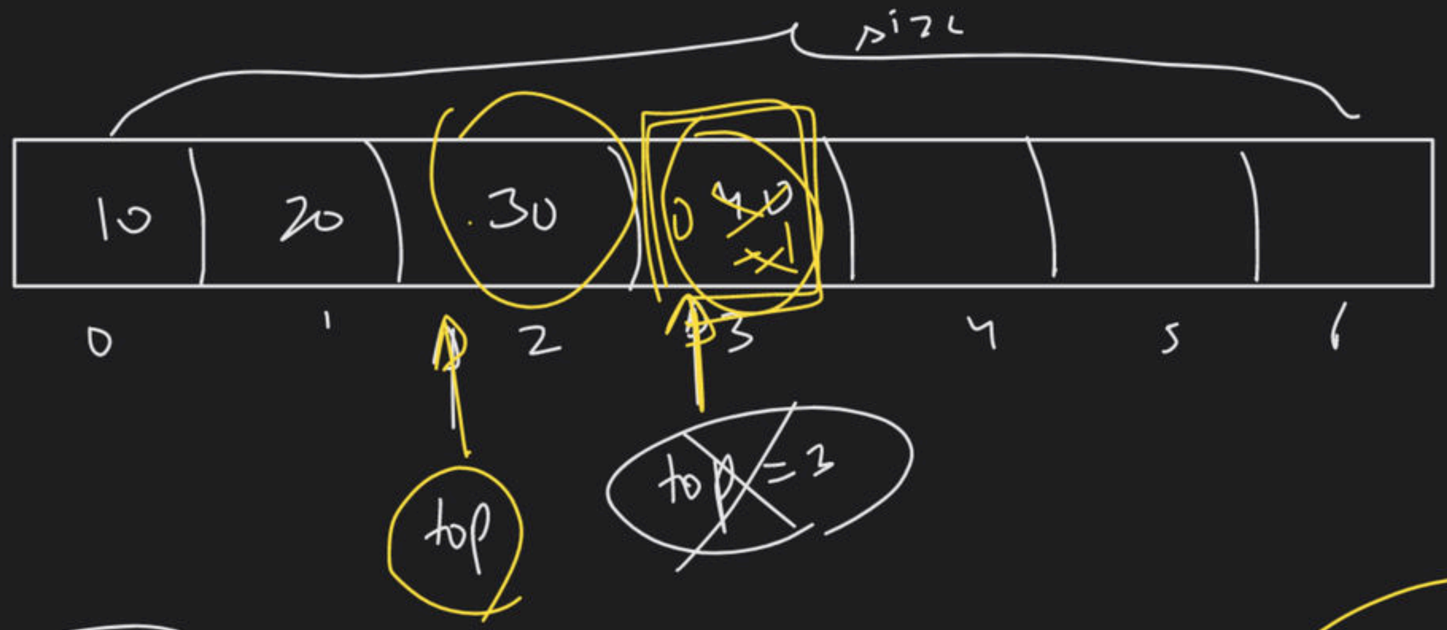
space not available

Overflow

insert

top ++

arr[top] = data




top = -1 → stack empty

