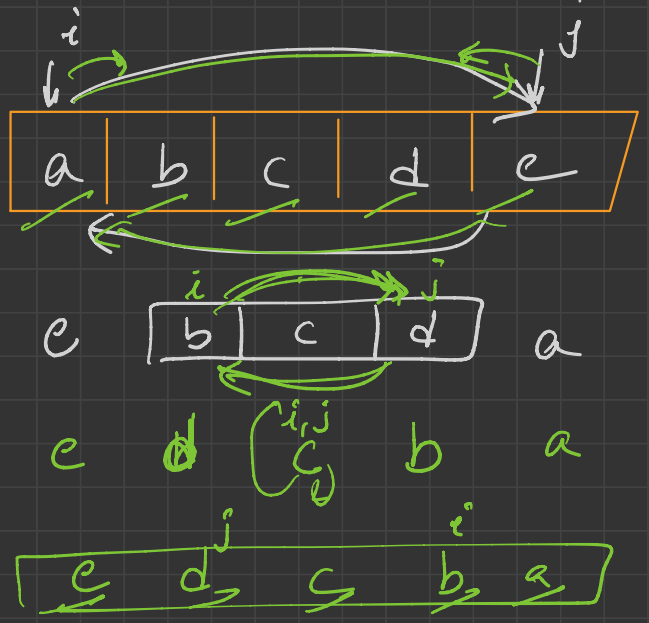
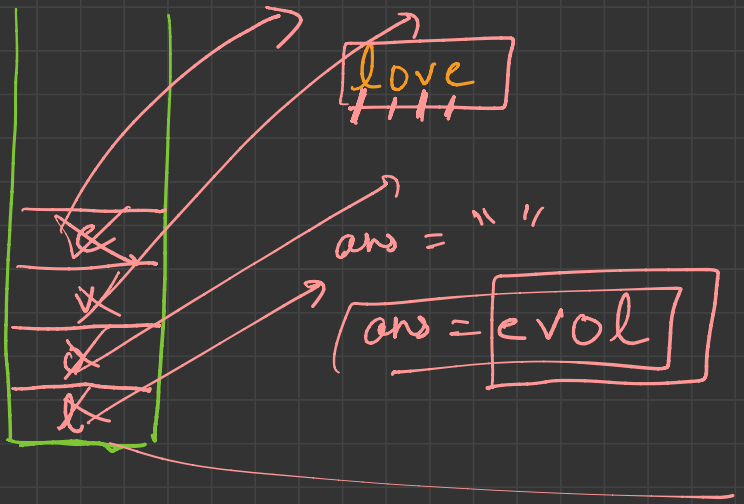
Stacks

→ Reverse a string using Stack



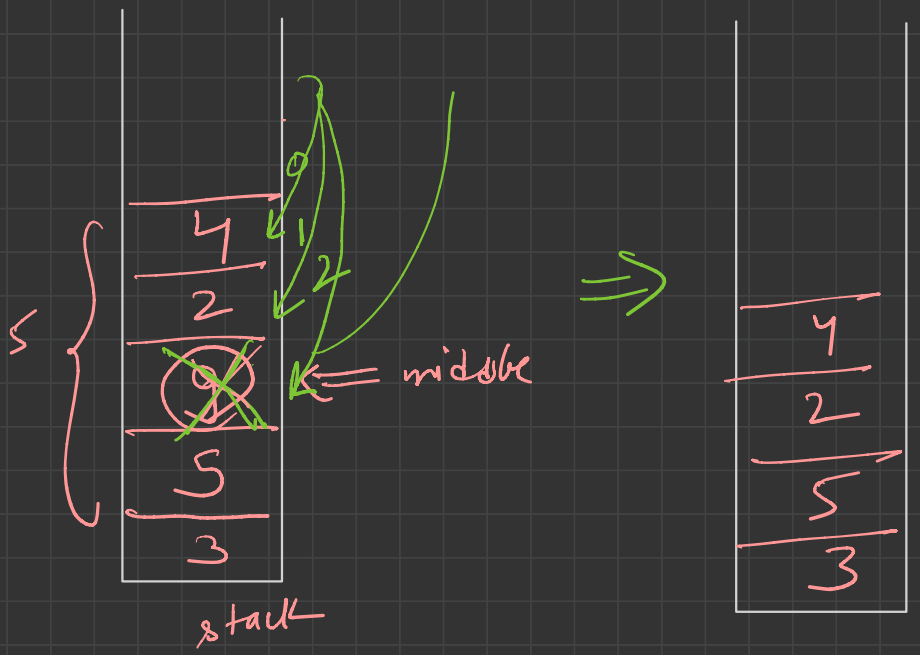
→ Reverse a string using stack → LIFO



reverse

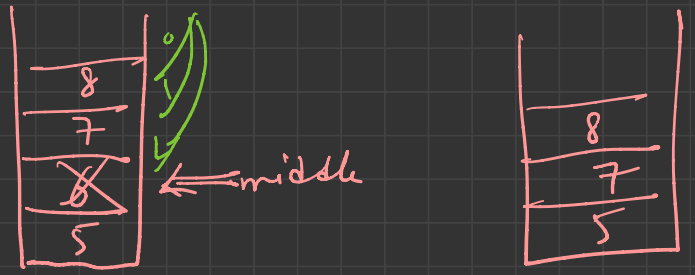
O(N)

S.C → O(N)

