



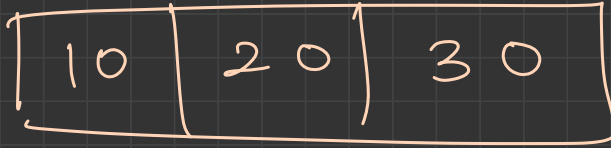
# # implementation #

⇒ Stack ← vector

30  
20  
10

top()   
 ↙   
 top - most element   
 of stack

arr



LIFO

size C

LIFO

pop

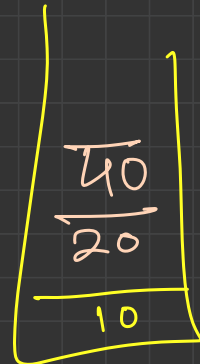
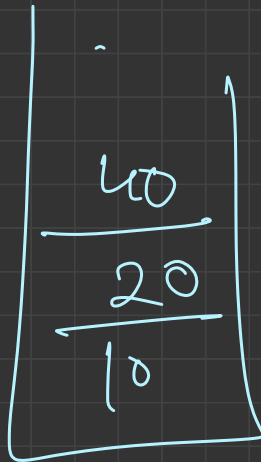
30

30

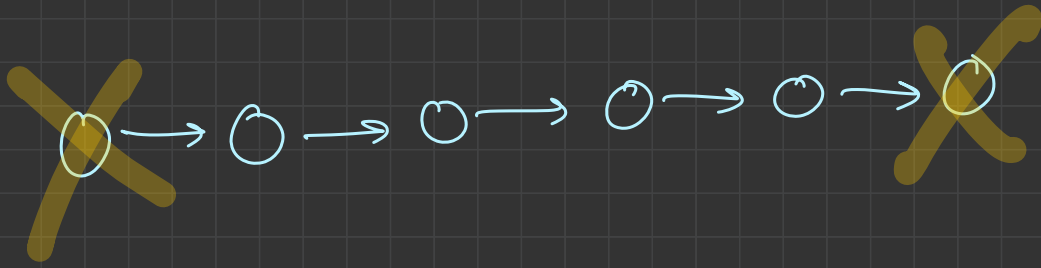
pop-back

30

30



40



1 element  
1 to delete

most optimized

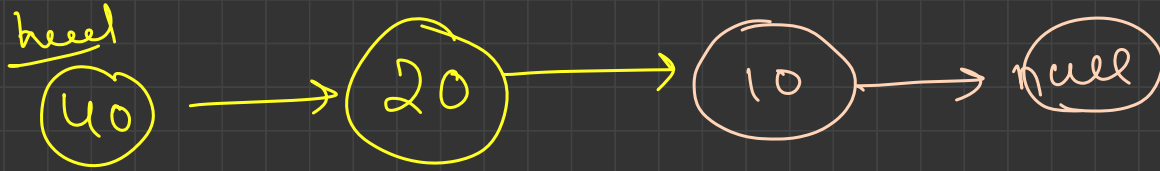
top

pop()

first val

linked list 1st  
starting me

`pop()`  
`push()` = clear  
`pop()`



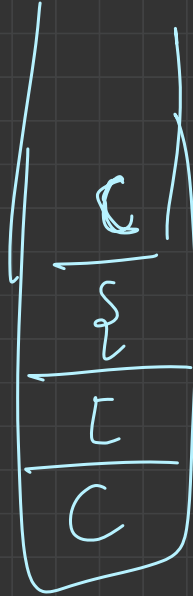
example :

" ( [ { ( ) { } { } ] ) "

( [ { ( ) { } { } ] )

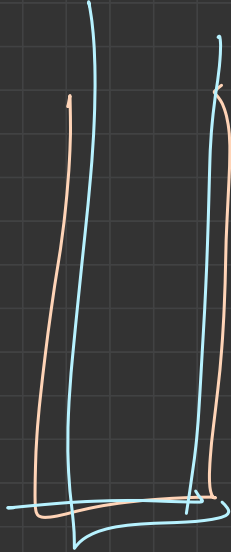
~~↑~~ ~~↑~~ ~~↑~~ ~~↑~~ ~~↑~~

Stack



's' '}'  
→  
else opening match

Full - Case :-



" [ ] " "

[ ]

" [ ] { } " "

( )