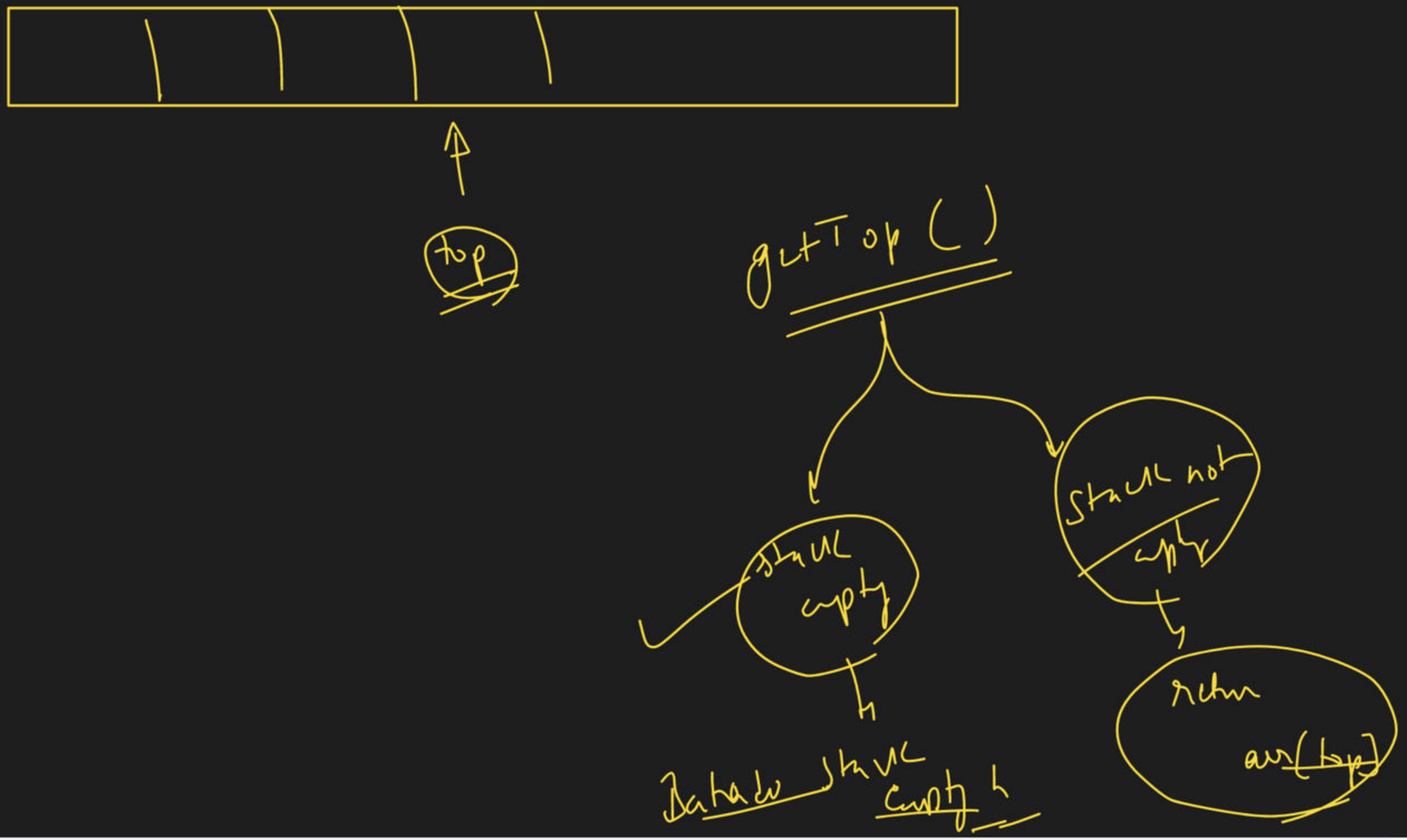
remove

stack empty

stack not empty

Underflow

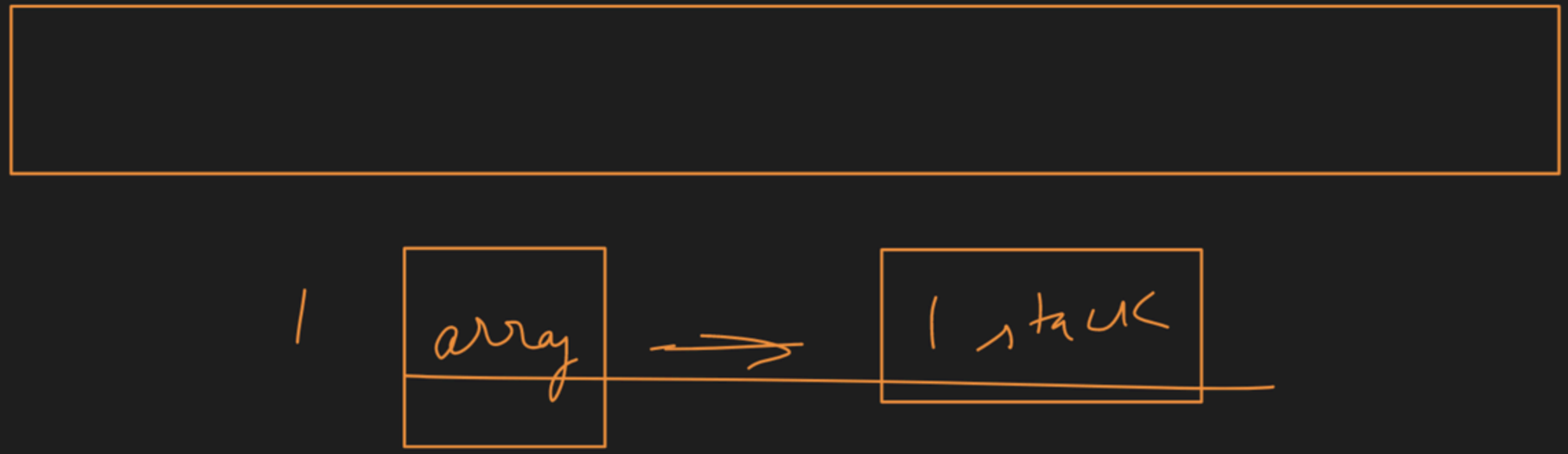