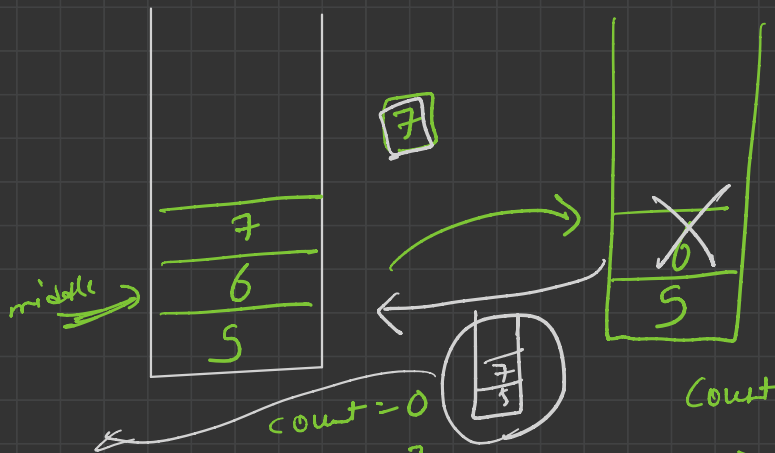


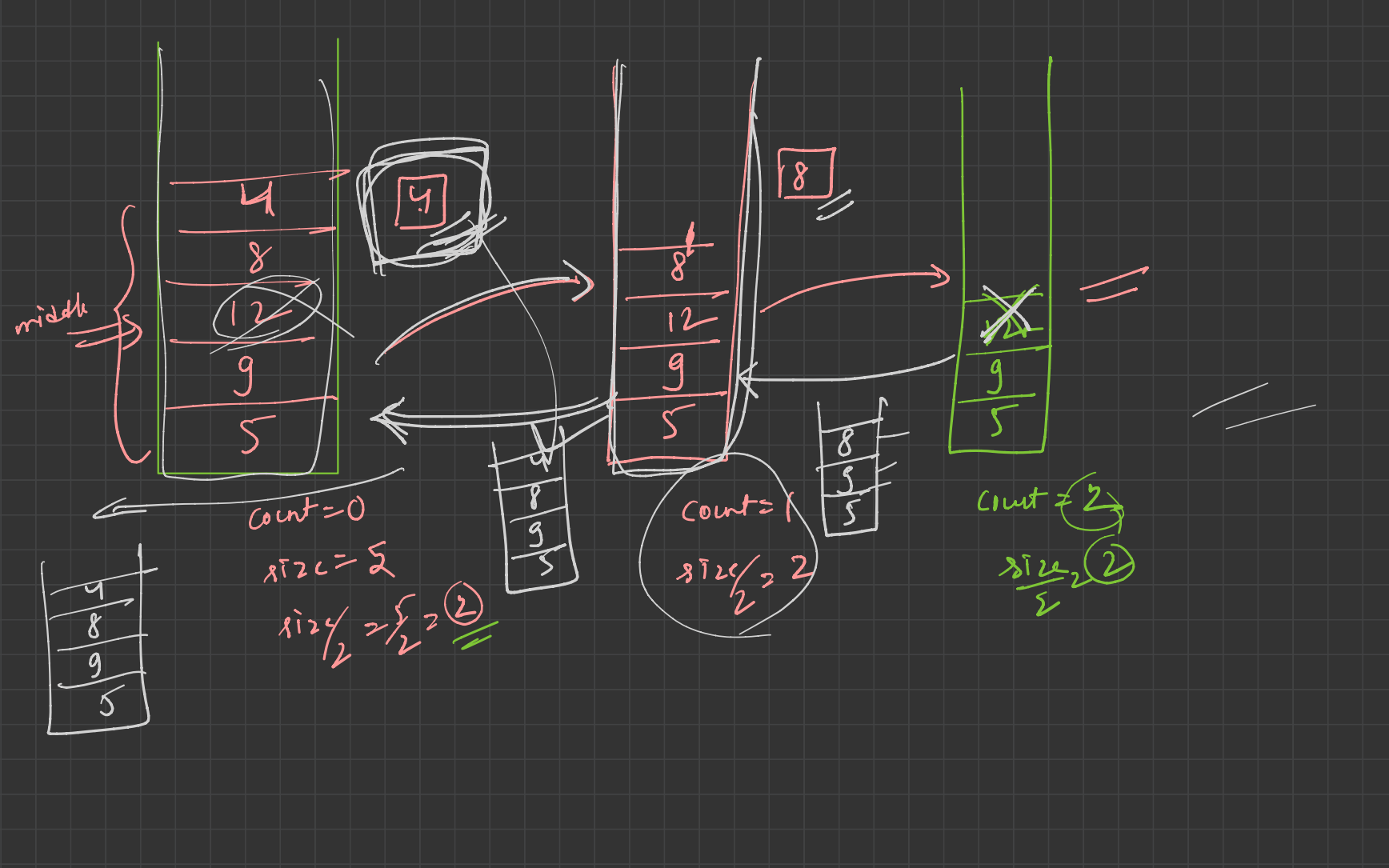
st.size() = 5

count = size/2



st.size() ≥ 4


7
7
6
5
middle
0
5
7
5
count = 0
size = 2
ans
count = 1
size = 3
size/2 = 3/2 = 1
size/2 = 3/2 = 1


4
8
12
9
5
middle
4
8
12
9
5
8
8
12
9
5
4
8
9
5
8
5
5
4
8
9
5
count=0
size=5
size/2 = 5/2 = 2
count=1
size/2 = 2
count = 2
size/2 = 2/2

