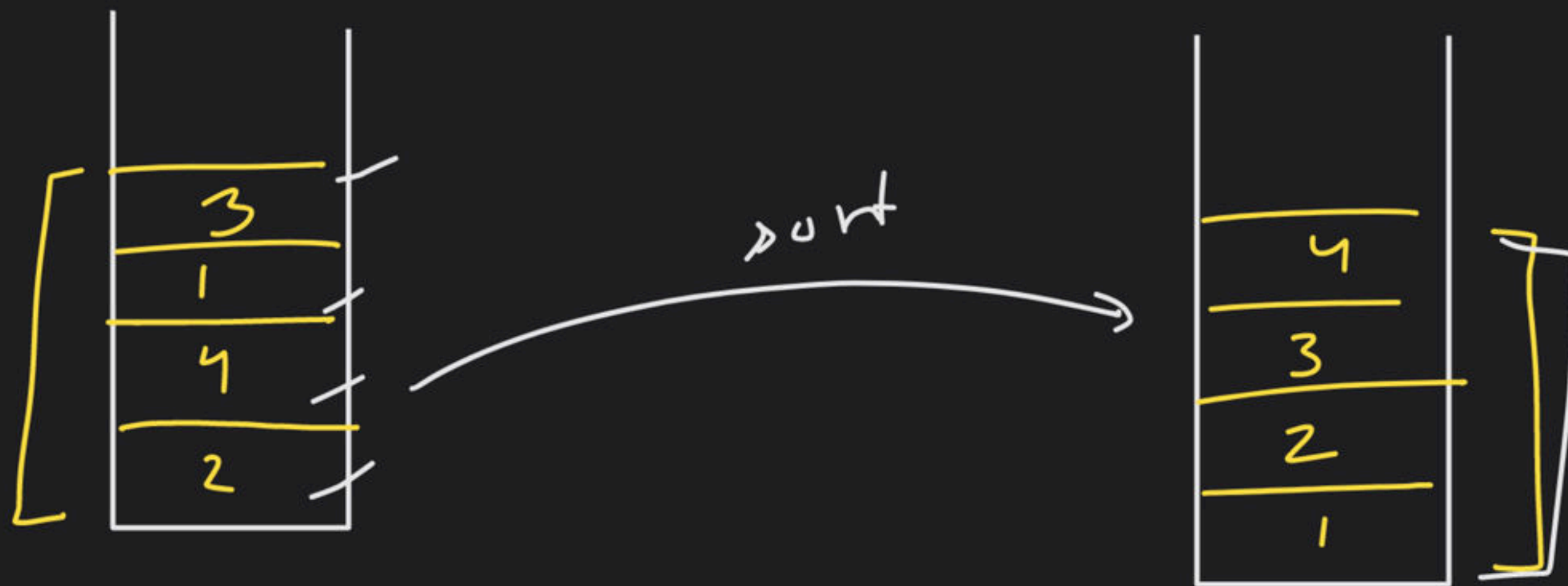
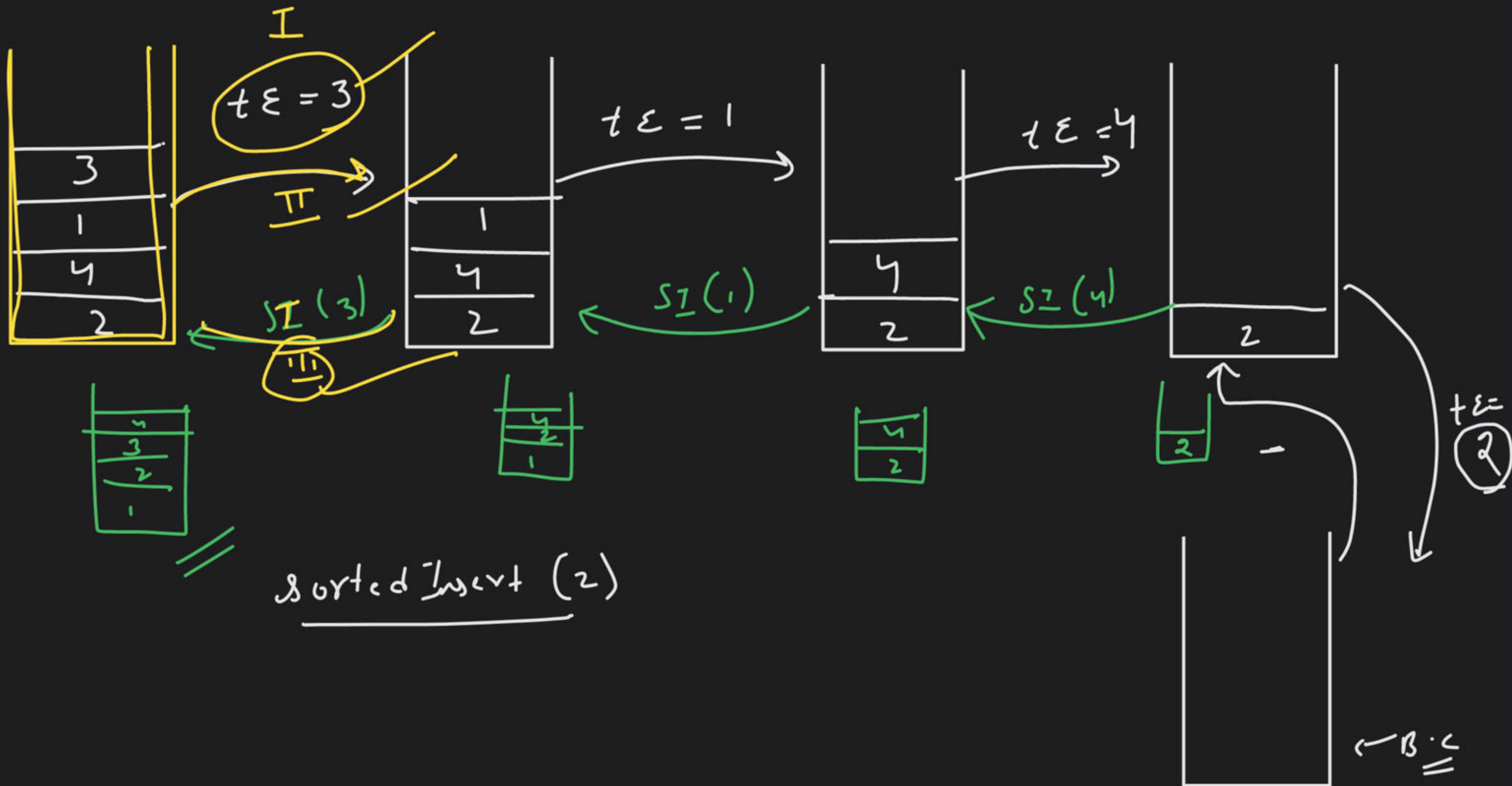


