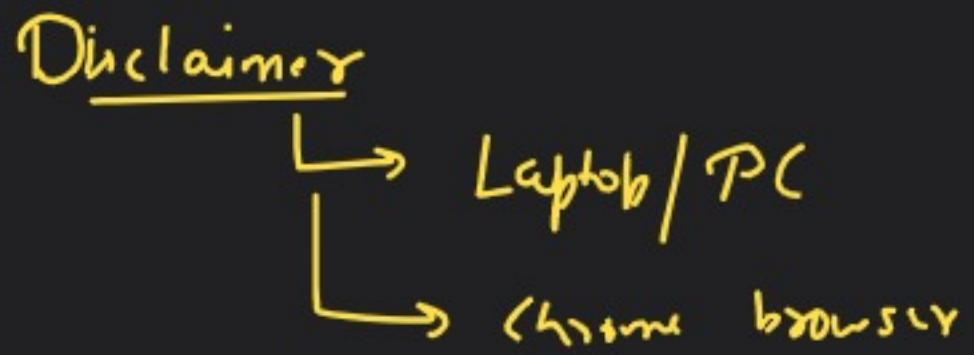


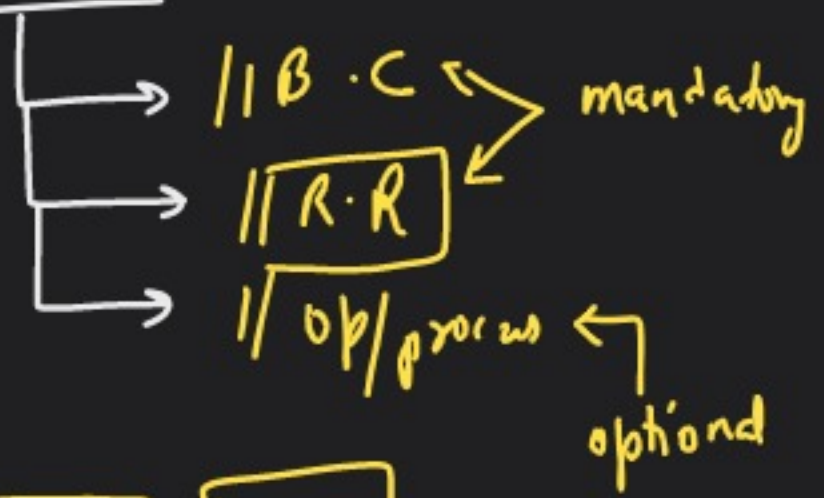
Recursion on Arrays & Strings and Doubt Clearing | LIVE

Special class

Vishal Pachauri • Aug 27, 2022

Recursion - Arrays & Strings

J 51 class → what is Recursion?