→ Valid Parenthesis

{ } =

{ ( ) }

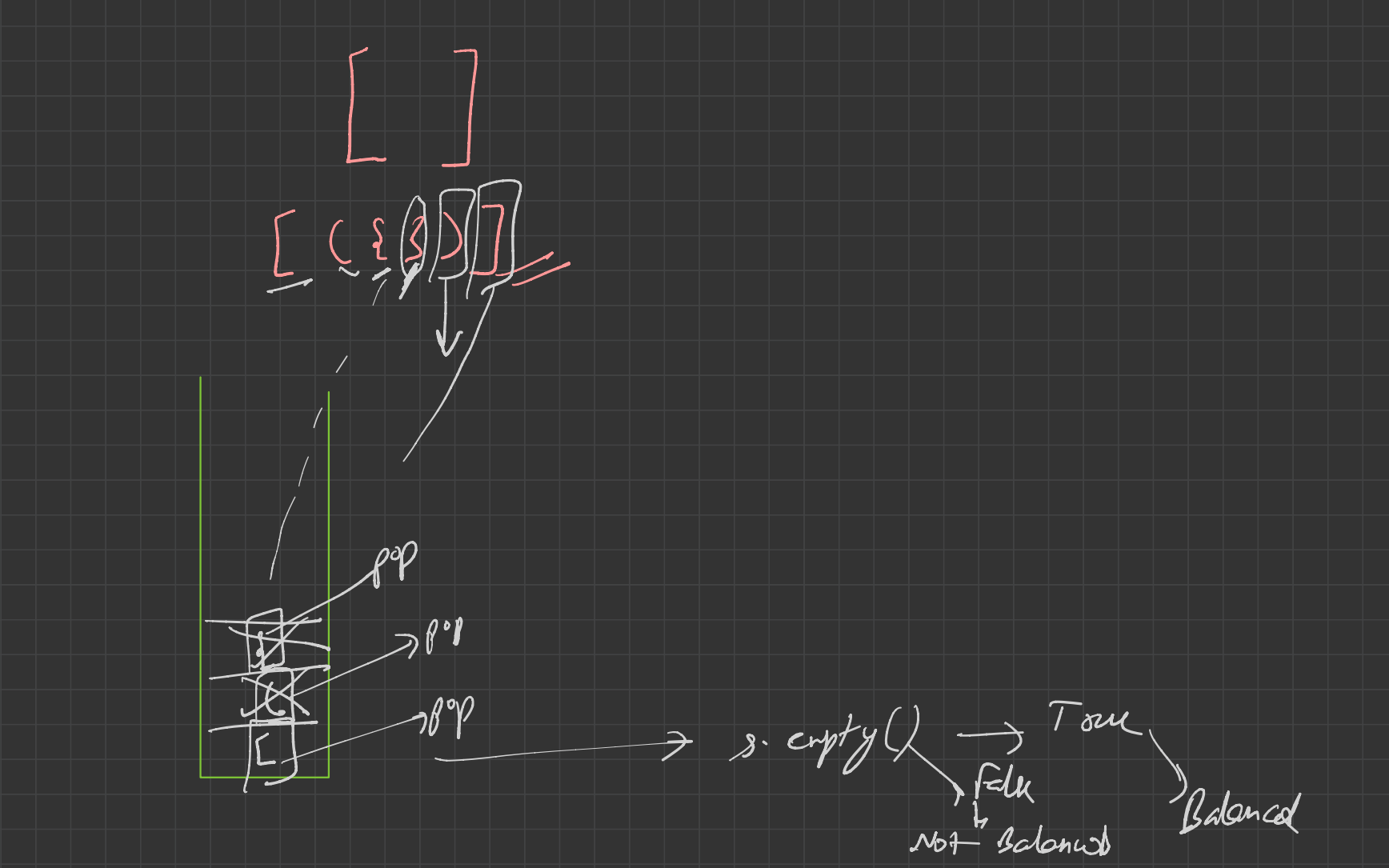
} {

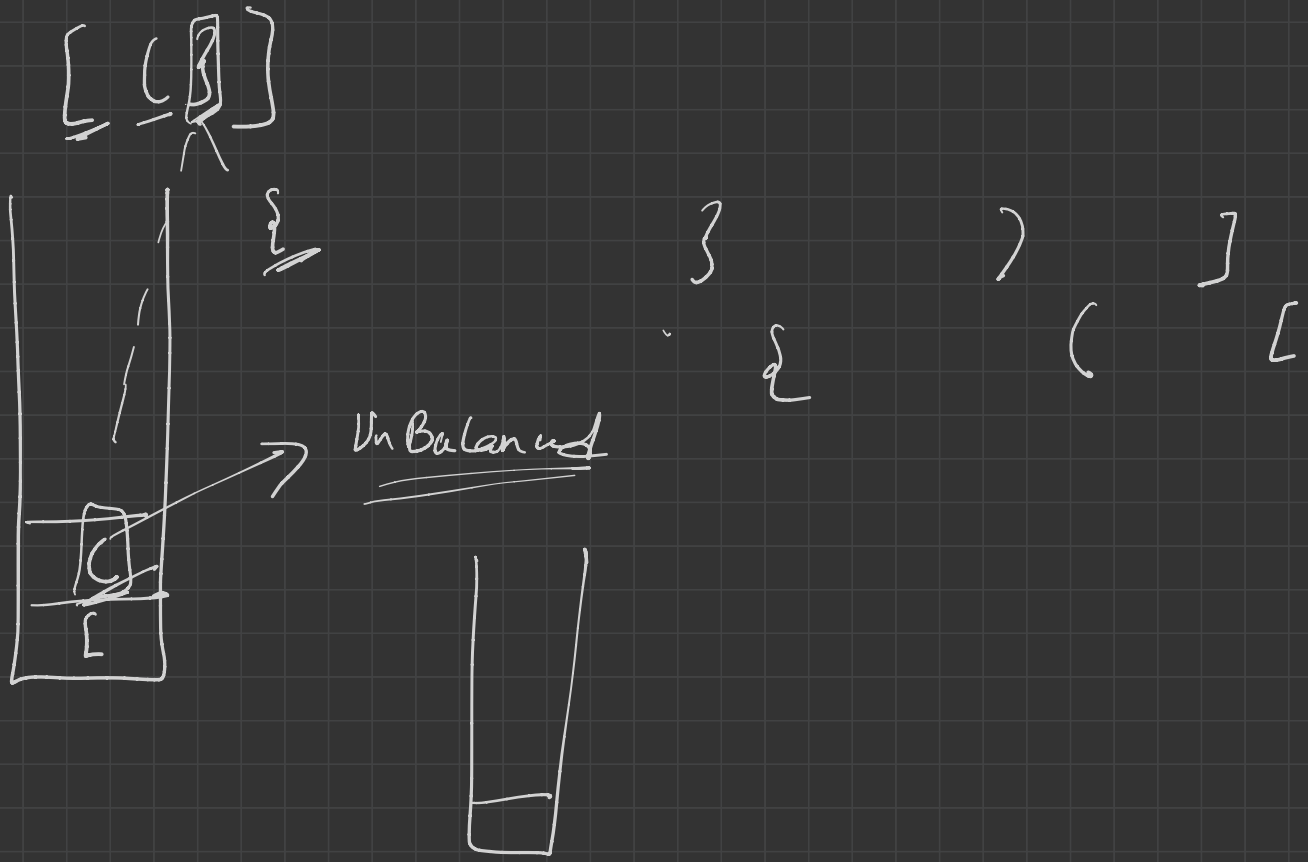
)(

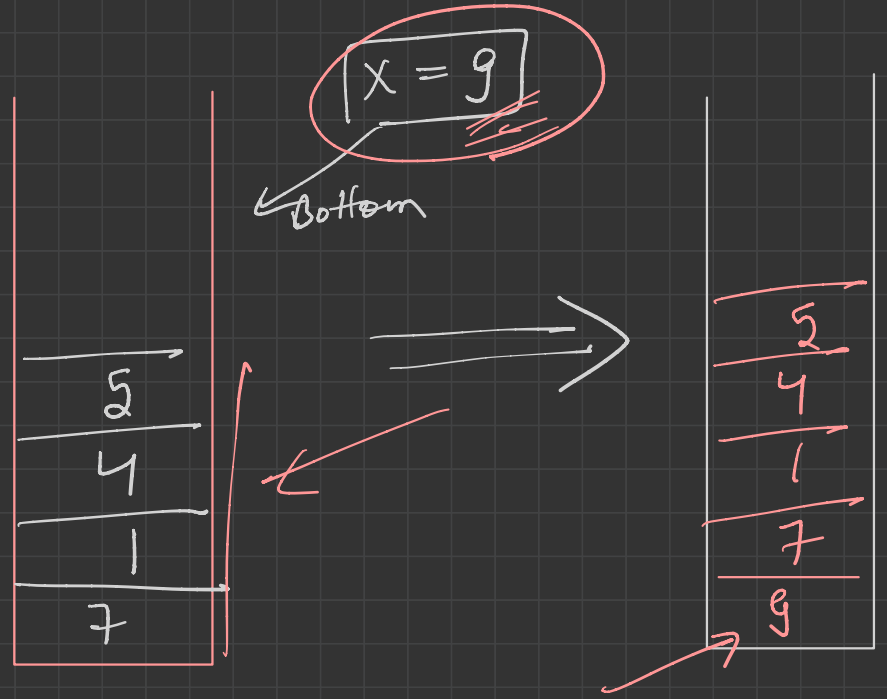
( ) =

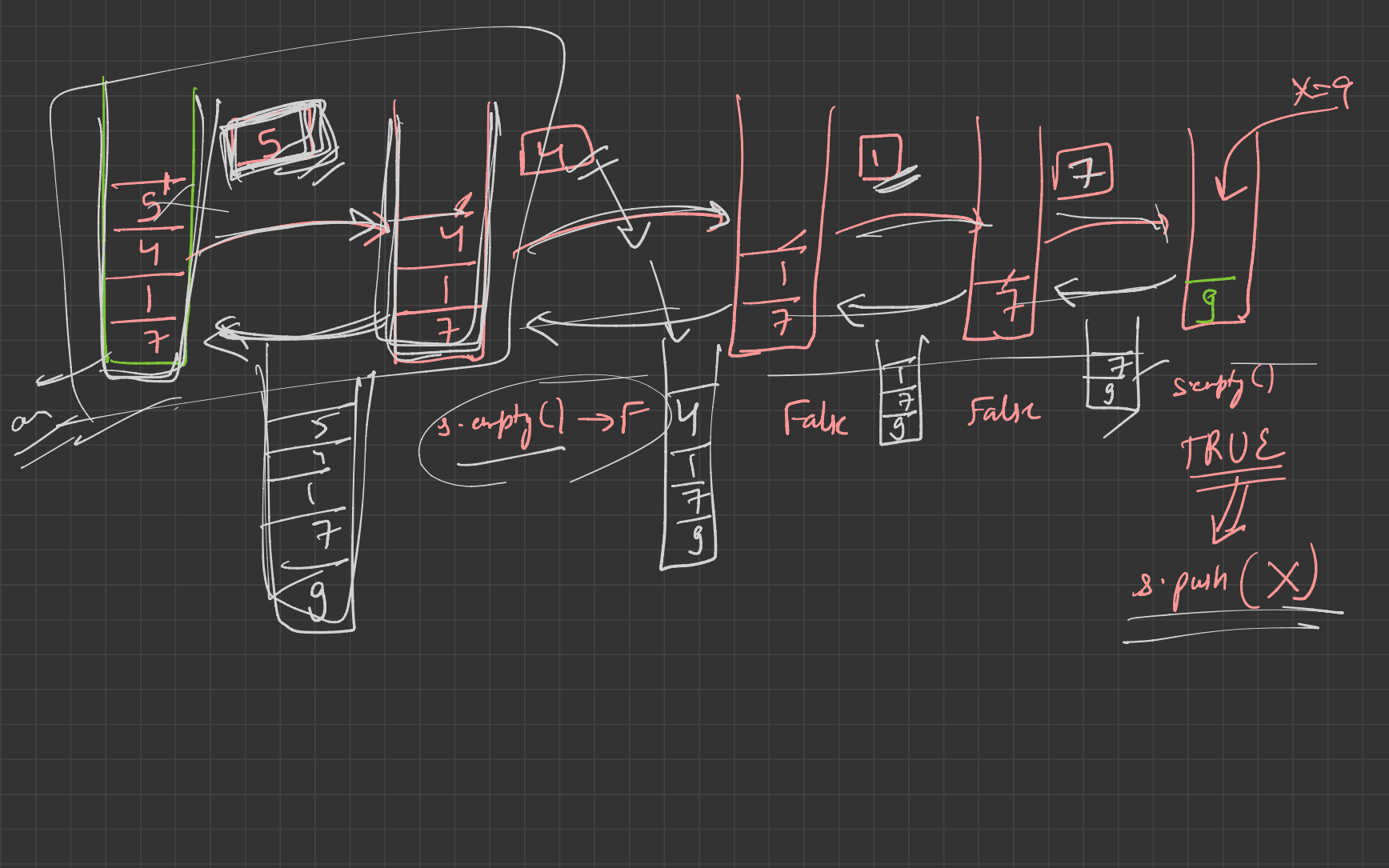
[( )]

[( )]