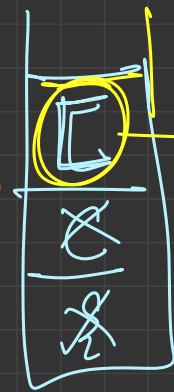
" { } [ ] " "





match

empty



X

un  
matched  
the  
guy

next greater element

# monotonic stack #

#

index:-

0	1	2	3	4	5	6
1	4	3	2	10	4	7

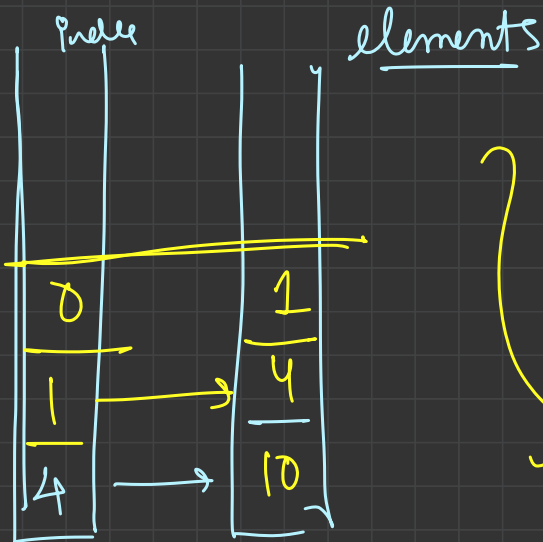
10 ke liye 4 tak  
ans nahi ho  
Skta

nge\_index:-

1	4	4	4	7	6	7
---	---	---	---	---	---	---

ans

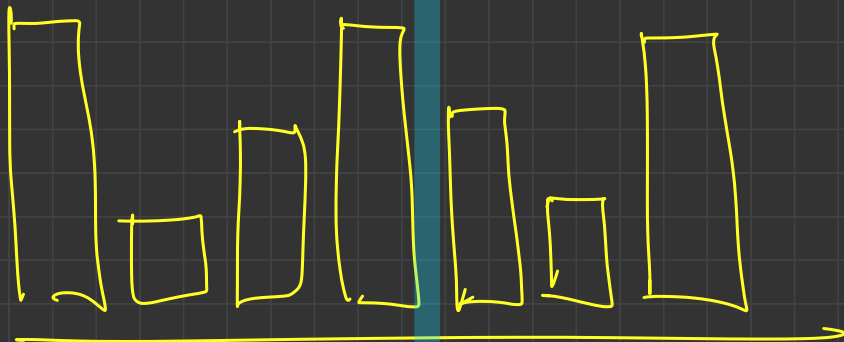
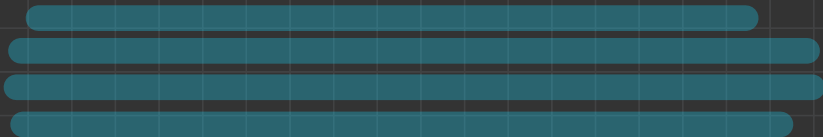
1	4	4	4	7	6	7
---	---	---	---	---	---	---



①

monotonic  
↑ tree

Build

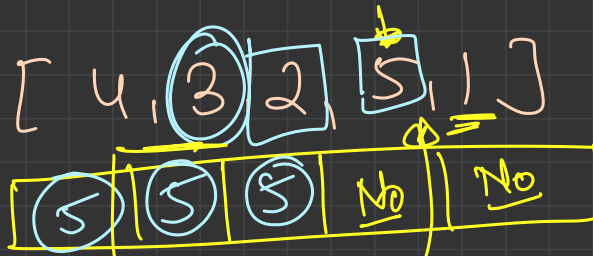


Algorithm

⊕ jab tum ek index par aaye ho  
Sabse phle sare chote elements  
hata do. ---->

⊕ → stack ekhali na (index)  
[ stack.top() ]