→ print Inc

→ print Dec

→ print Dec & Inc

→ print Inc & Dec → H/w

H/w → Head Tail

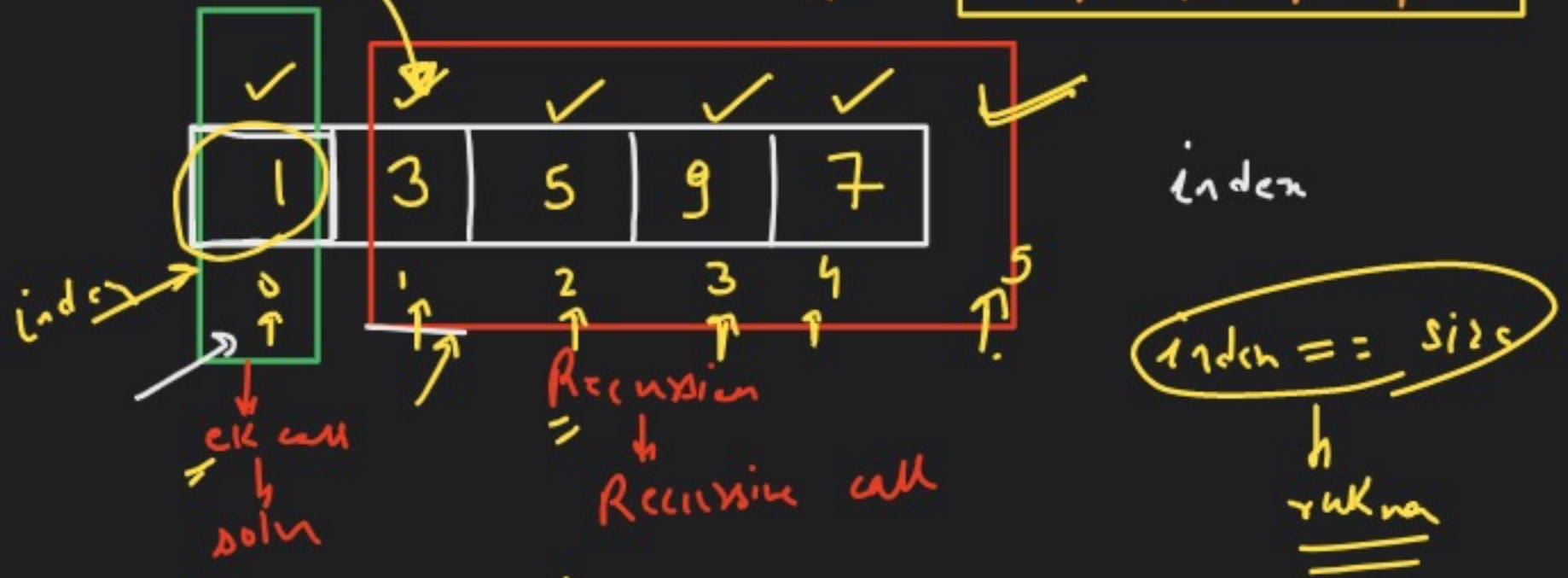
→ array →

→ Recursion on Arrays:-

① Print array

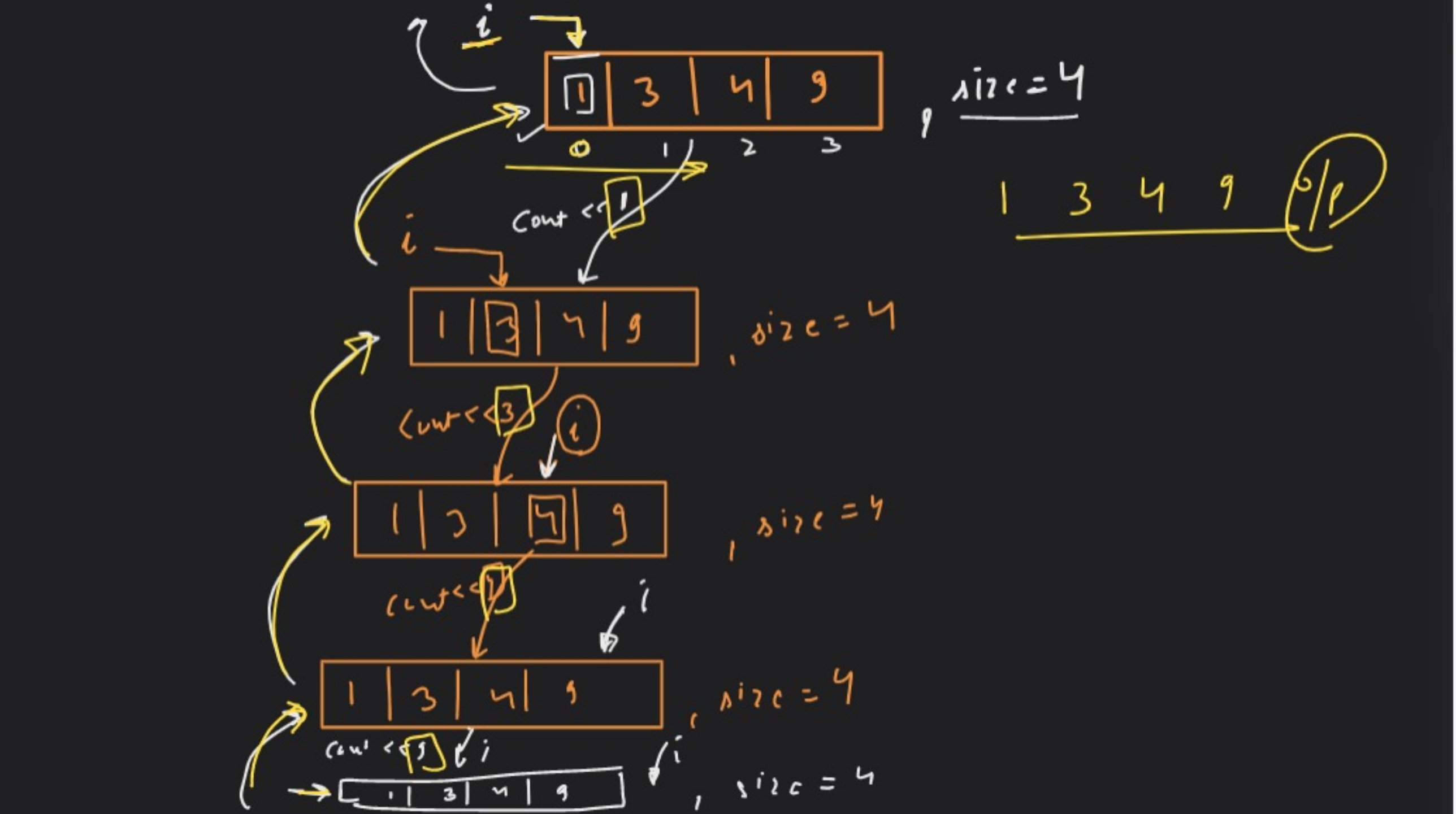
i/p → arr → {1, 3, 5, 9, 7}

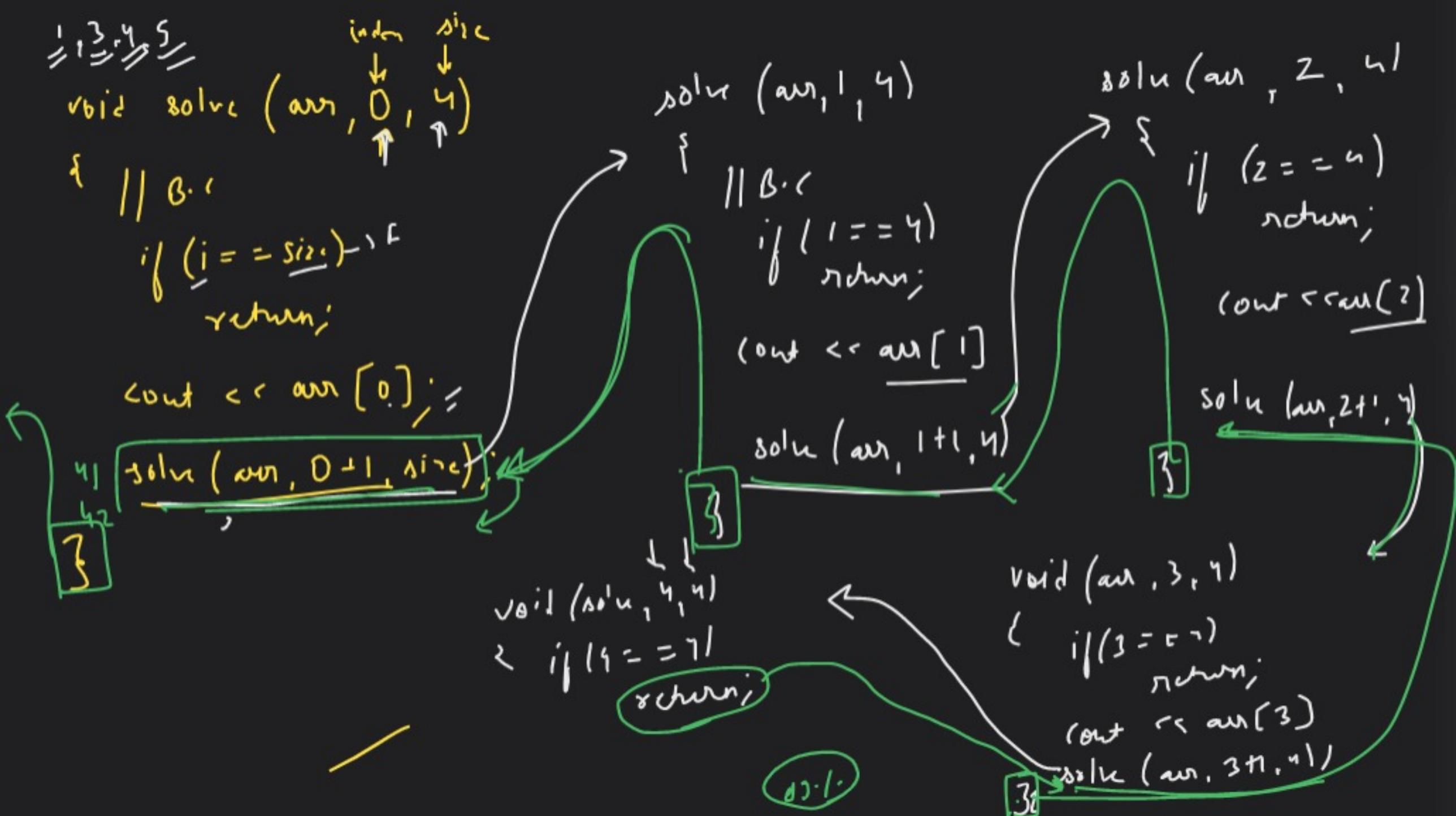
o/p → 1, 3, 5, 9, 7



(arr, 0) ⟶ arr, 1

```
void solve (arr, index, size)
{
    // Base case
    if (index == size)
        return;

    cout << arr[index];

    solve(arr, index+1, size);
}
```


i
1 3 4 9
0 1 2 3
size = 4
1 3 4 9 o/p
cout << 1
i
1 3 4 9
size = 4
cout << 3
i
1 3 4 9
size = 4
cout << 4
i
1 3 4 9
size = 4
cout << 9
i
1 3 4 9
size = 4