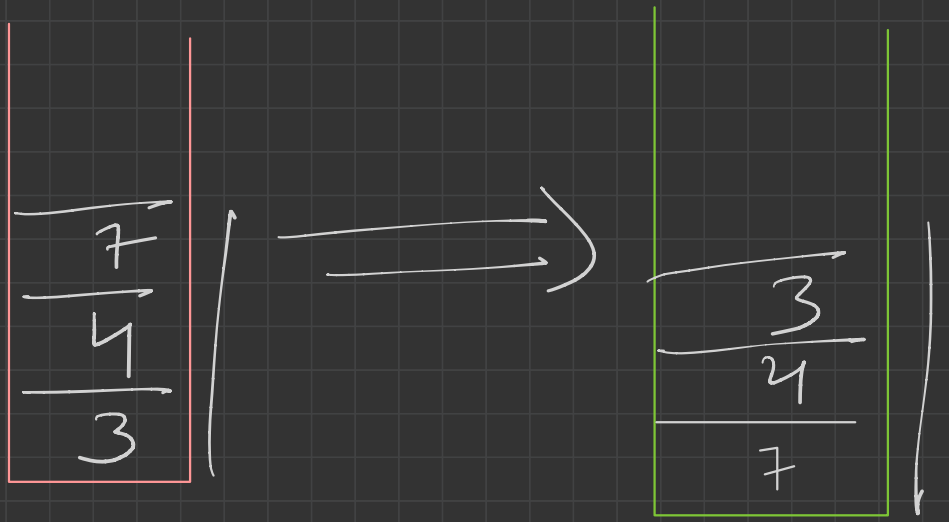

[ ]
[({ })]
pop
pop
pop
s.empty()
True
False
Balanced
Not Balanced

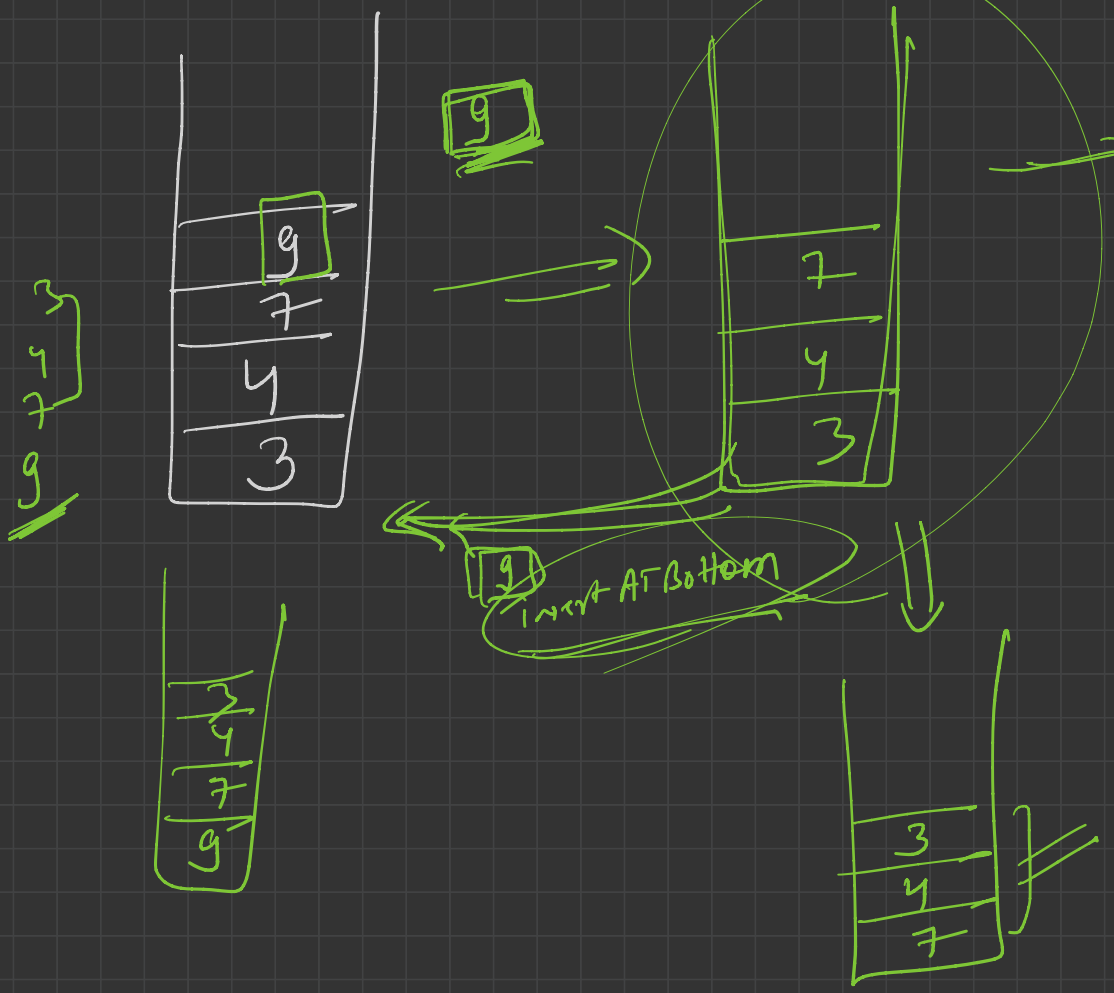

Unbalanced


X = 9
Bottom
5
4
1
7
5
4
1
7
9


5
4
1
7
9
s.empty() → F
Falsc
Falsc
s.empty()
TRUE
s.push(X)

→ Reverse stack





$O(n^2)$

S.C → H/W

Algo:-

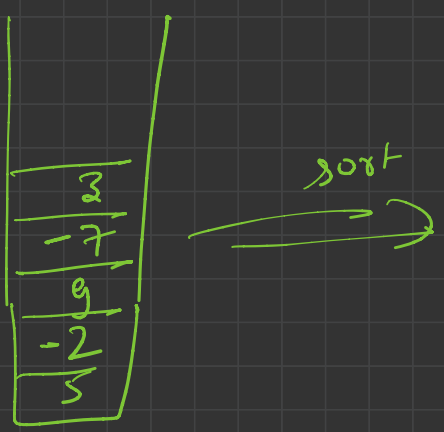
→ stackTop side me rakhlo

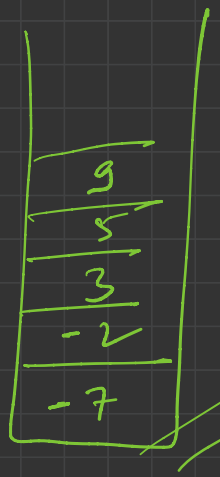
→ use recursion to reverse remaining stack

→ wapas aate hue, insertAtBottom (same stack TOP)

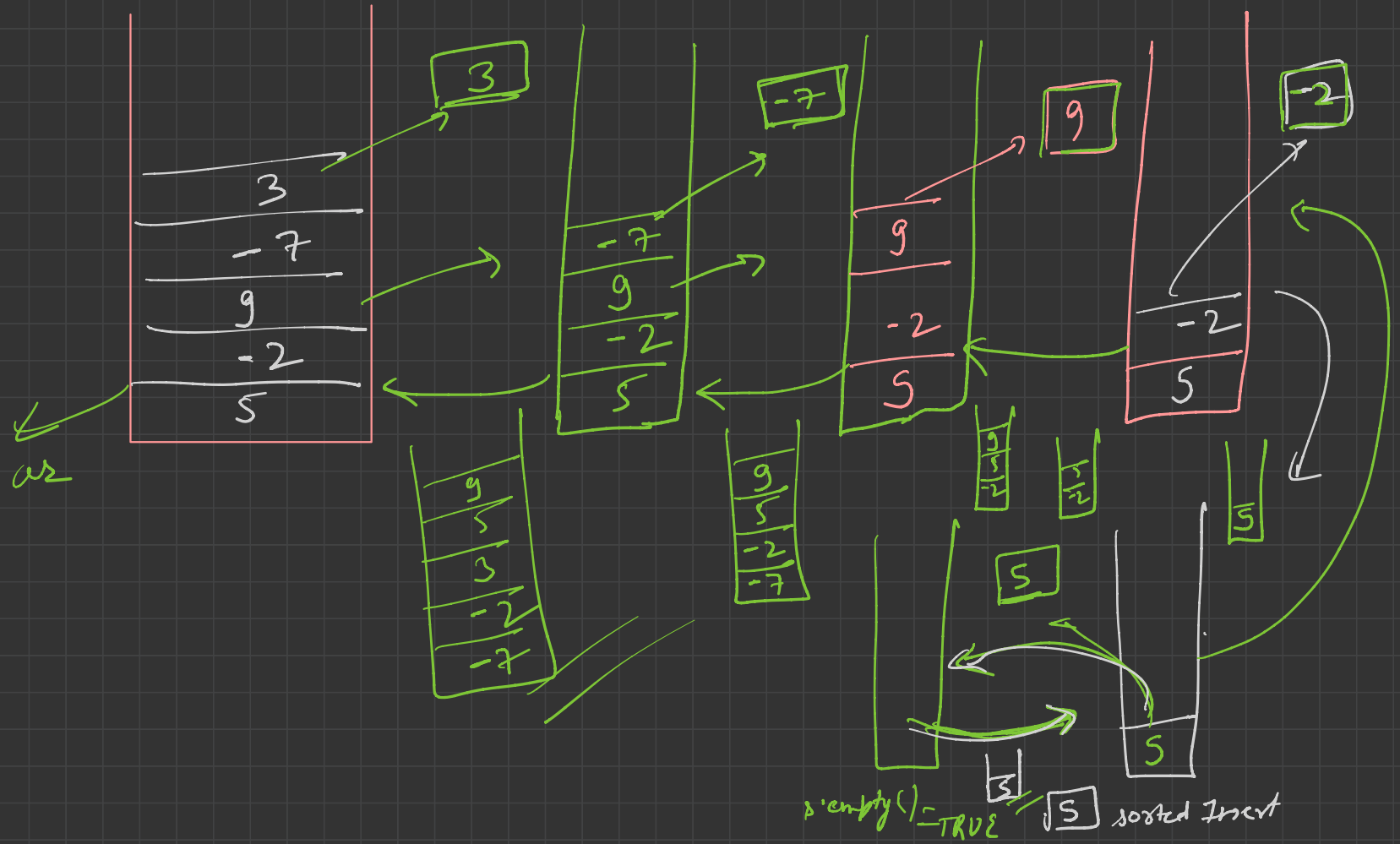
→ return ans;