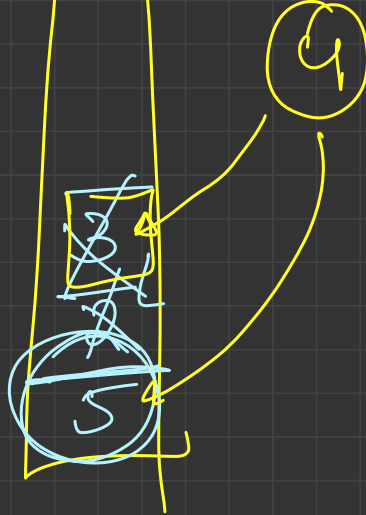
⊕ khud ko insert kar lenge

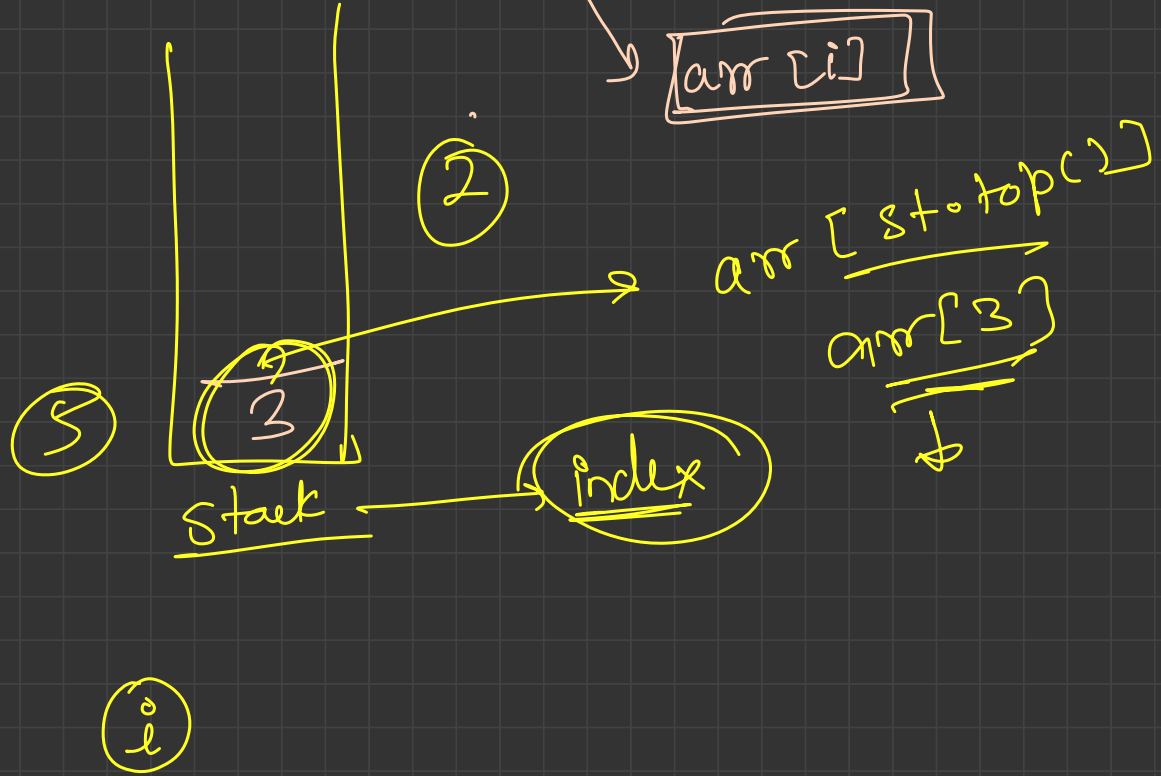
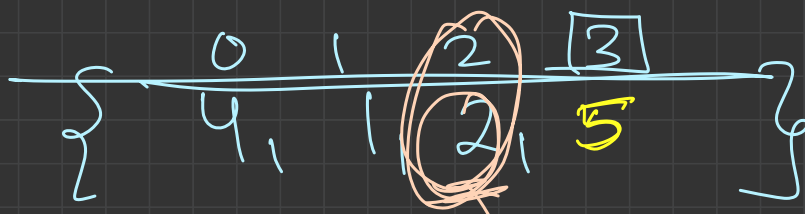


galdi batao  $O(N)$

$\approx O(N)^2$

fun





0 1 2 3 4  
[ 2, 1, 4, 3, 2 ]  
nge [ 2, x, x, x ]

1

$\text{nge}[i] = \text{st.top}();$

4

1  
2

$\{ \overset{0}{\underline{4}}, \overset{1}{\underline{3}}, \overset{2}{\underline{1}}, \underline{4}, \underline{5} \}$