1 3 4 5
index
size
void solve (arr, 0, 4)
{ // B.C
if (i == size) return;
cout << arr[0];
solve (arr, 0+1, size);
}
solve (arr, 1, 4)
{ // B.C
if (i == 4) return;
cout << arr[1]
solve (arr, 1+1, 4)
}
solve (arr, 2, 4)
{
if (2 == 4) return;
cout << arr[2]
solve (arr, 2+1, 4)
}
void (arr, 3, 4)
{ if (3 == 4) return;
cout << arr[3]
solve (arr, 3+1, 4)
}
void (solve, 4, 4)
{ if (4 == 4) return;

print (arr, size, size-1)

(arr+1)

→ Reverse print :-

++ index ⟷ index++

call by value

(1) index = 5

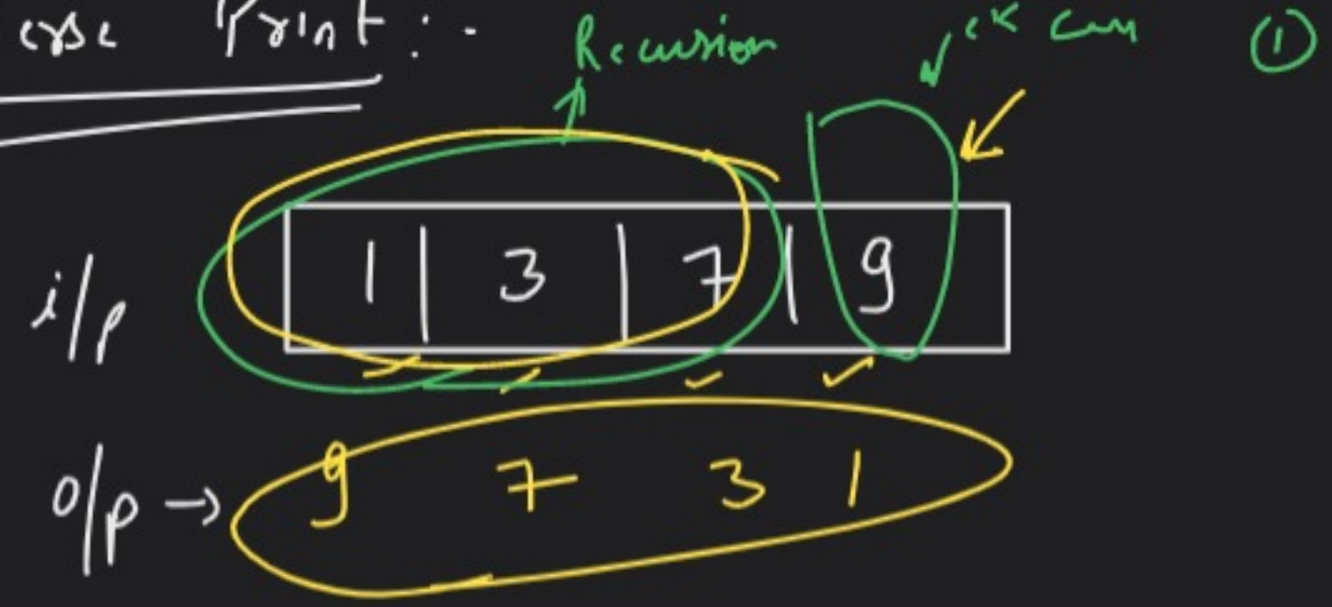
(2) int arg = index++

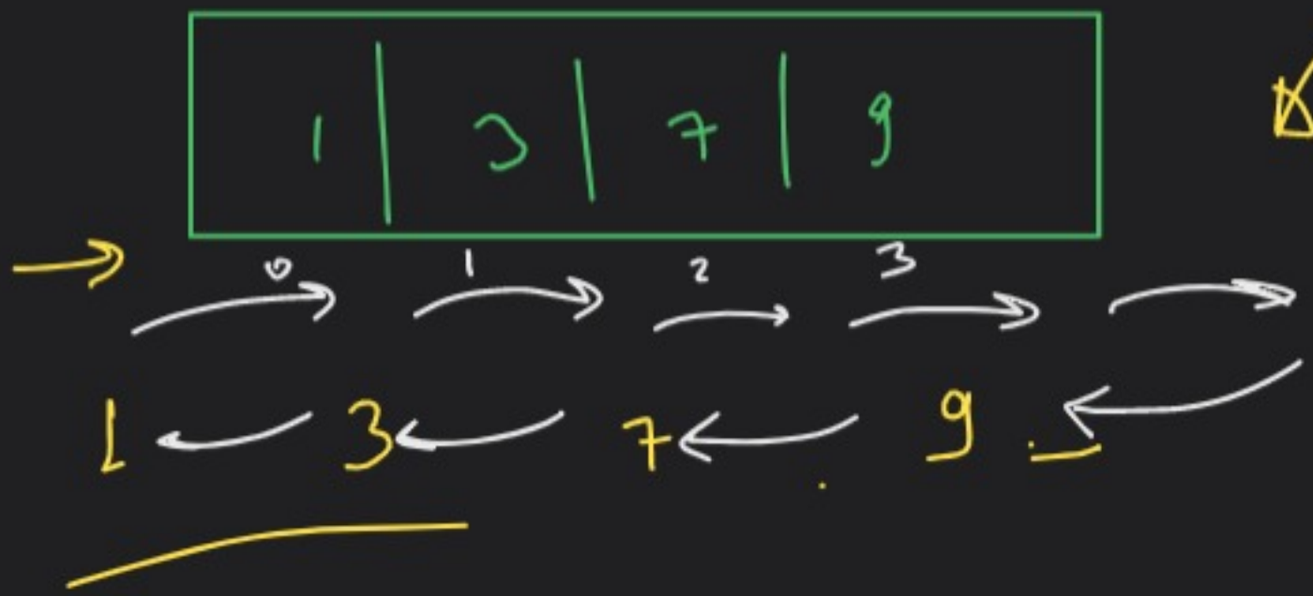
(3) cout << arg << index ;