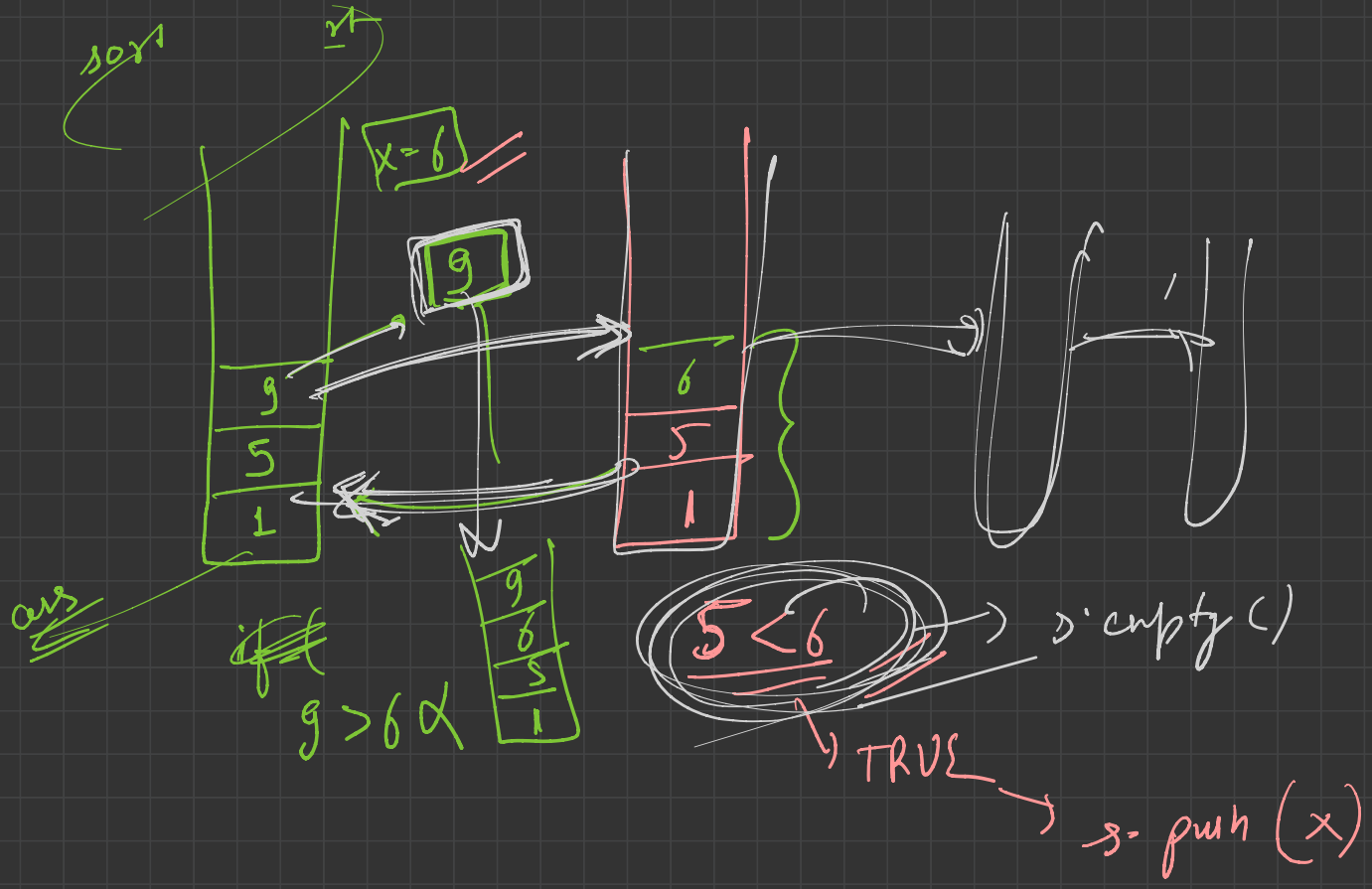
→ Sort a Stack

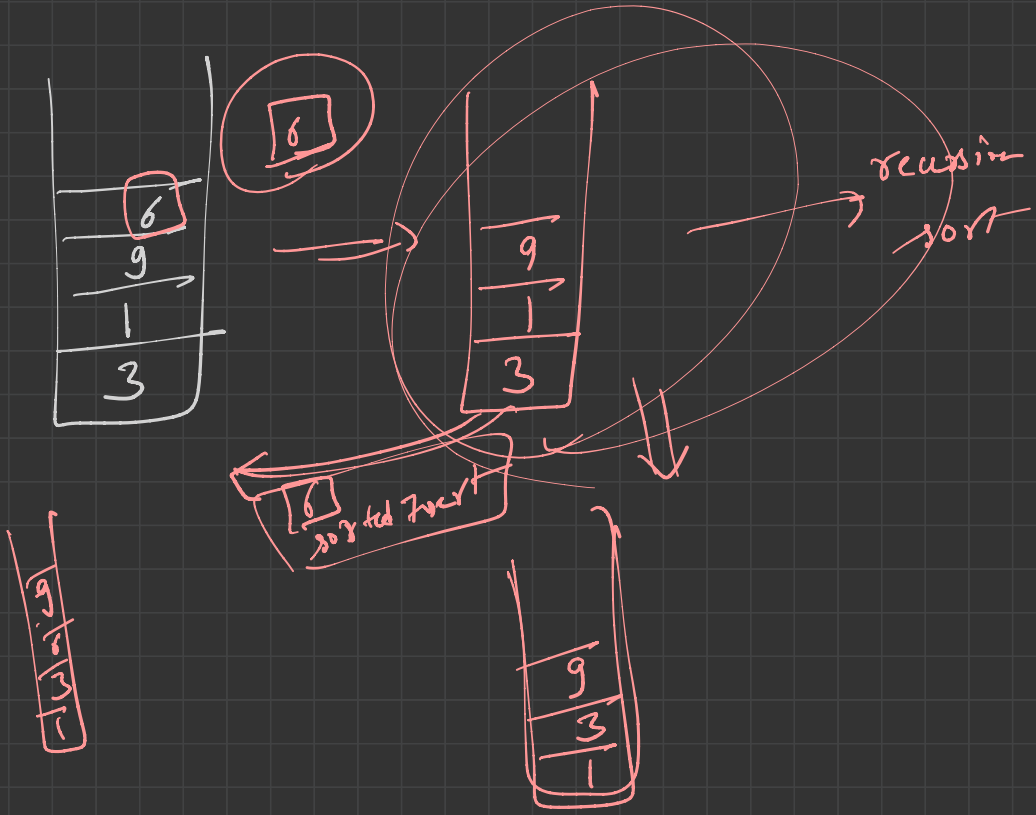


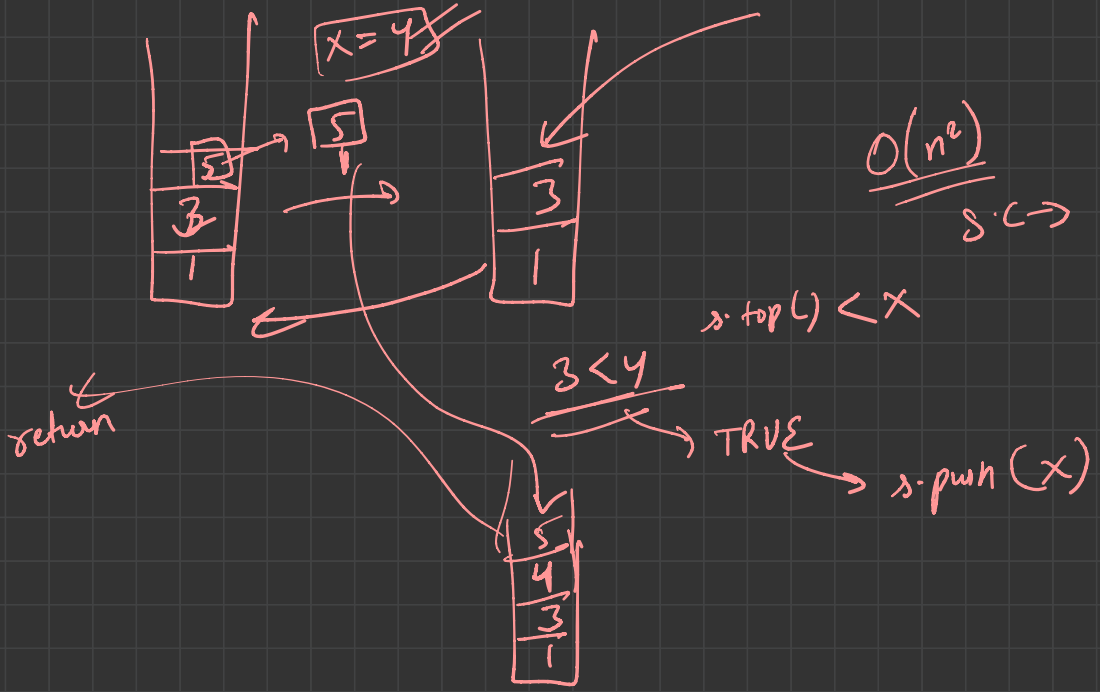


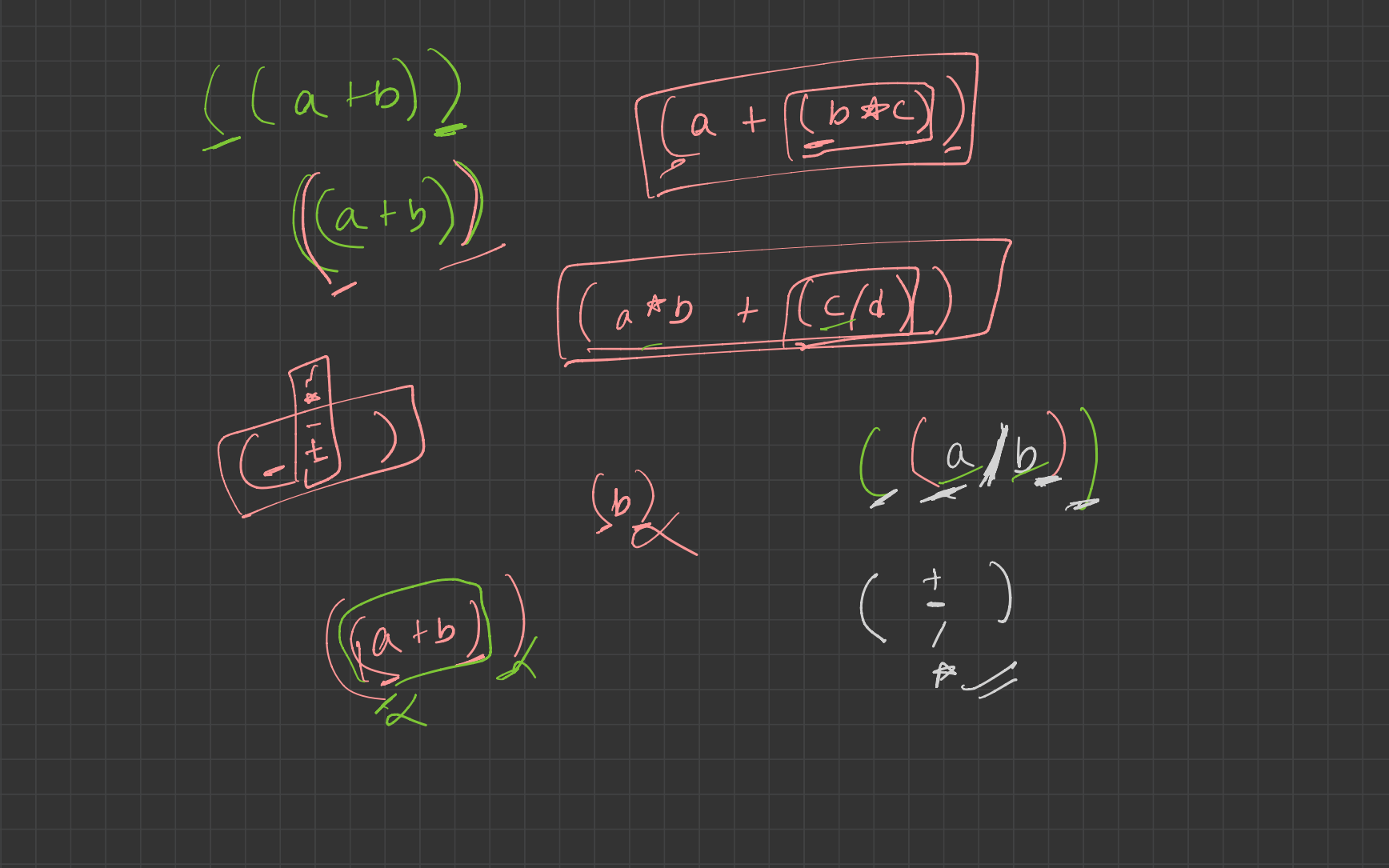
sorted Insert


3
-7
9
-2
5
as
s.empty() = TRUE
sorted Insert


sort
x=6
9
9
5
1
6
5
1
aux
if
9>6 ✓
9
6
5
1
5<6
s.empty()
TRUE
s.push(x)

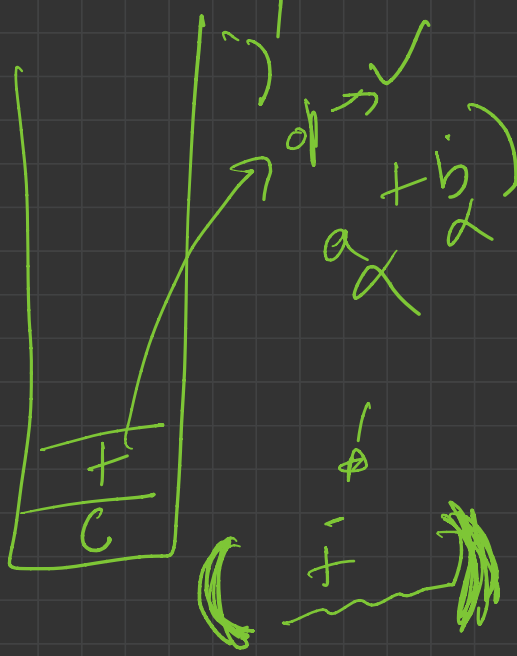

6
6
9
1
3
recursion
sort
9
1
3
6
sorted stack
9
3
1
9
3