→ output

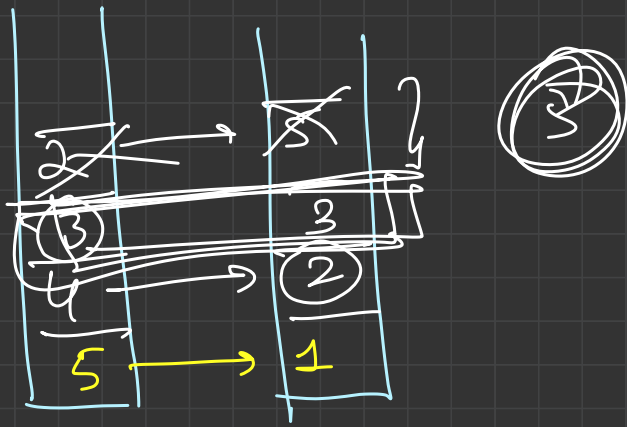
$\{ \overset{0}{2}, \overset{1}{\textcircled{4}}, \overset{2}{5}, \overset{3}{3}, \overset{4}{2}, \overset{5}{\underline{1}} \}$

4  $\textcircled{5}$  6

next smallest







{ # prev-greatest element }  
 { # prev-smallest element }

0 1 2 3 4 5 6 7  
 4, 3, 2, 5, 1, 4, 3, 2

prev-greatest | -1 | 0 | 1 | -1 | 3 | 3 | 5 | 6 |

prev-  
smallest

-1	-1	-1	2	-1	4	4	4	1
----	----	----	---	----	---	---	---	---

prev-smallest arr

0	1	2	3	4	5	6
<u>100</u>	<u>80</u>	<u>60</u>	<u>70</u>	<u>60</u>	<u>75</u>	<u>85</u>

array:-

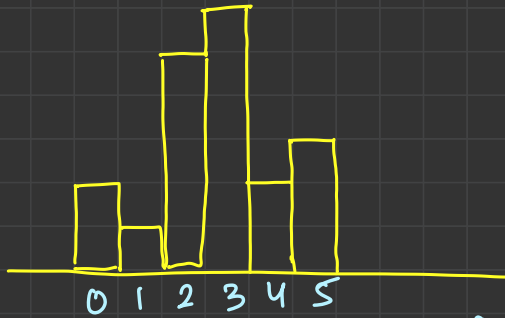
1	1	1	2	1	4	6
---	---	---	---	---	---	---