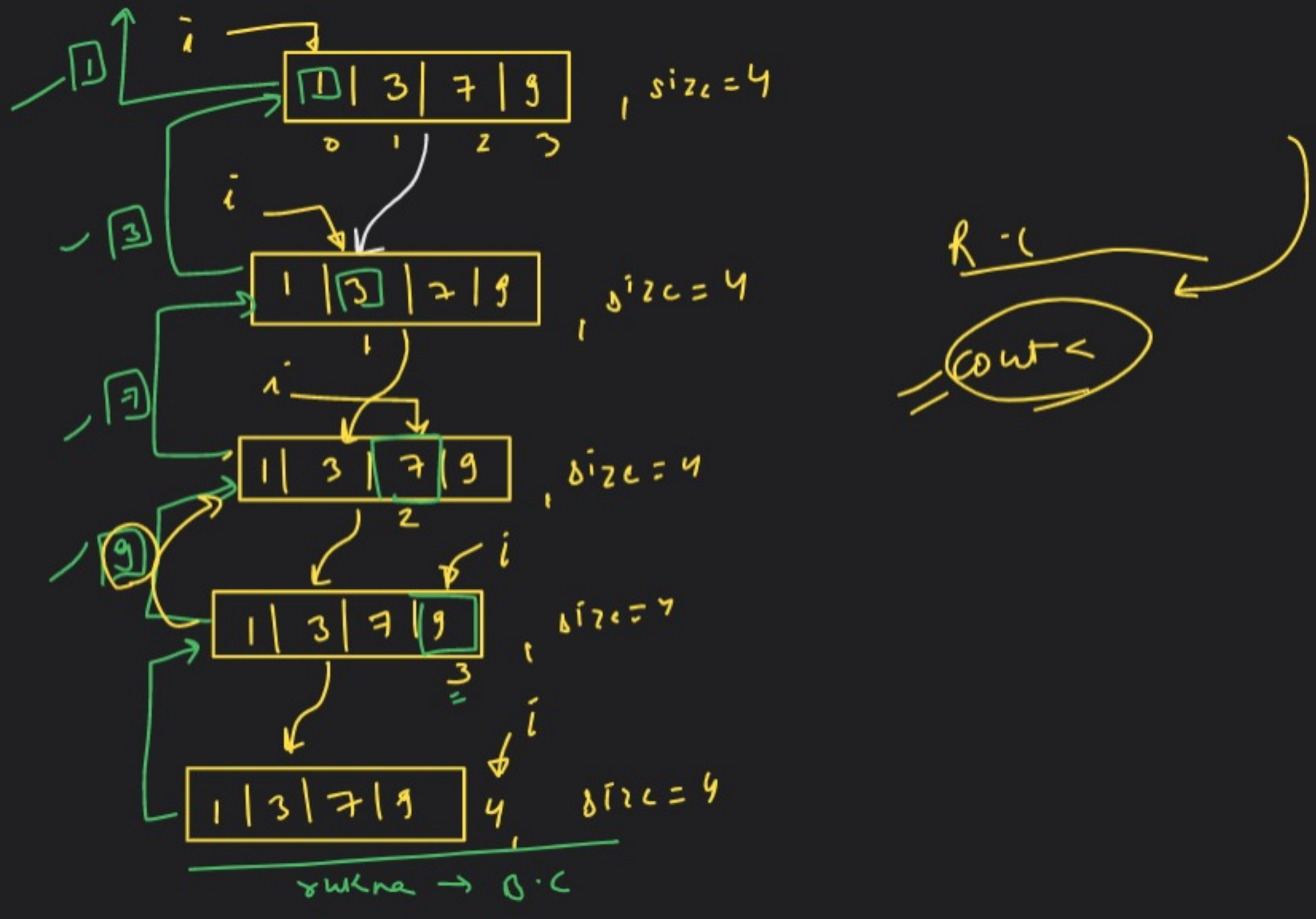
arg = 5

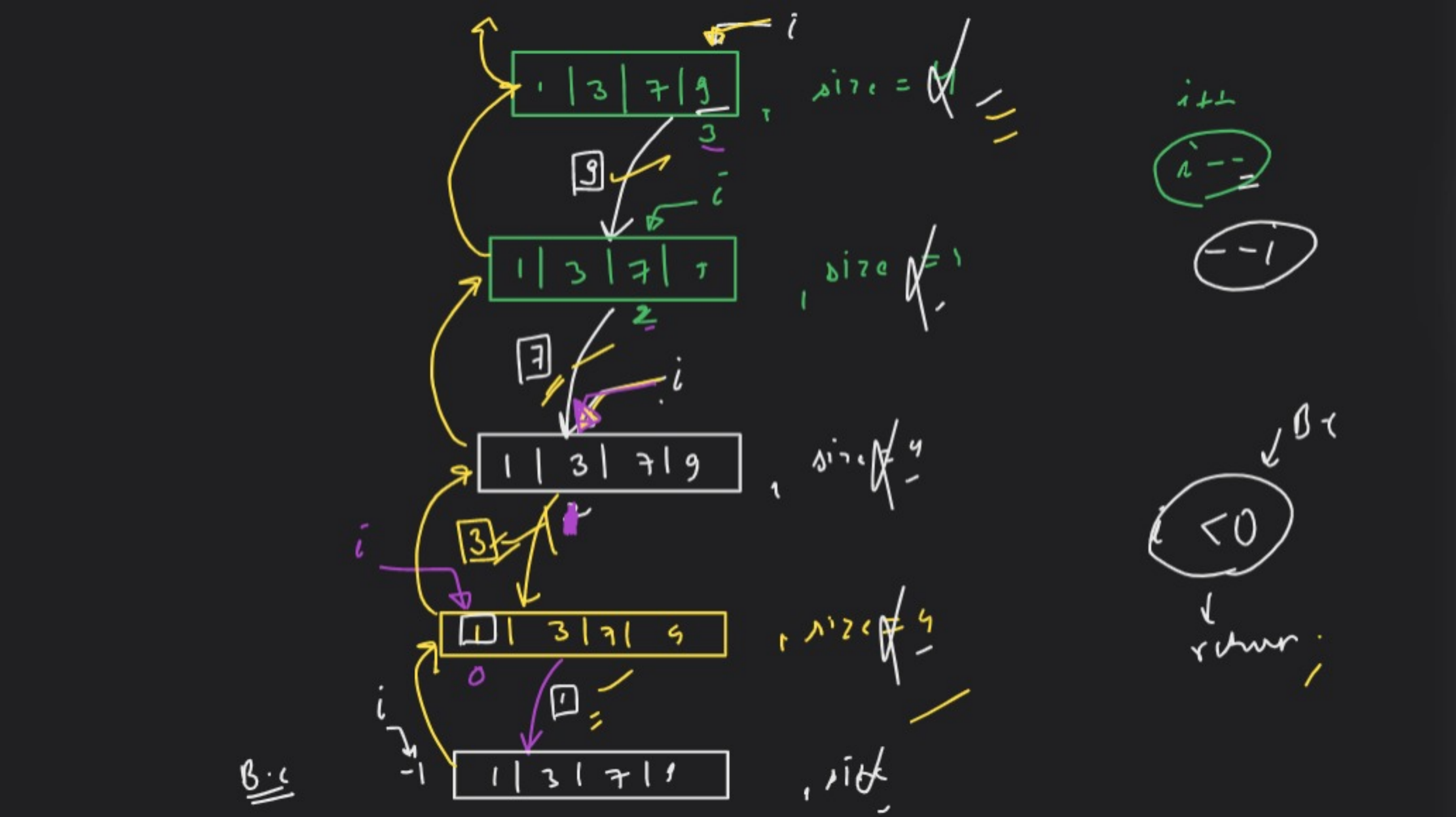
index = 6

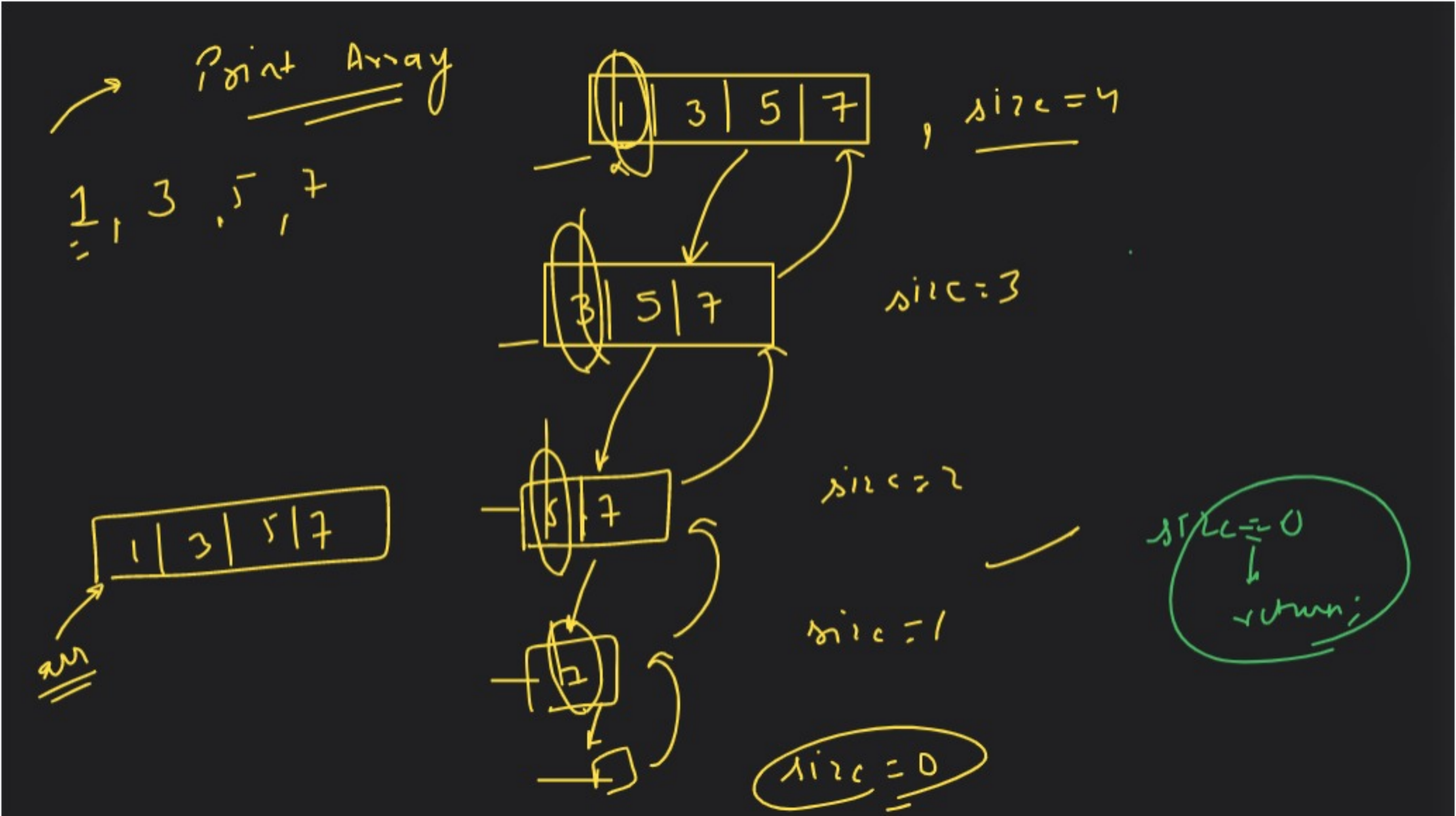
Reverse Print :-