1


x=4
5
5
3
1
3
1
O(n²)
s.c→
s.top() < x
3<4
TRUE
s.pwn(x)
return
5
4
3
1


((a+b))
((a+b))
(a + (b*c))
(a*b + (c/d))
(b)
((a/b))
((a+b))

→ i/p → string = "(a+(b*c))"

Approach:-



for ( )

{

ch = str[i];

if (ch → '(' || operator)

stack.push (ch)

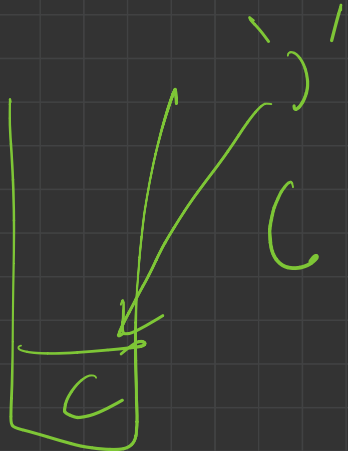
else

{

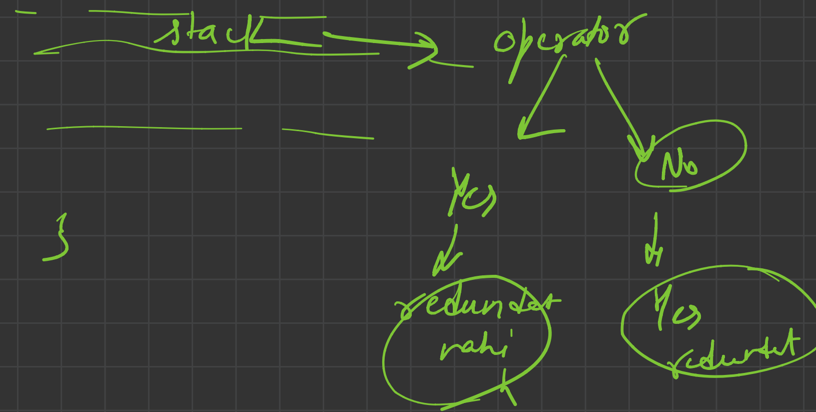
) or lower letter

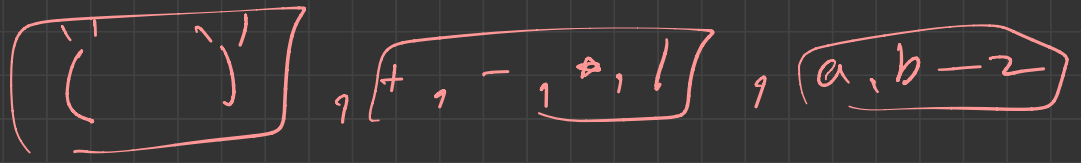
if (ch = = ')'