$$(6-0)$$

$$(4-3)$$

$$(5-1)$$

4



Area  $\Rightarrow$  maximum

$$\text{width} = \underline{\underline{(j - i) + 1}}$$

ek taska

Brute force

```
int ans = 0;
```

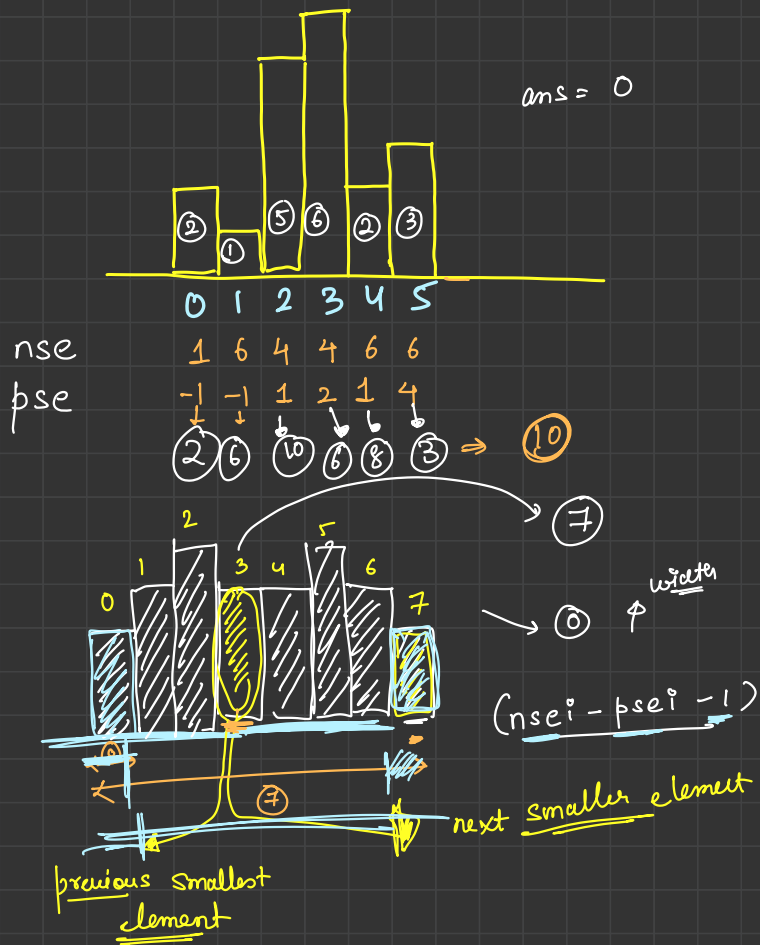
```
for (int i = 0; i < n; i++) {
```

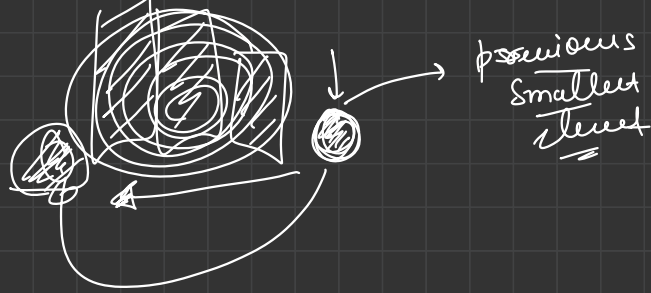
```
    for (int j = i; j < n; j++) {
```

```
    }
```

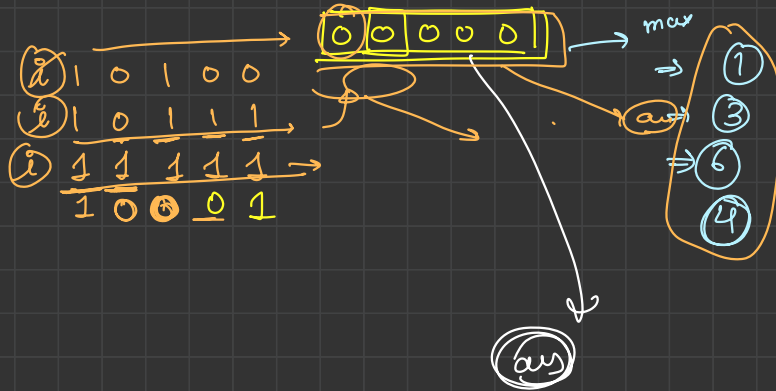
minimum

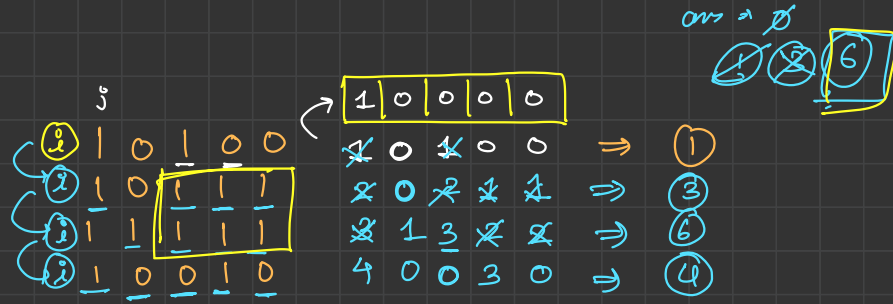
min  $\Rightarrow$  ~~1~~ 1





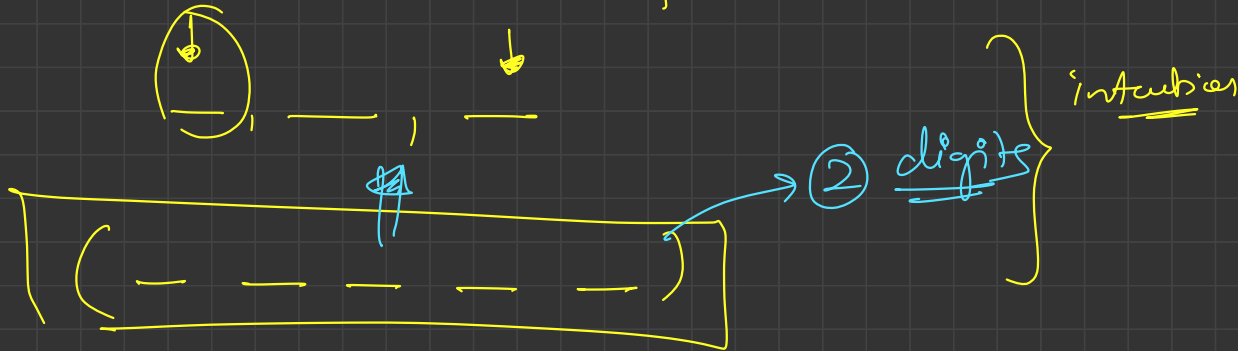
# Maximum Rectangle in a histogram #





# Remove k Digits #

k = 2



1 4 3 2 2 1 9

K = 3

~~2~~

~~1~~

0

{ 1 | 2 | ~~3~~ | 9 }

~~3~~

9

↓

9 | 2 | 1

↓

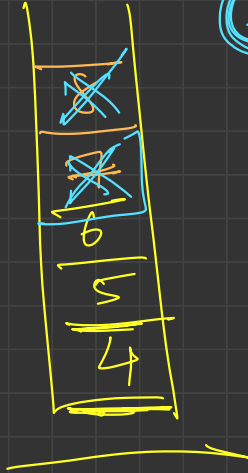
1 2 | 9

→ Smallest no

"4 5 6 7 8 3"