1.5hr

Foundation Course on Data Structures & Algorithms - Part II

① Sort a Stack





data = 3

data = 3

4
2
1

4

3
2
1

4
3
2
1

3 > 2

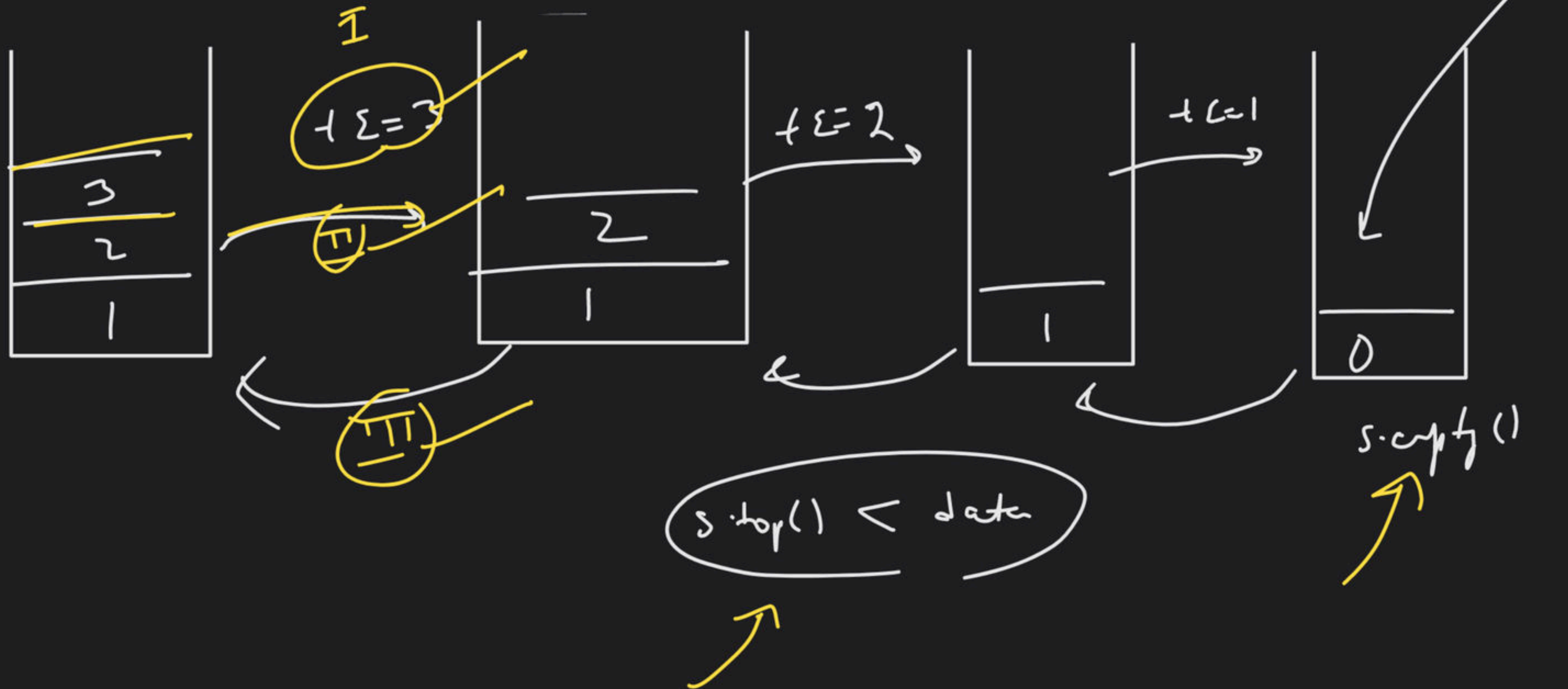
s.push(3)
return

s.top() < data
||
s.empty()

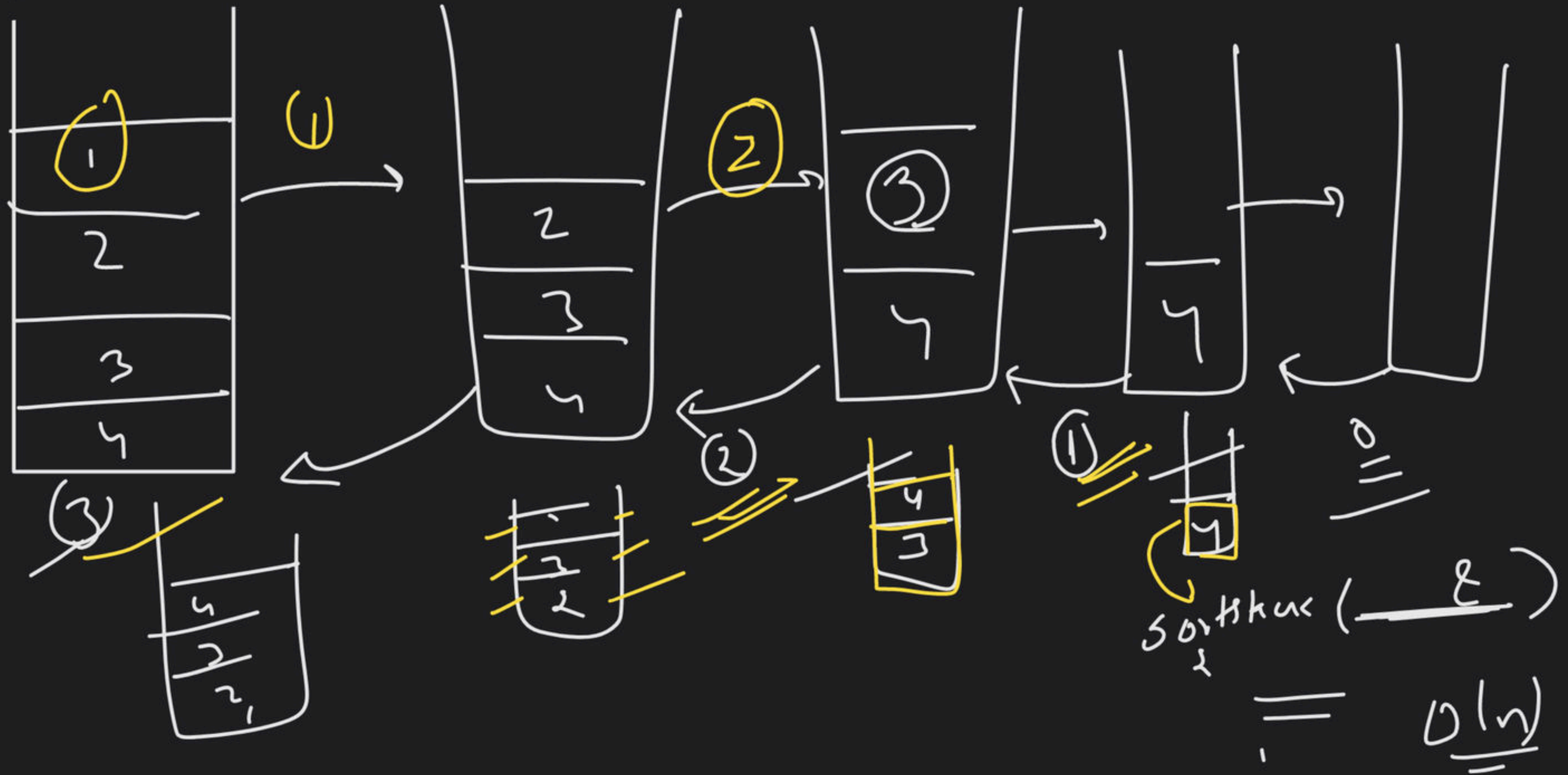
s.push(data)
return

T.C. → ?

$data = 0$



$$1 + 2 + 3 + \dots + n = \boxed{\frac{n \times (n+1)}{2}} \quad \text{--- } (n) \quad \underline{\underline{O(n^2)}}$$



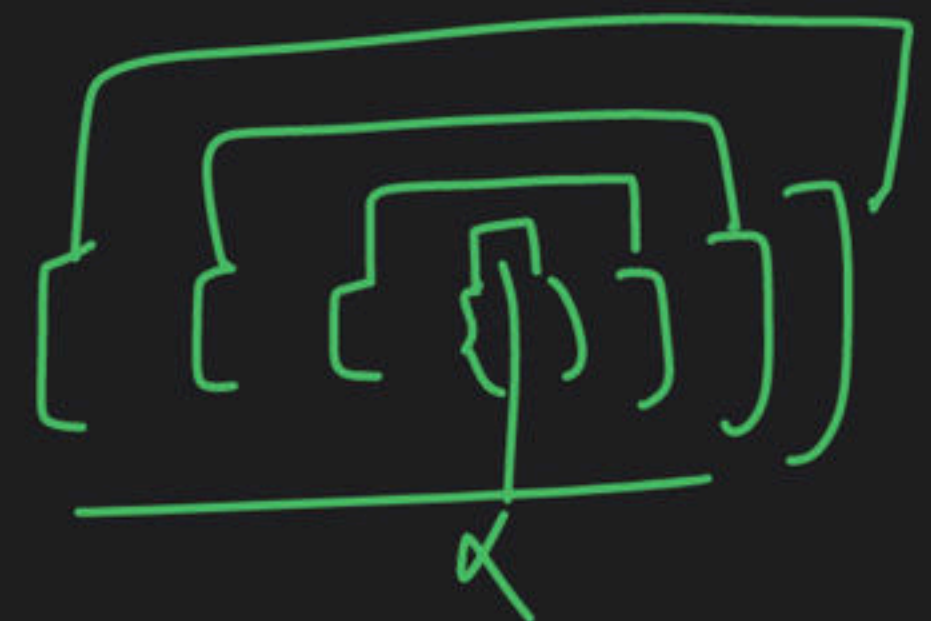
Valid parenthesis \rightarrow Balanced or not
 \rightarrow i/p