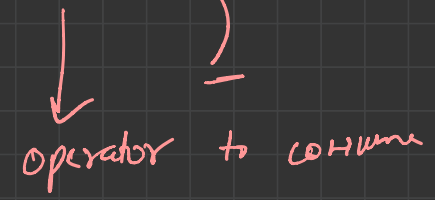
}



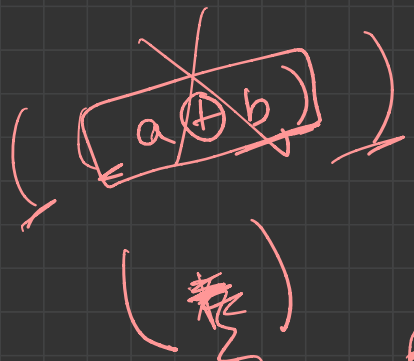
+ , - , * , /


stack
operator
Yes
No
redundant nahi
Yes redundant

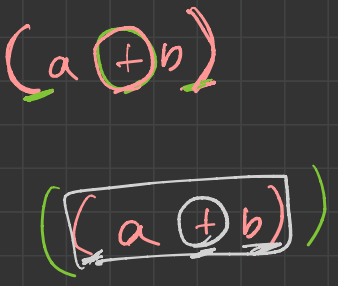
Logic:- 

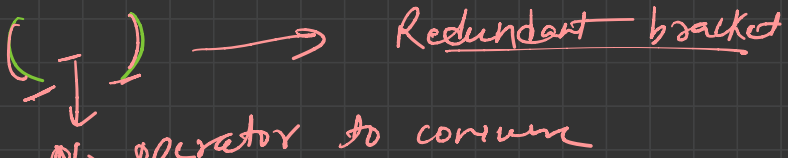
Redundant bracket → 

No operator to consume



No op⁻¹, A.B





No operator to consume

stack s:

((a + b))

'(' '+' '-' '*' '/'

s.push

if ')' → it is certain ki stack me open bracket bhi hoga

jab tak stack me open bracket nahi milta

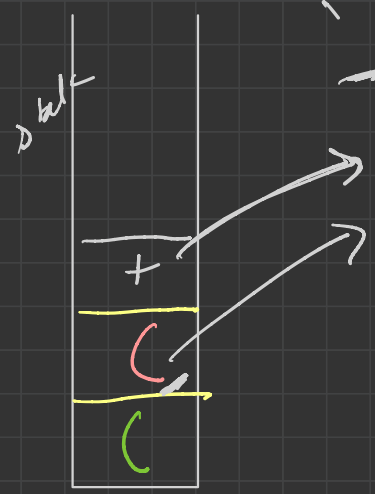
↳ stack top check karo for operator

Found → Redundant pair

Not found → Redundant pair

((a + b)))

DRY RUN



operator → R.B → No

')'

'('

operator Not found

↳ R.B →

return TRUE

((( )))

T.C -> O(N)

S.C -> N/A

→ i/p expression string → '{' '}'

{ } { } { { } }

{ } { { {

Not Valid

{ } { }

{ { } } { }

{ { } }

→ condition

$3 + 3 = 6$ $\quad 4 + 4 = 8$ even

no° of open = no° of close

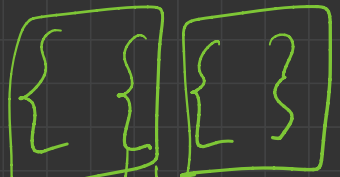
every close brace should have open brace before it

{ } { }

{ → }

} → {

str =

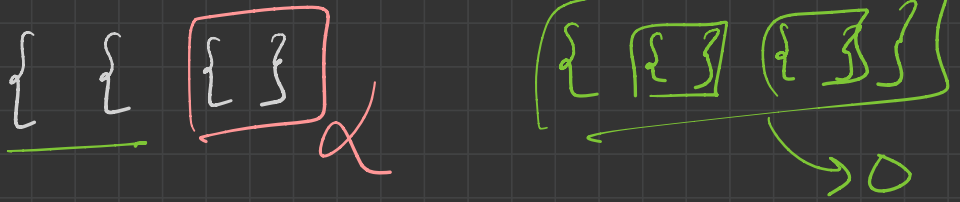


1

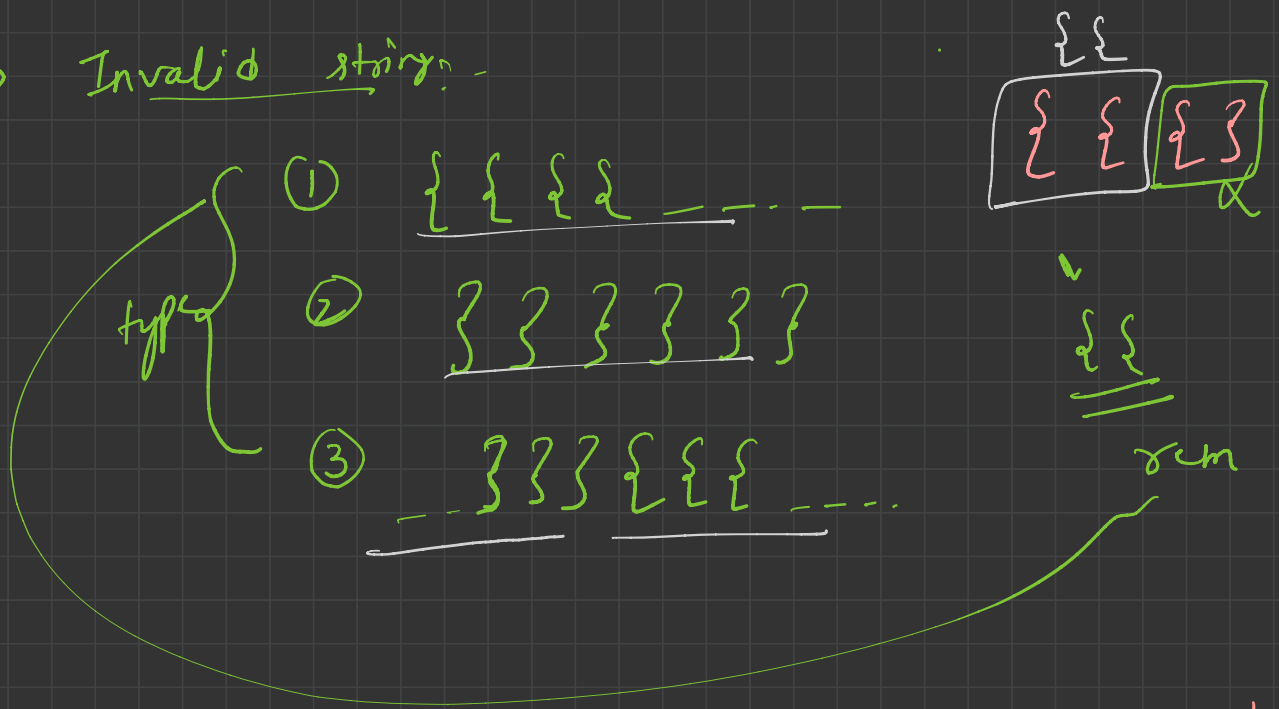
{ } { } → Valid

Logic:- if str length → odd

↳ return -1;



→ Invalid string:-



Algo:- (I) odd → -1

(II) i/p string → valid part remove → stack

↳ remaining part → above 3 types

(III) a & b count, ans = $\left(\frac{a+1}{2}\right) + \left(\frac{b+1}{2}\right)$ → return ans;