3

$k=2$



$k=2$

khataun

$k=2$

13



⇒

1 4 5 3 8 5 7

~~7~~  
5  
~~8~~  
3  
~~5~~  
~~7~~  
1

7

~~8~~

5

3

~~8~~

~~7~~

1

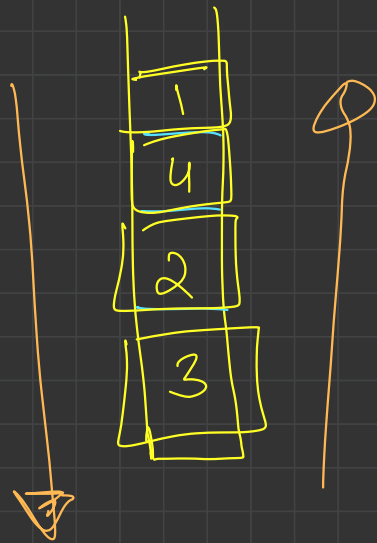
3

3 5

k=4

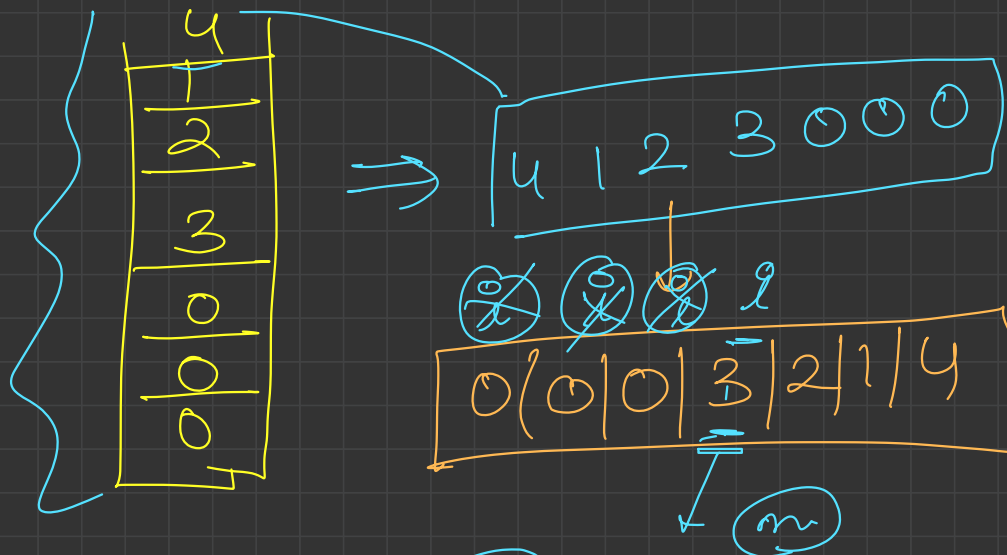
k=2

1



$\Rightarrow$  3 2 4 1


A horizontal arrow pointing from the number 2 to the number 4 in the sequence 3 2 4 1.



flag = ~~false~~ true

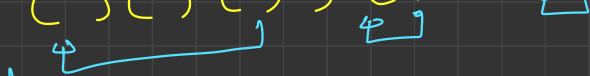
print

"( ( ) ( ( ( ( ( ( "



2<sup>nd</sup> time

"( ) ( ) ( ) ) ( ) ( ( ) )"



Valid  
parentheses

-1 0 1 2 3 4 5 6 7 8 9

"( ) ( ( ) ) ) ) ( )"