()
{ }
[]

ex \rightarrow i/p \rightarrow { }

{

{ ([]) } \rightarrow



} (

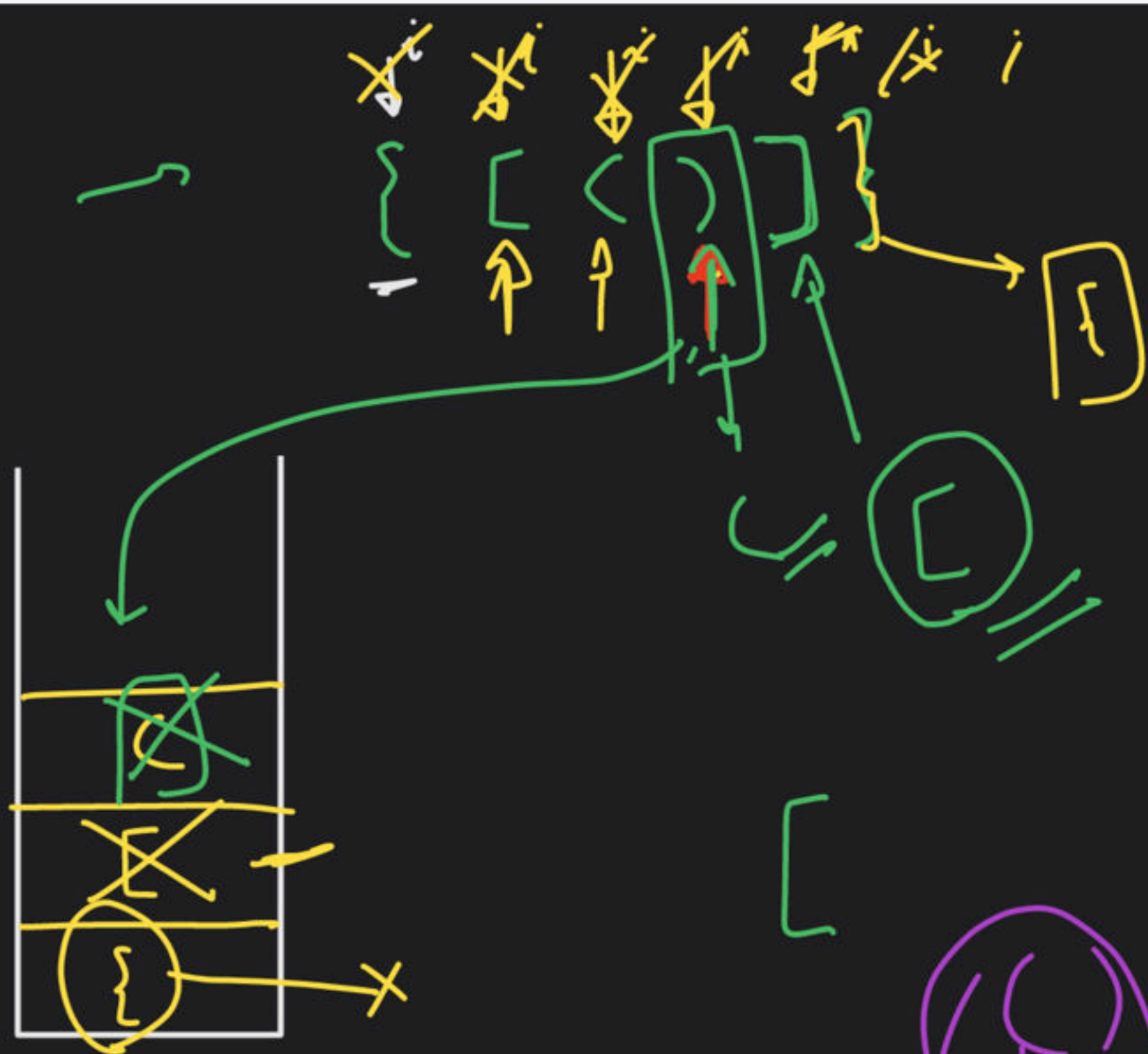
{ ([]) } =

[] ()

} {

) ([]

{ }

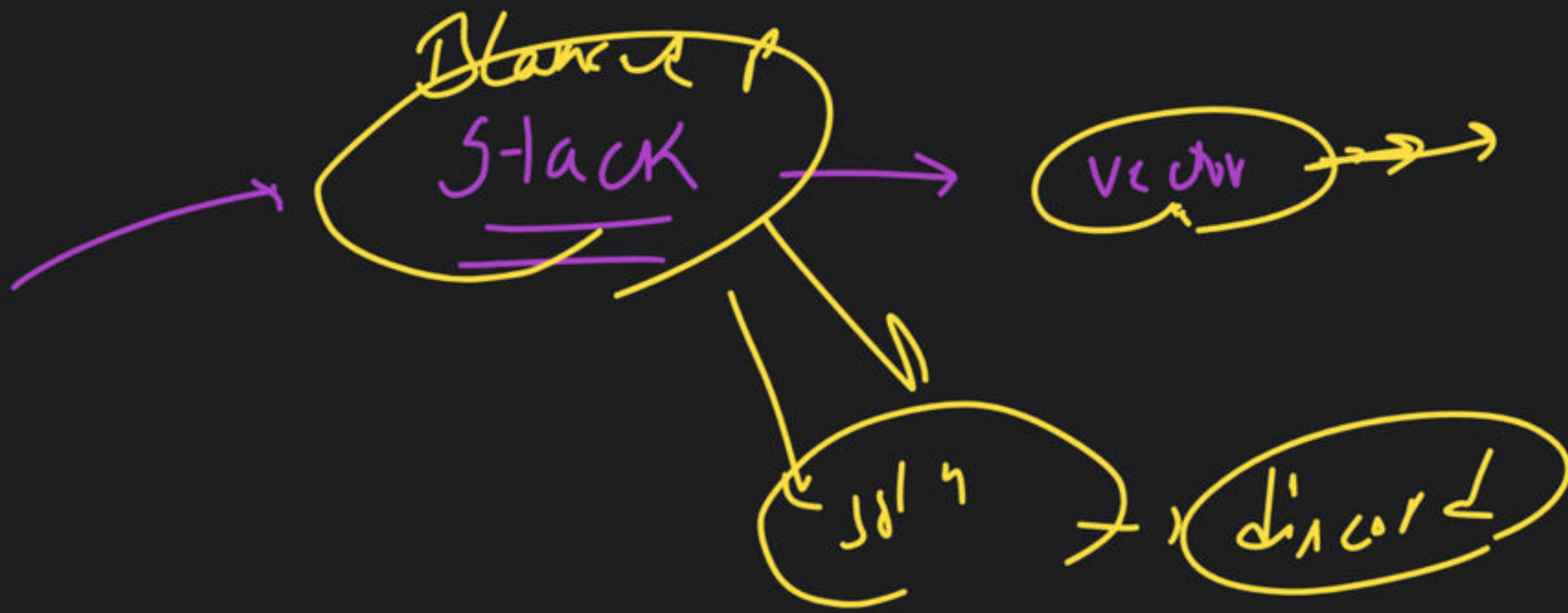


$O(n)$

```

for (int i = 0; i < s.length(); i++)
{
    if (open Brace)
    {
        s.push(ch);
    }
    else
    {
        if (s.top() == match)
            s.pop();
        else
            return false;
    }
}
if (s.empty())
    return true;
else
    return false;

```

→ Redundant brackets :-

~~(((a+b)))~~

() + - * /

i/p → s → match exp

ex s → (2+3).