top--

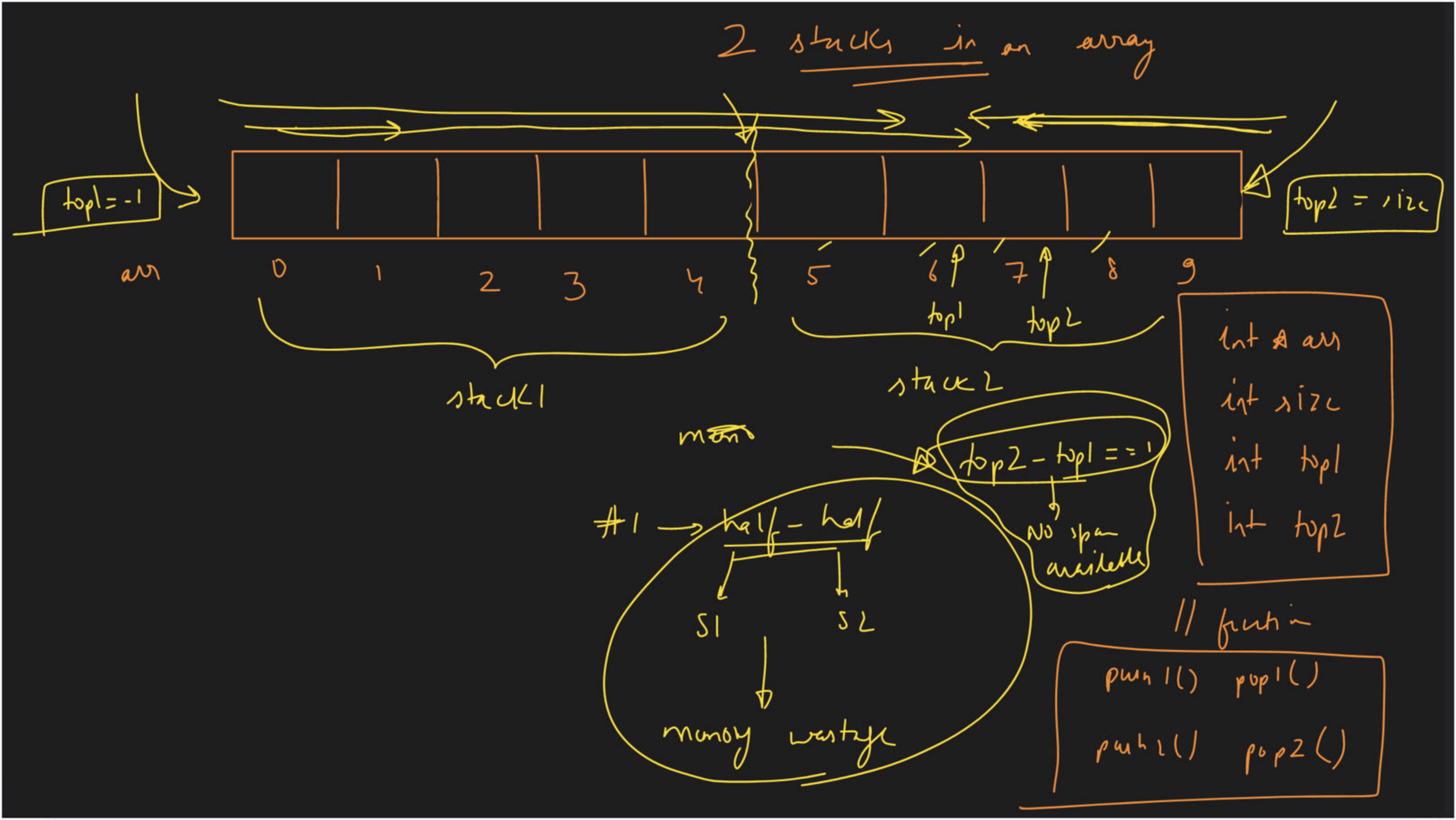

top
getTop()
stack empty
Return -1
stack not empty
return arr[top]