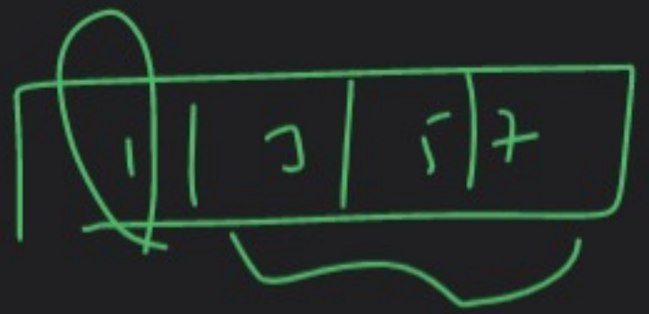





i
1
1 | 3 | 7 | 9
0 1 2 3
size = 4
i
3
1 | 3 | 7 | 9
1
size = 4
i
7
1 | 3 | 7 | 9
2
size = 4
9
i
1 | 3 | 7 | 9
3
size = 4
i
1 | 3 | 7 | 9
4
size = 4
powrót → 0.c
R-(
cout<


i
1 | 3 | 7 | 9
3
9
i
1 | 3 | 7 | 1
2
7
i
1 | 3 | 7 | 9
3
i
1 | 3 | 7 | 9
0
1
i
-1
1 | 3 | 7 | 1
i++
i--
-1
<0
return