(2+(3+1))

⌘ (2+1) ⌘

(2)

((((a+b)))

$$\left(a + \underbrace{(b * c)}_p \right)$$

$$\left(\overset{p}{\underbrace{(b * c)}} \right)$$

$$\left(\underbrace{(a * b)}_p + \underbrace{(c / d)}_p \right)$$

DST

Longest character sequence without repeating character

→ LL → middle element

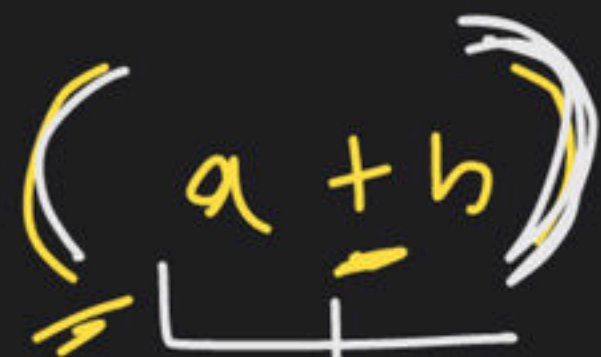
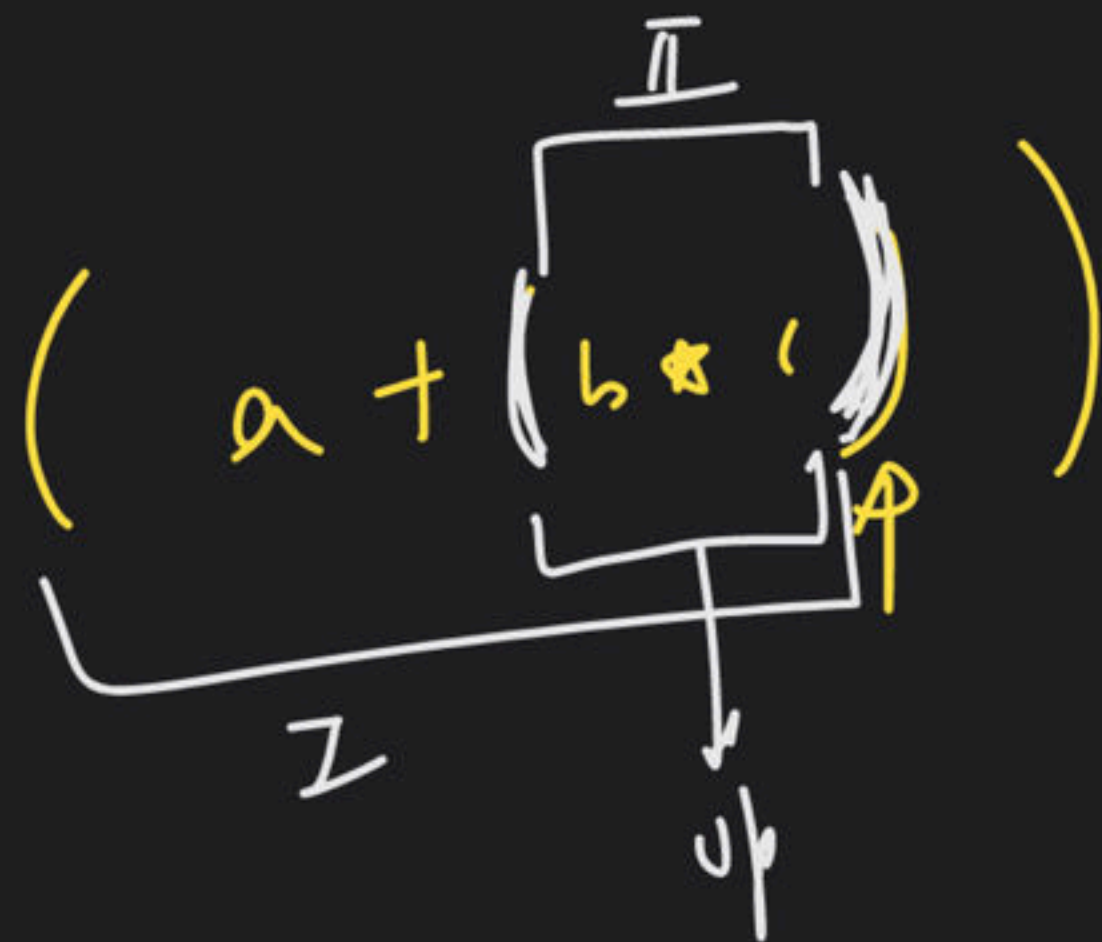
→ reverse string

→ LIS → (dp)

→ Search in a rotated array

→ 2 sum - / Three sum → array

→ i/p → exp →



Yes → Think

No → Redundant



$h \rightarrow x \rightarrow R_{-}$

Code

```
for (int i = 0; i < s.length(); i++)  
{  
    ch = s[i]  
    if (ch == '(' || ch == '[' || ch == '{')
```

```
        s.push(ch);
```

```
    else if (ch == ')')
```

```
    {  
        while (s.top() != '(')
```