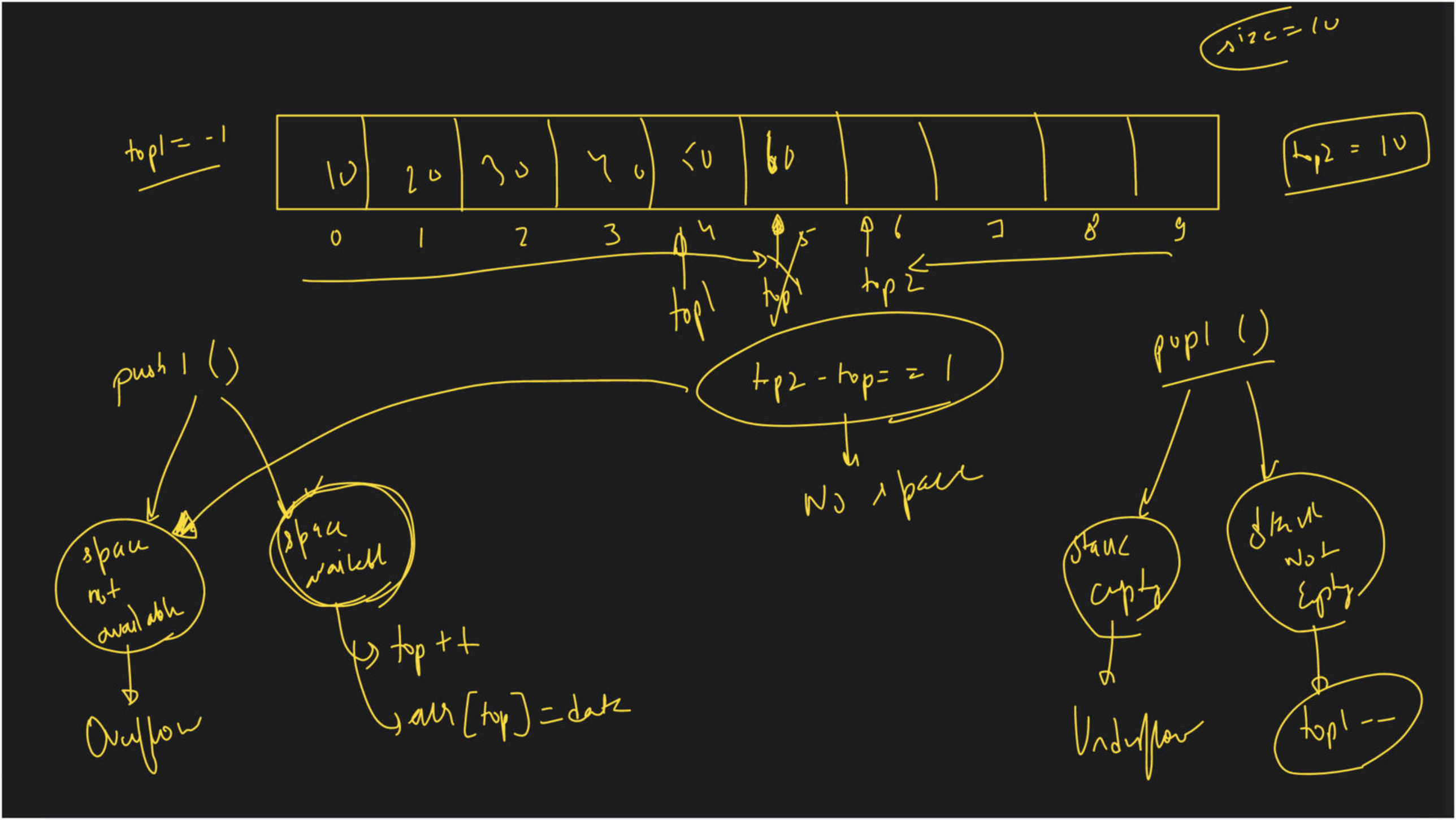

top ++

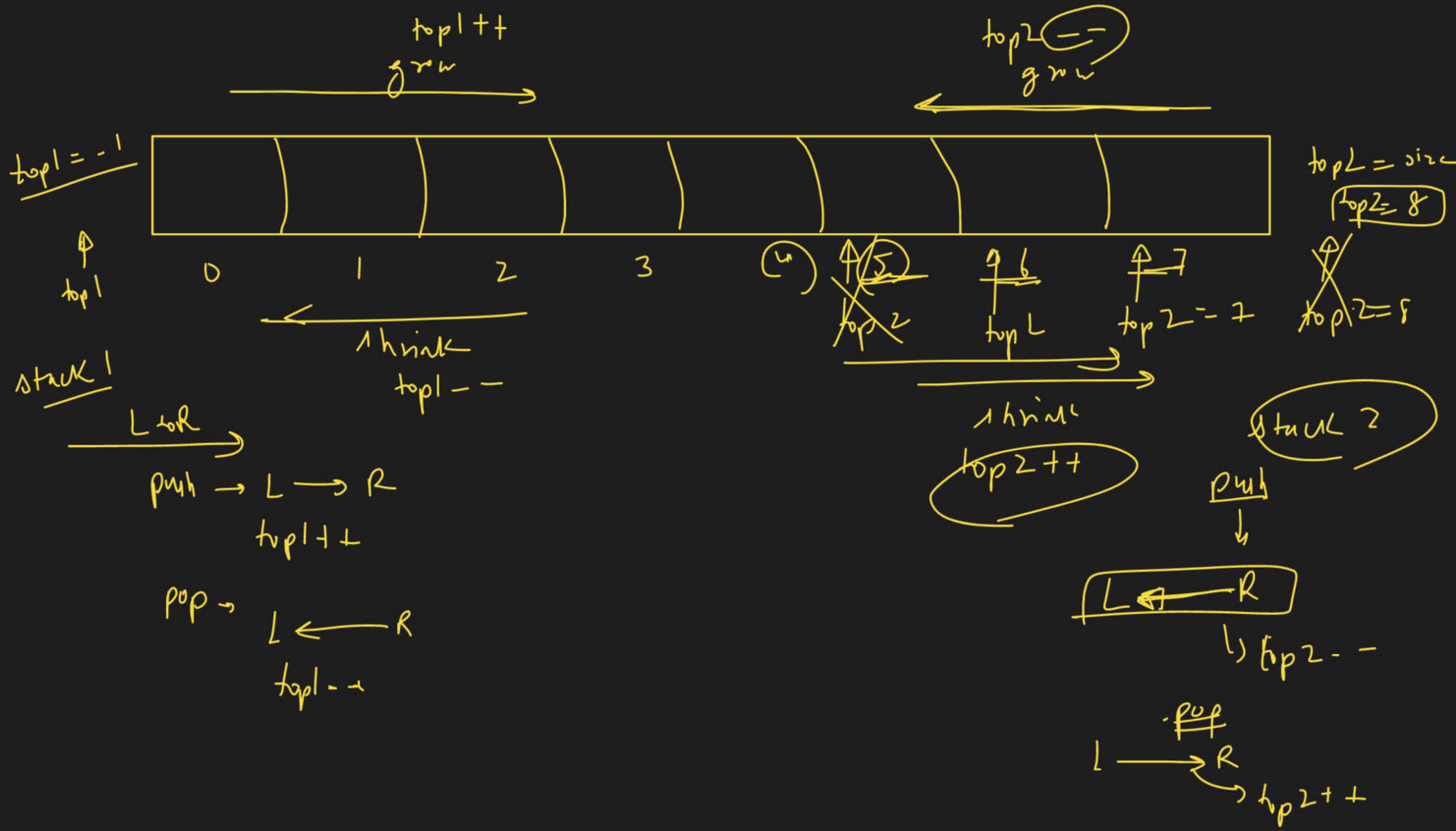
2 + 3

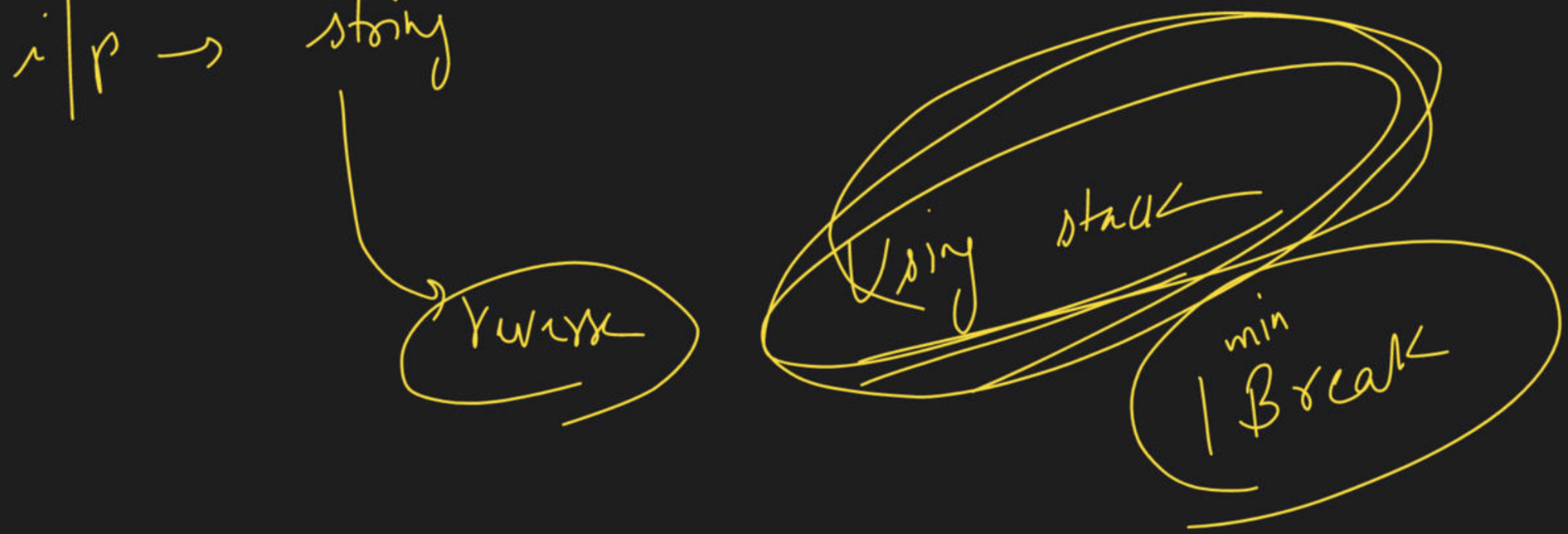
no of elmnt = 3


1
array
1 stack

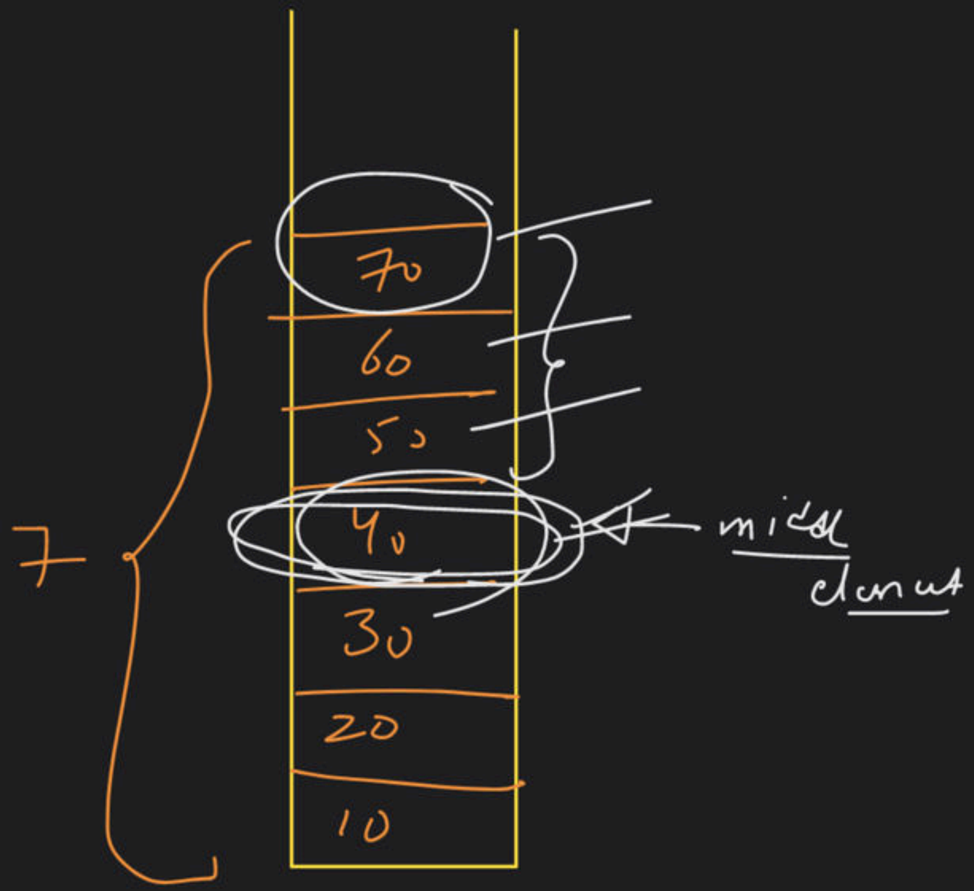

2 stacks in an array
top1 = -1
top2 = size
arr
0 1 2 3 4 5 6 7 8 9
top1