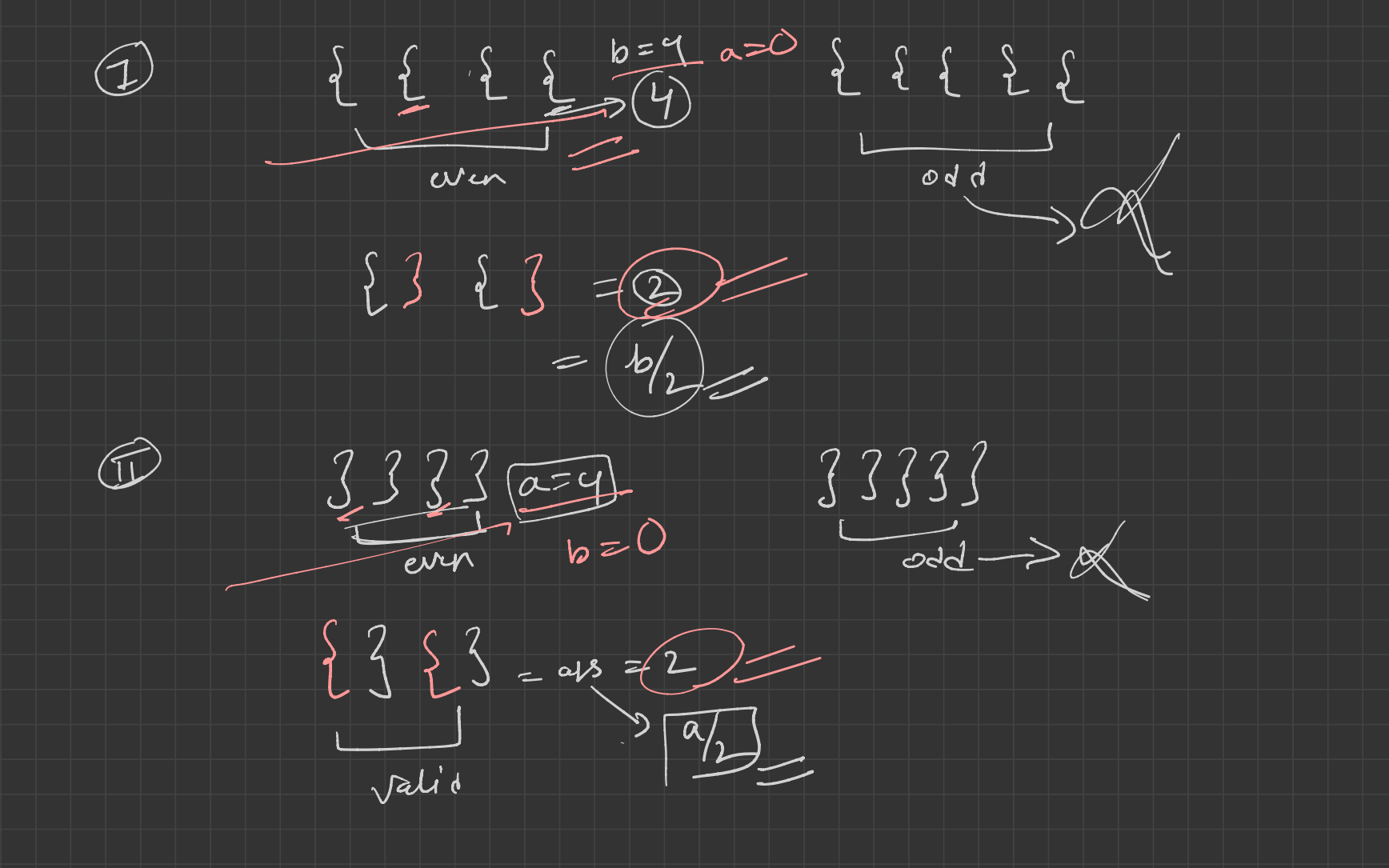
}}} {{{

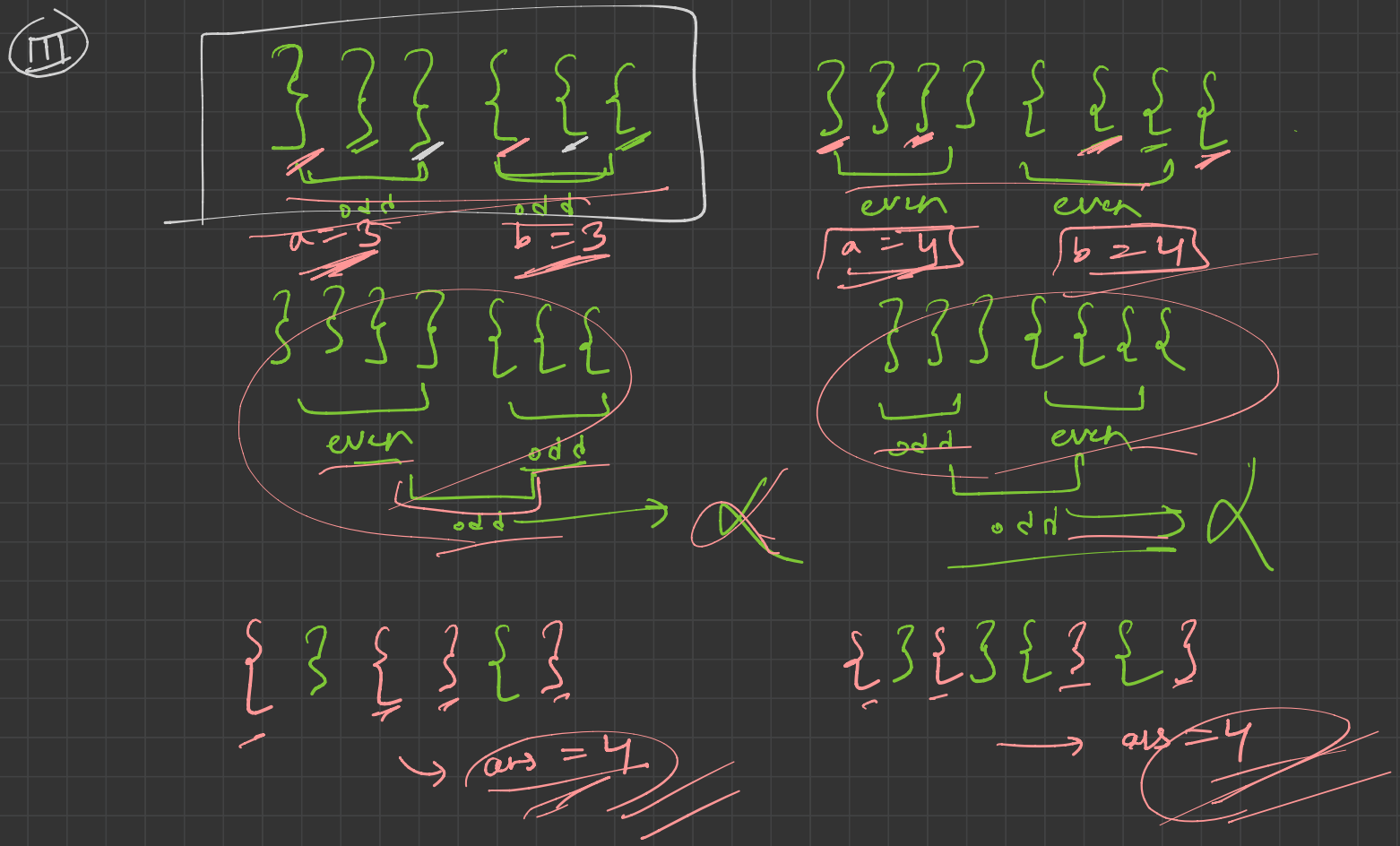
a → count of close braces

3

b → count of open braces

3


I
b=4
a=0
4
even
odd
2
b/2
II
a=4
even
b=0
odd
valid
a/2


odd
odd
a=3
b=3
even
even
a=4
b=4
even
odd
odd
odd
even
odd
ans=4
ans=4

$$ans = \left(\frac{a+1}{2}\right) + \left(\frac{b+1}{2}\right)$$

$$= \frac{0+1}{2} + \frac{4+1}{2} = \frac{5}{2} = 2$$

$$= \left(\frac{4+1}{2}\right) + \left(\frac{0+1}{2}\right) = \frac{5}{2} = 2$$

$$= \left(\frac{3+1}{2}\right) + \left(\frac{3+1}{2}\right)$$

$$= \frac{4}{2} + \frac{4}{2} = 2+2 = 4$$

$$= \left(\frac{4+1}{2}\right) + \left(\frac{4+1}{2}\right) = \frac{5}{2} + \frac{5}{2} = 2+2 = 4$$