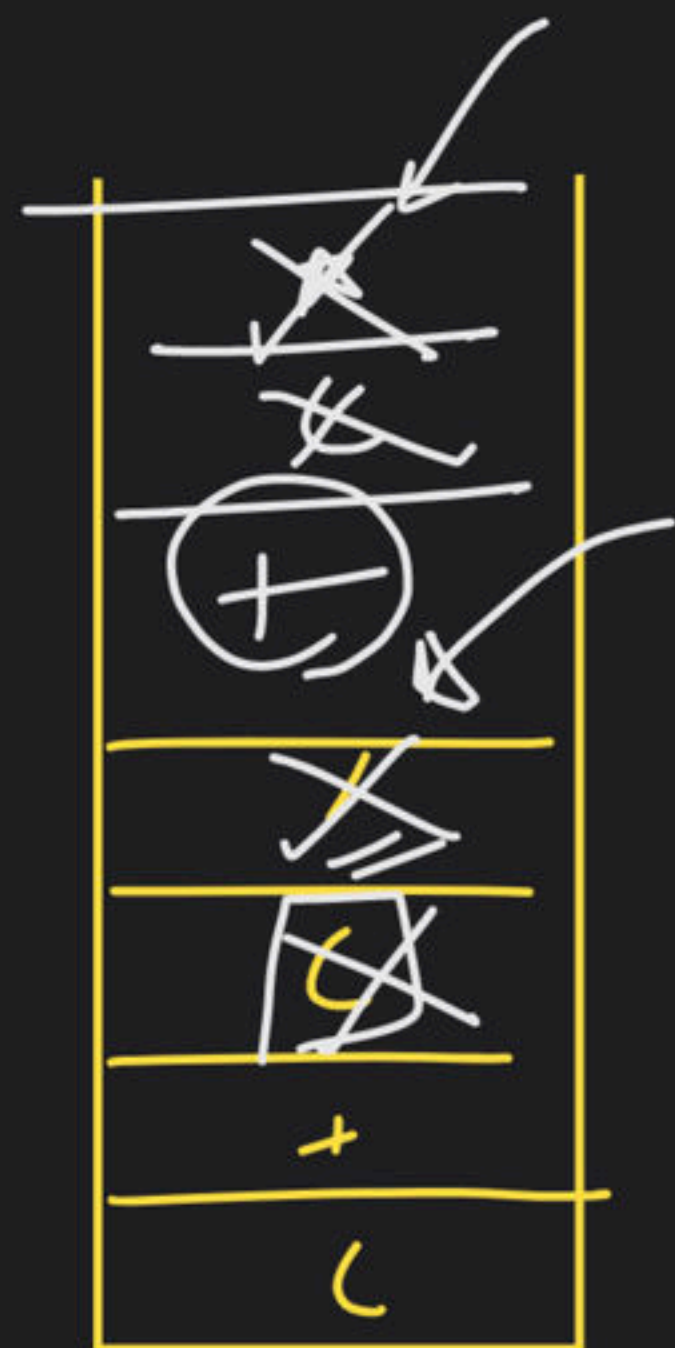
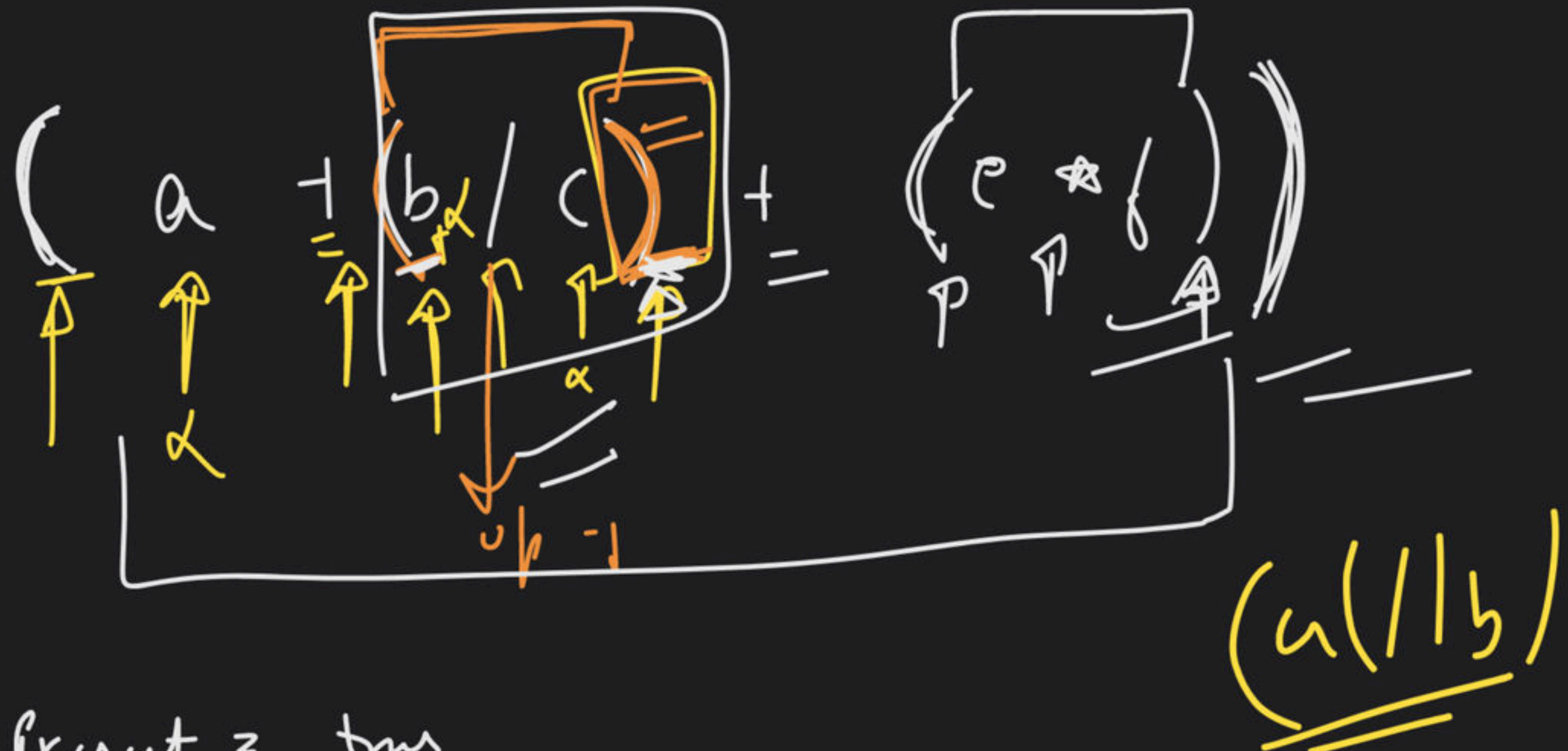
```
        {  
            // cancel operation  
            s.pop();
```

```
        }
```


```
    }
```

```
}
```


exp \rightarrow



Redundant
Brackets



$(N_0) \rightarrow (R \approx B)$

$$\text{while } (\text{!top}()) = '()'$$

```

    }
    if (isR == true)
        return true
    }
    → pop