         ↑

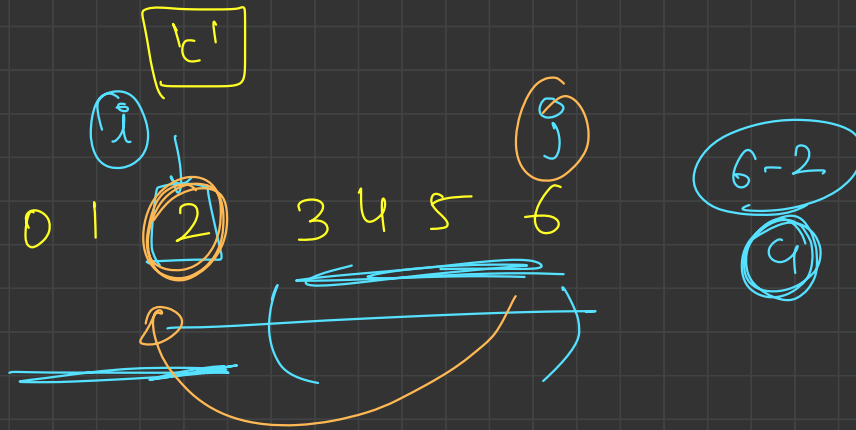
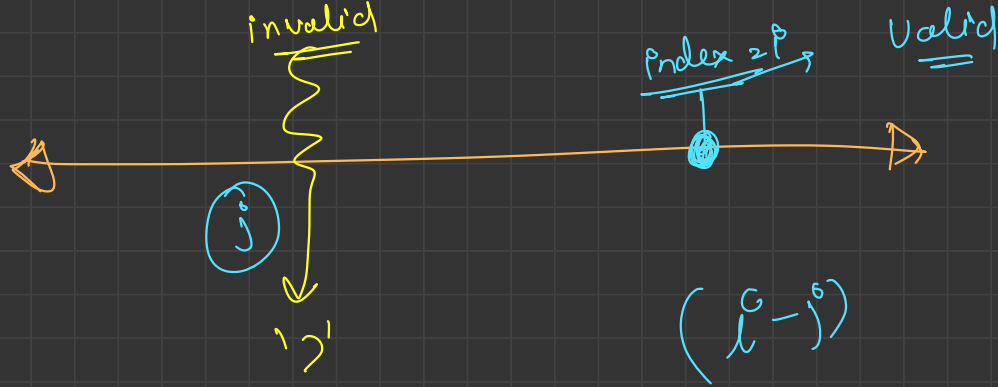
(6)

(2)

i - start to pop()

-1

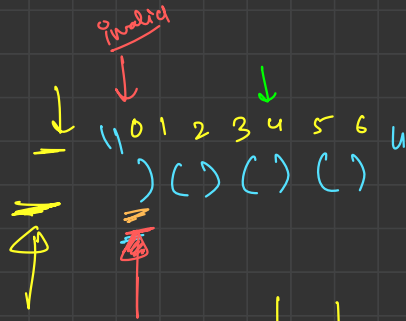
→ invalid index



$i - \text{st. to } p()$

stop - ( Invalid index kasa bhai )