top2
stack1
stack2
top2 - top1 == 1
No space available
#1 → half - half
S1
S2
memory wastage
int * arr
int size
int top1
int top2
// functions
push1() pop1()
push2() pop2()


size = 10
top1 = -1
10
20
30
40
50
60
top2 = 10
0 1 2 3 4 5 6 7 8 9
top1
top
top2
top2 - top == 1
No space
push1 ()
space not available
Overflow
space available
top++
arr[top] = data
pop1 ()
stack empty
Underflow
stack not Empty
top1 --

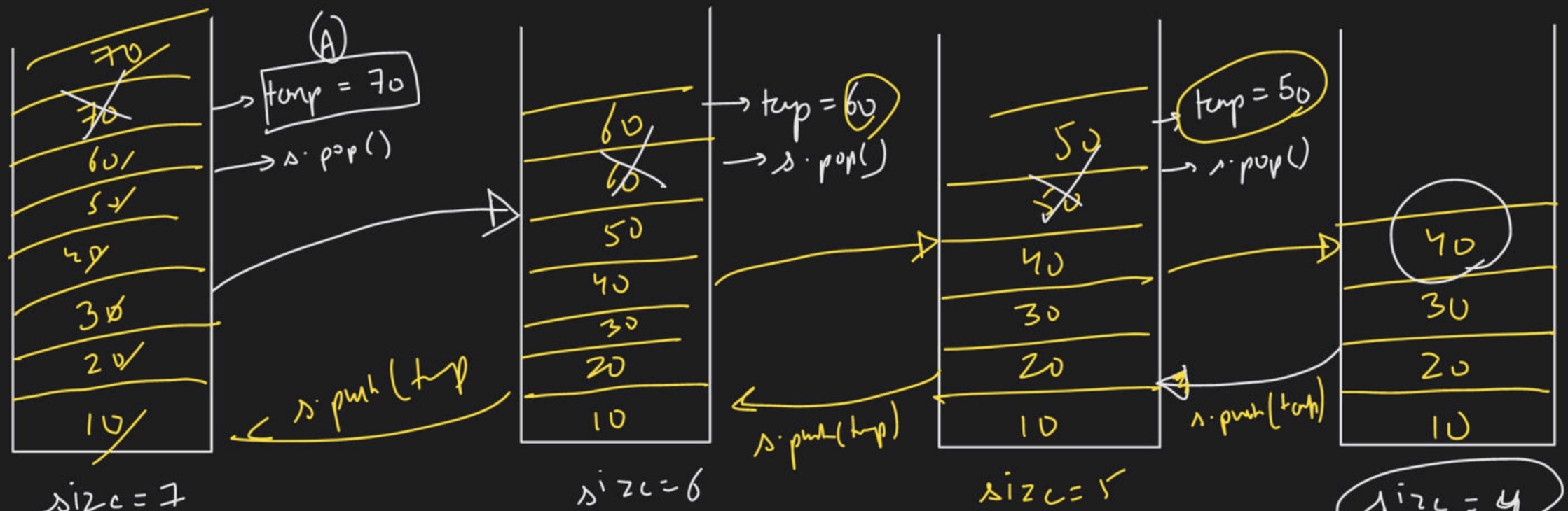

top1++
grow
top2--
grow
top1 = -1
top2 = size
top2 = 8
top1
0
1
2
3
4
5
6
7
top2
top1
top2 = 7
top2 = 8
shrink
top1--
stack 1
L to R
push → L → R
top1++
pop →
L ← R
top1--
shrink
top2++
stack 2
push
L ← R
top2--
pop
L → R
top2++


i/p → string
using stack
min
Break

Find middle element in a stack




top/2




total size = $\frac{7}{2}$ = 3 + 1

if (total size / 2 + 1 == size)

print s.top()

return