```


$$(a \mid b)$$

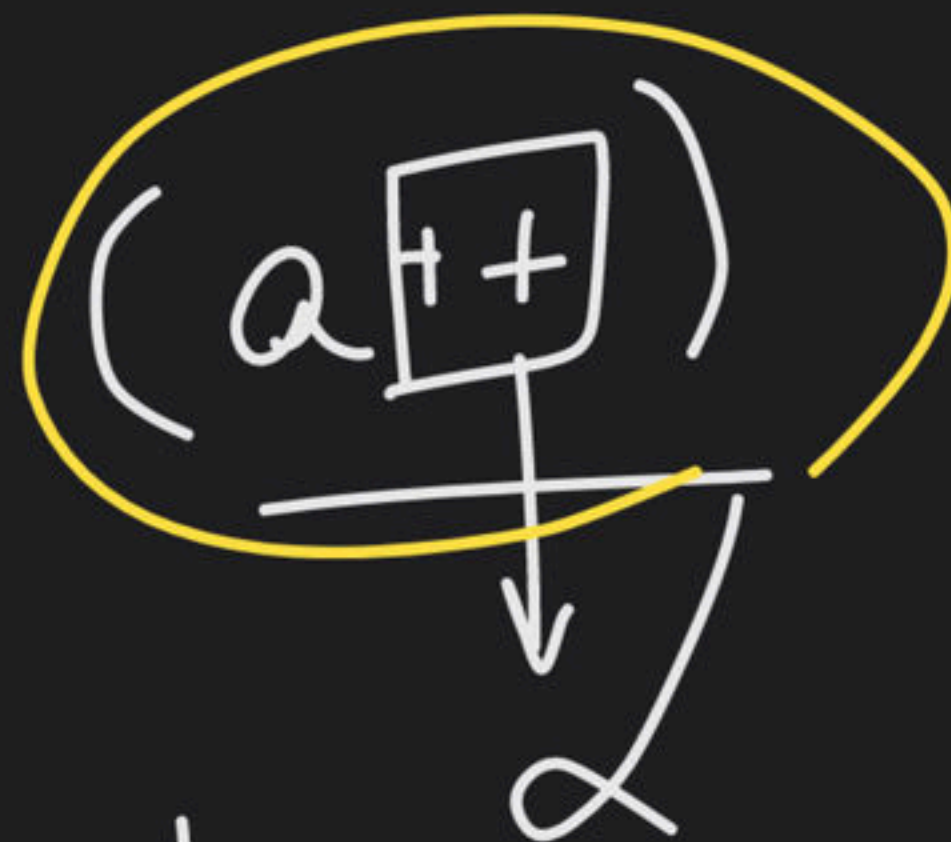
Valid
Exp



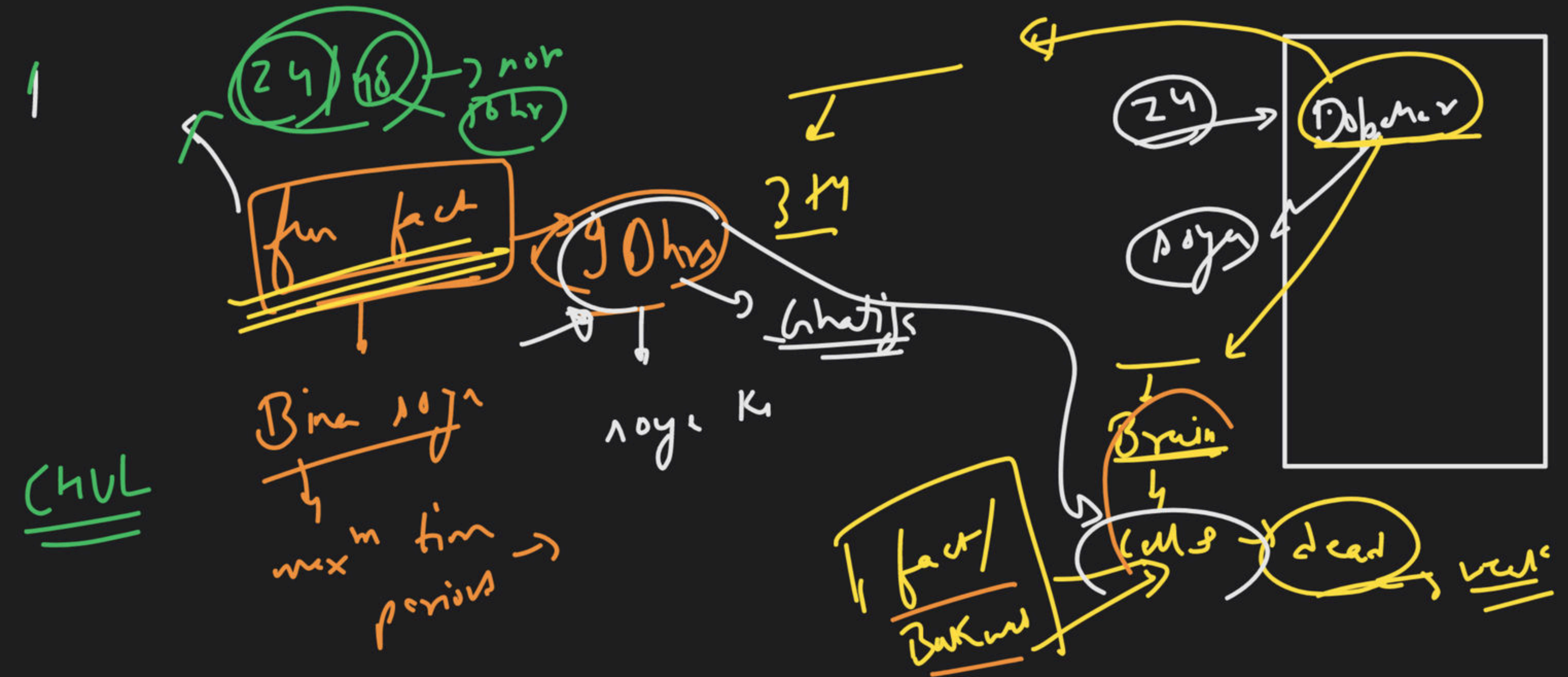




) b a /)

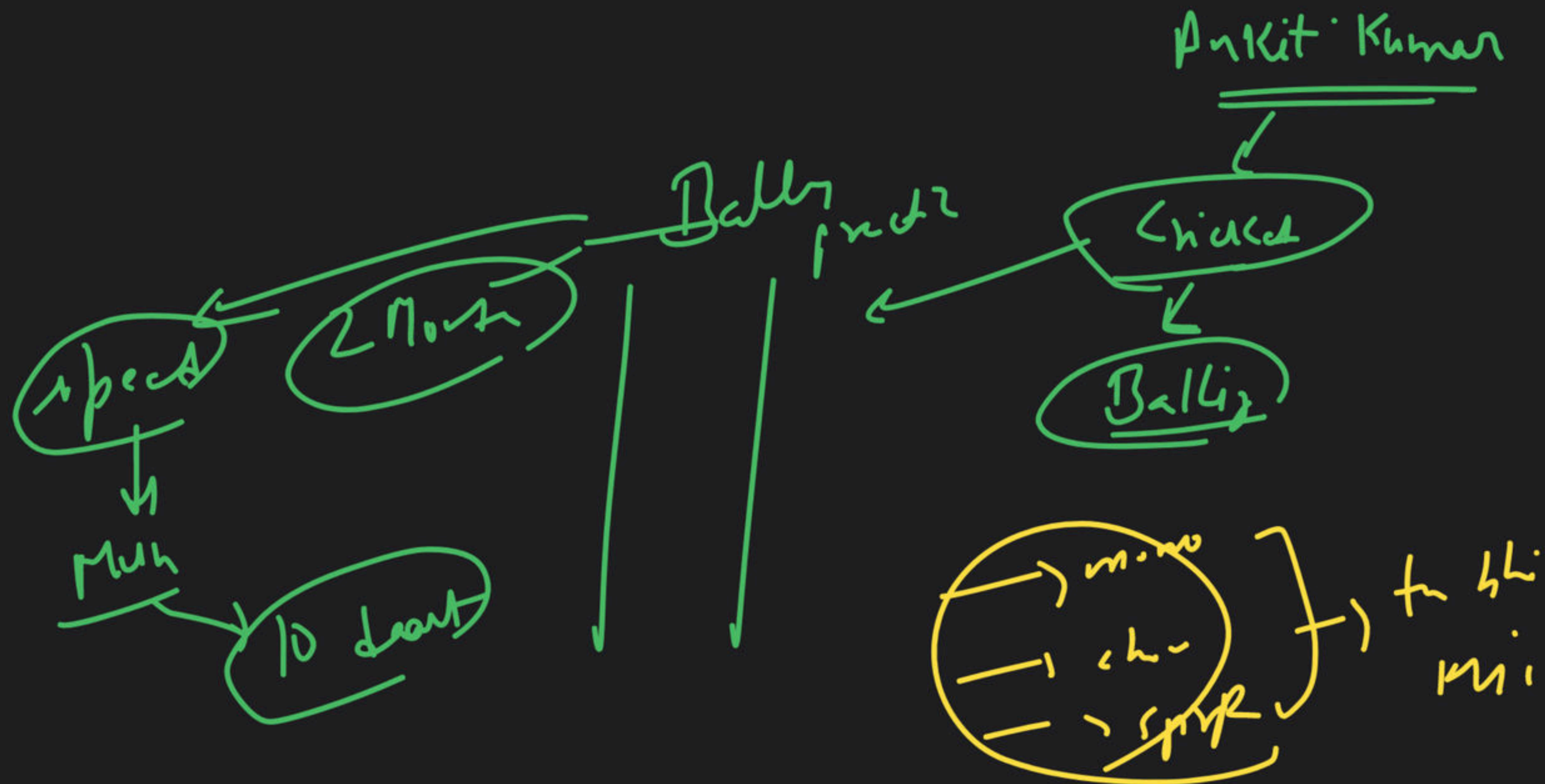


Minimum bracket Reversal → (H/W)