kadam  
kasano  
hu



du

2-0

2

~~2~~

pop

(4-0) => 4

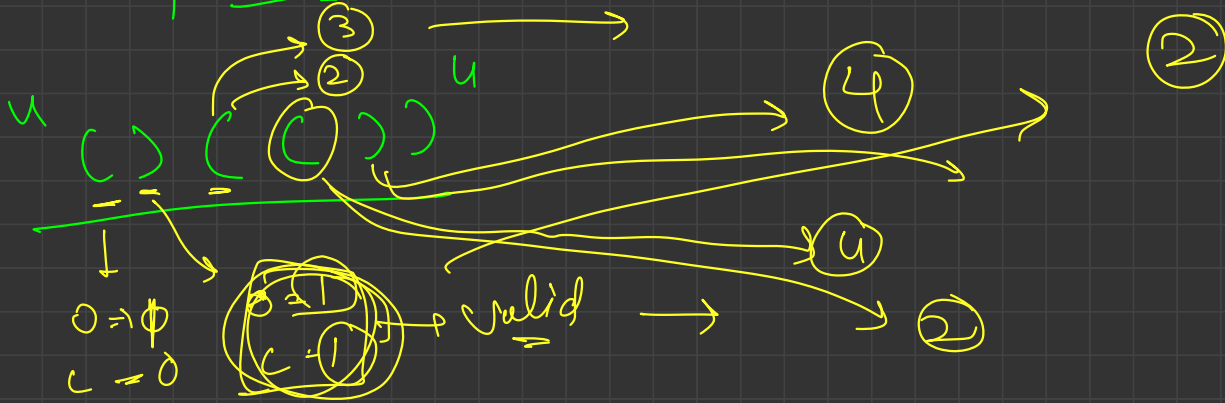
invalid hogc

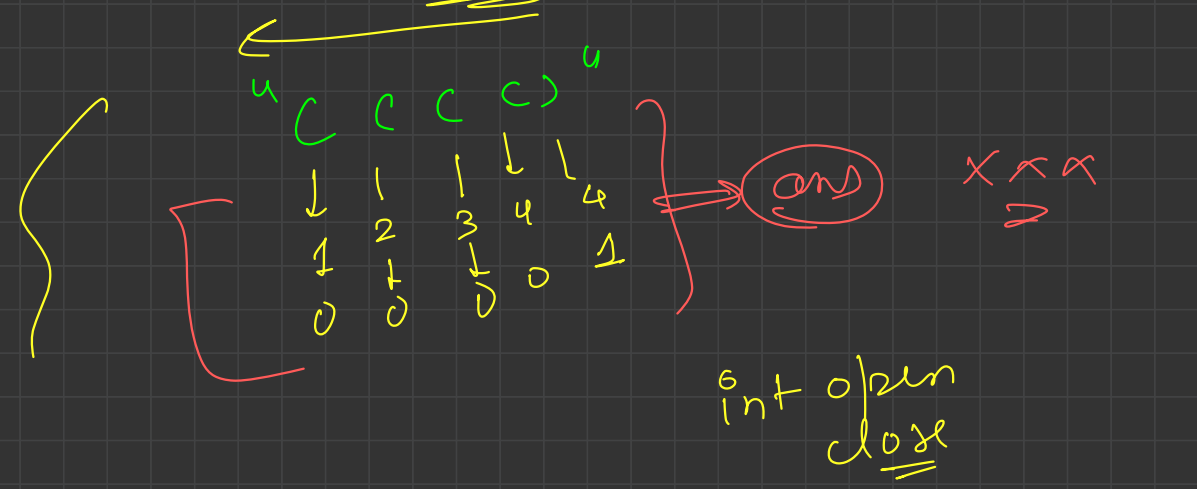
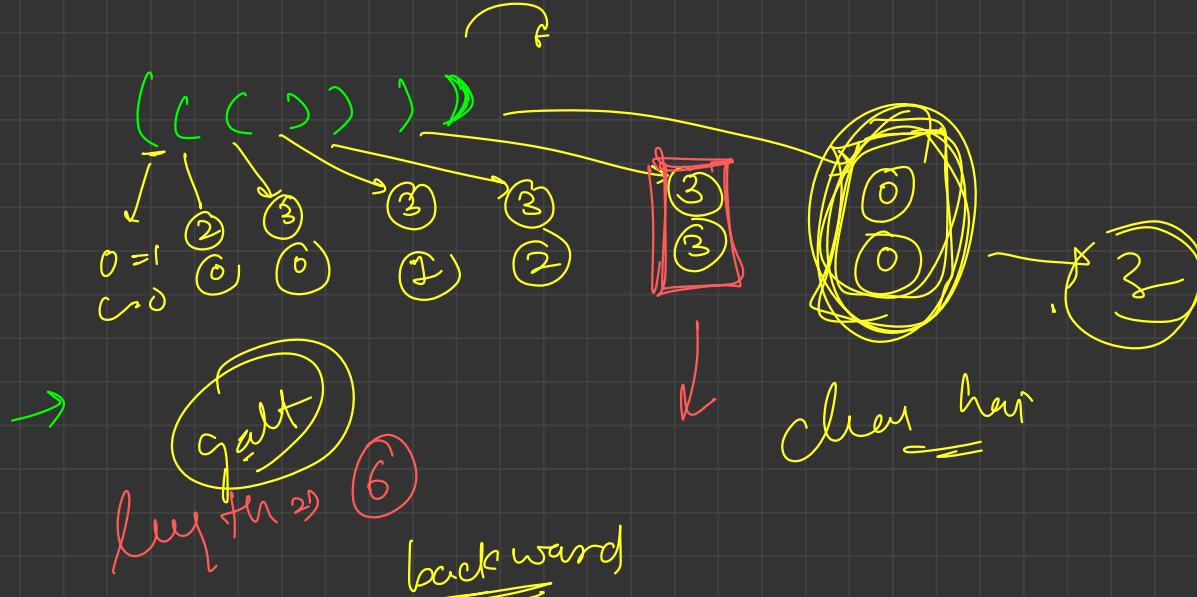
1  
101  
101



$^u [ ( ( ( ) ( ( ^u \rightarrow \textcircled{2}$

valid  
open = closing same







{