S.A

↳ gym - 1 1 m. 1



i/p \rightarrow string \rightarrow exp = "} {" \rightarrow { } \rightarrow 2

"{ { { { " \rightarrow { { } } \rightarrow 2

"{ { { } } { " \rightarrow { { { } } } \rightarrow 1

Valid & Leaky
not possible

{ - }

{ - }

⊖

Q →

Next, smaller Element →

Interviews
fav ques

i/p →

{ 2, 1, 4, 3 }

o/p →

{ 1, -1, 3, -1 } →

[1, -1, 3, -1] → ans

//

{ 2, 1, 4, 3 }

T.C. $\rightarrow O(n^2)$

Approach

Brute force

```

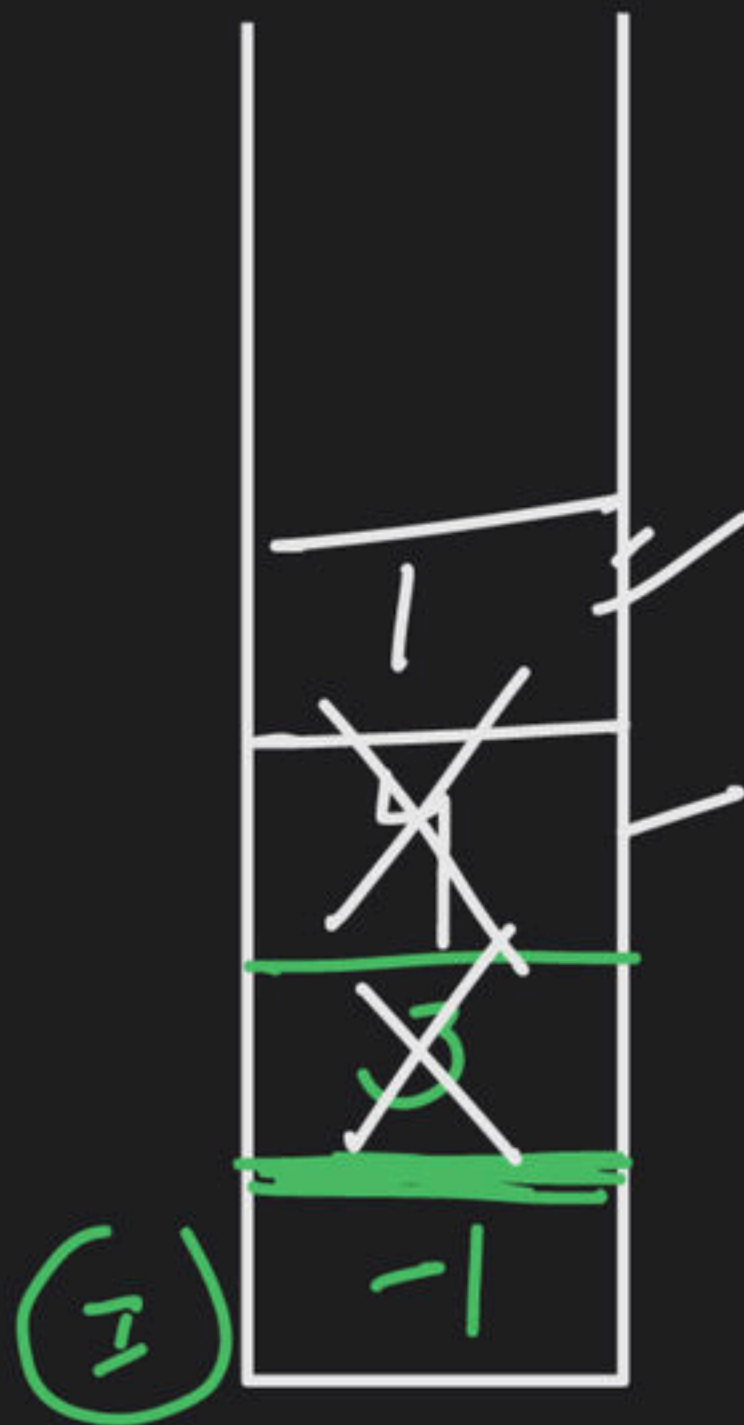
for (i  $\rightarrow$  0; i < n; i++)
{
    for (j  $\rightarrow$  i+1; j < n; j++)
    {
        // Comparison logic
    }
}
    
```

2 Stack

{ 2, 1, 4, 3 }

↑ ↑ ↑ ↑

item



$s.top() < item$, an

$s.top() == -1$ ←

$1 > 3$ $2 > 1$

[1, -1, 3, -1]

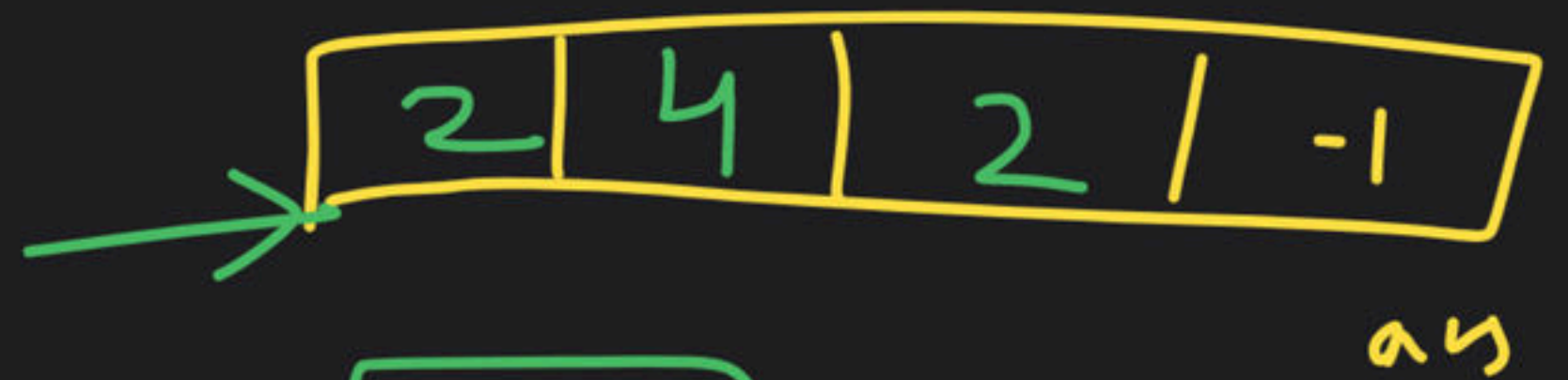


{ 3, 5, 4, 2 }

↑ ↑ ↑ ↑
 item item item

easy-
 push

T <



5 < 3

4 < 5

4 < 3

is.top() < item, an

2 < 3

2 < 4

4
3
2
1
-

24

A hand-drawn diagram of a 5x5 grid. The top-left 3x3 sub-grid contains the numbers 1, 2, 3, 4, 5 in its first row. The rest of the grid is empty. A large bracket on the right side of the grid is labeled with a circled '5'.

$$\begin{array}{r} h + h \\ \hline 2h \end{array}$$

h

{ 6 4 3 2 1 }
 7 7 7 4 4

Code!

H/W

H/W

Ques

ans[]

stack<int> s

s.push(-1);

for (int i = n-1; i >= 0; i--);

{

int item = arr[i];

while (s.top() != -1 or s.top() >= item)

{

pop

3



→ push arr[i]

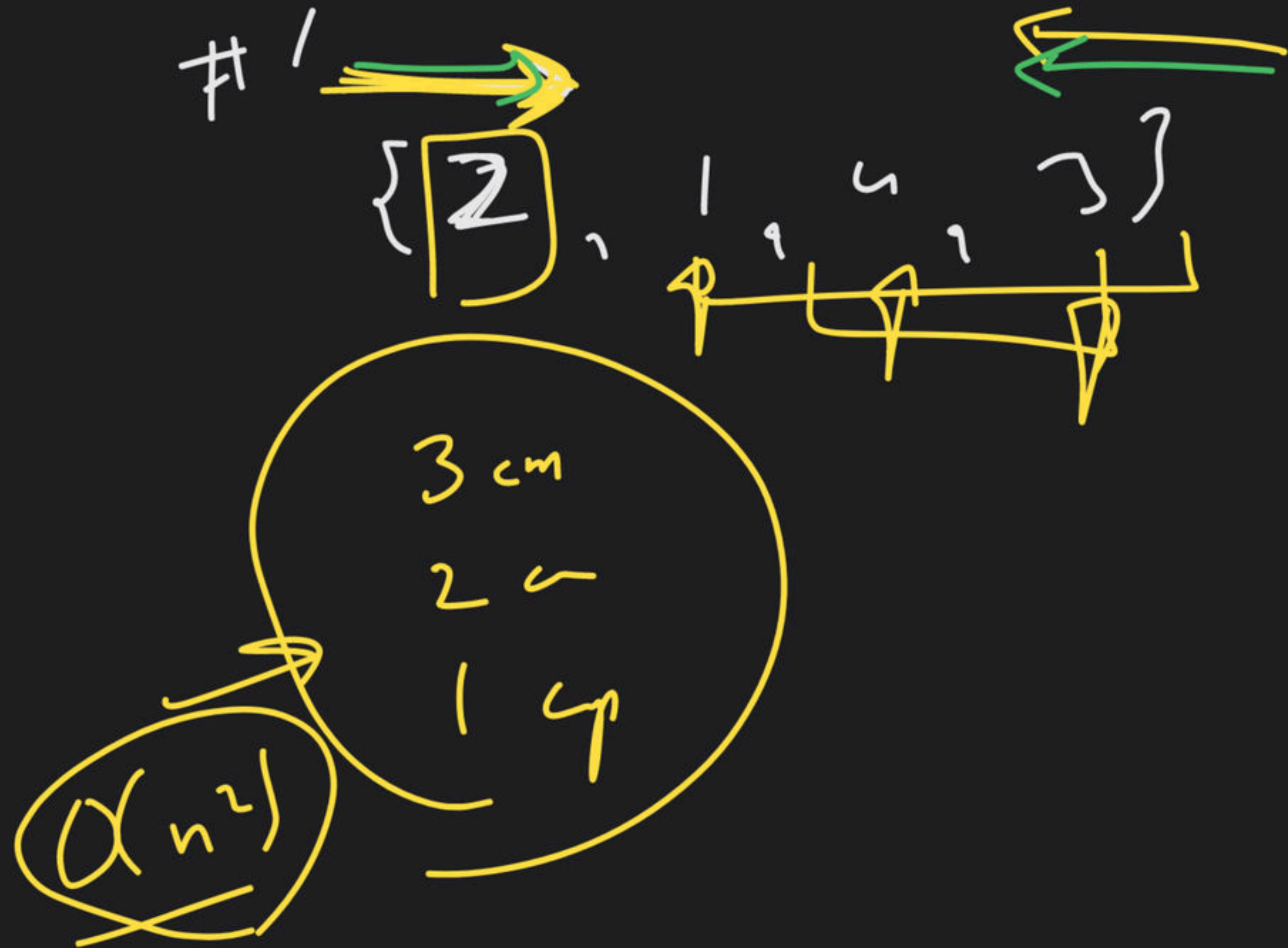


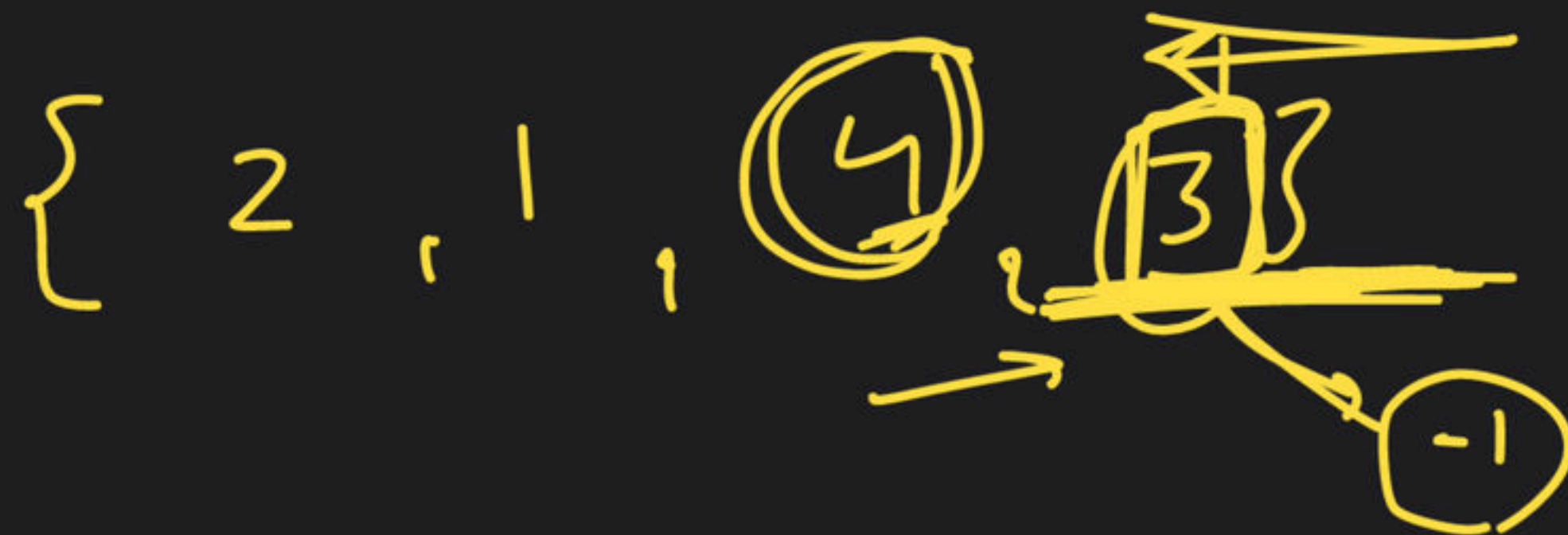
→ push curr element

ans

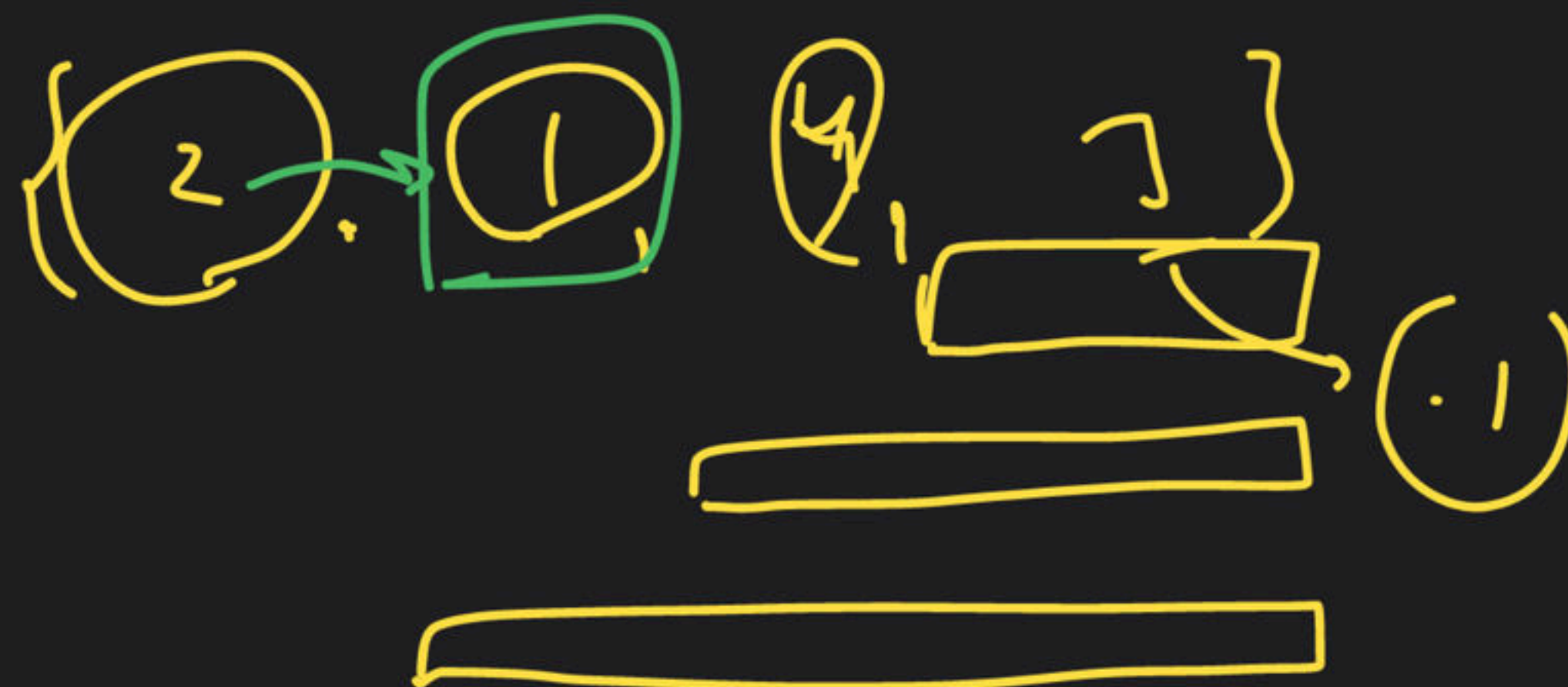
→ Largest Rectangular area in hist¹

→ celebrity Problem





LIFO



/ prev smaller element shy
) H/W 1.5
 ↓ α
 2

→
 (doubt) 15-30 min
 ↙ Dipe Knih
 Lect.

→ $(a_{-1}^T) + (b_{-2}^T)$

u
 c
 ↘ $(\frac{5}{2} + \frac{6}{2} + 1)$