Print Array
1, 3, 5, 7
1 | 3 | 5 | 7
size = 4
3 | 5 | 7
size = 3
5 | 7
size = 2
7
size = 1
size = 0
1 | 3 | 5 | 7
arr
size == 0
return;



T · R

op + R · C

T · R

R · C + op

int arr[] = {1, 3, 5, 7};

arr → base address of the array

Symbol table

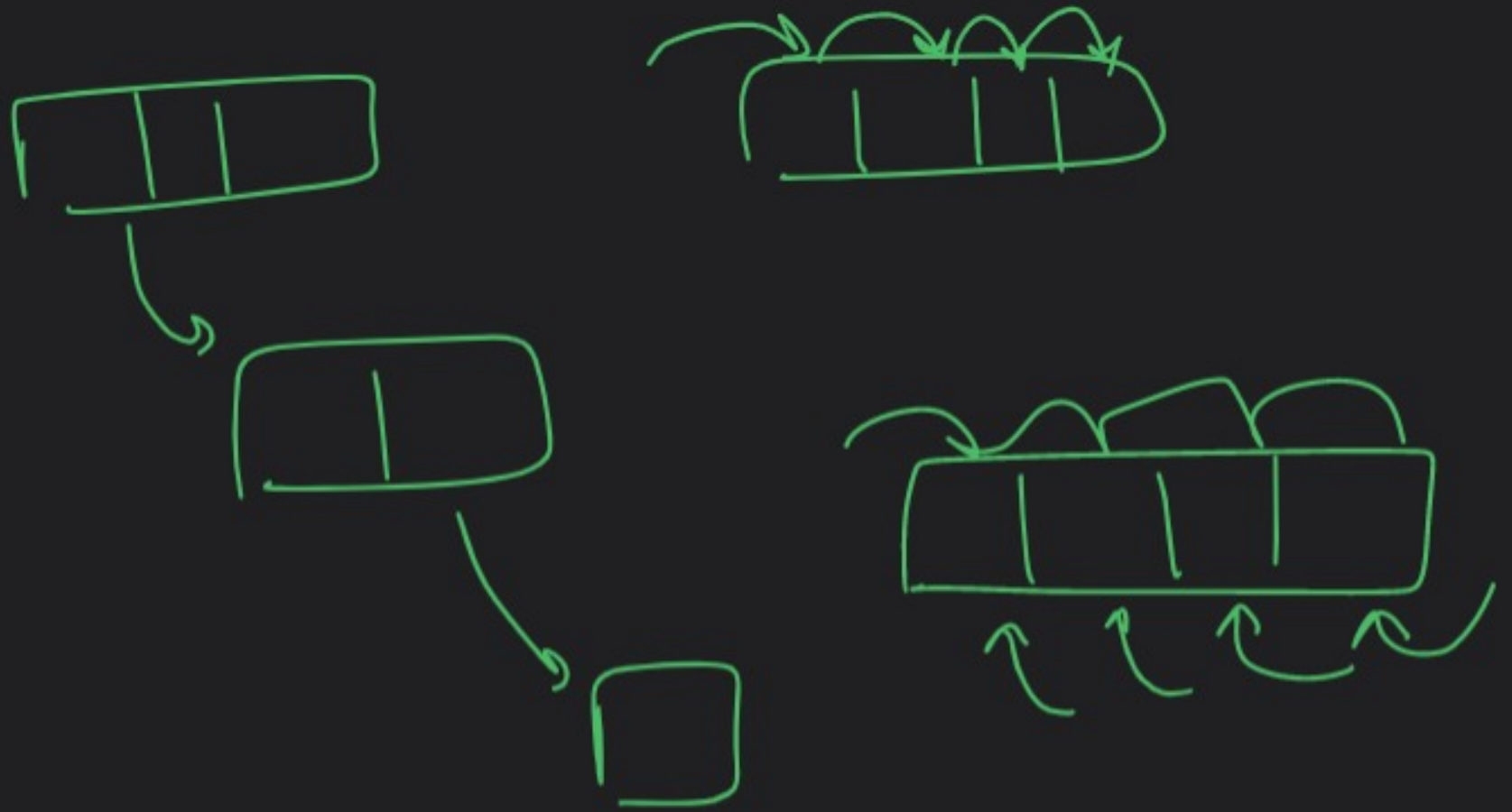
arr → 520

531

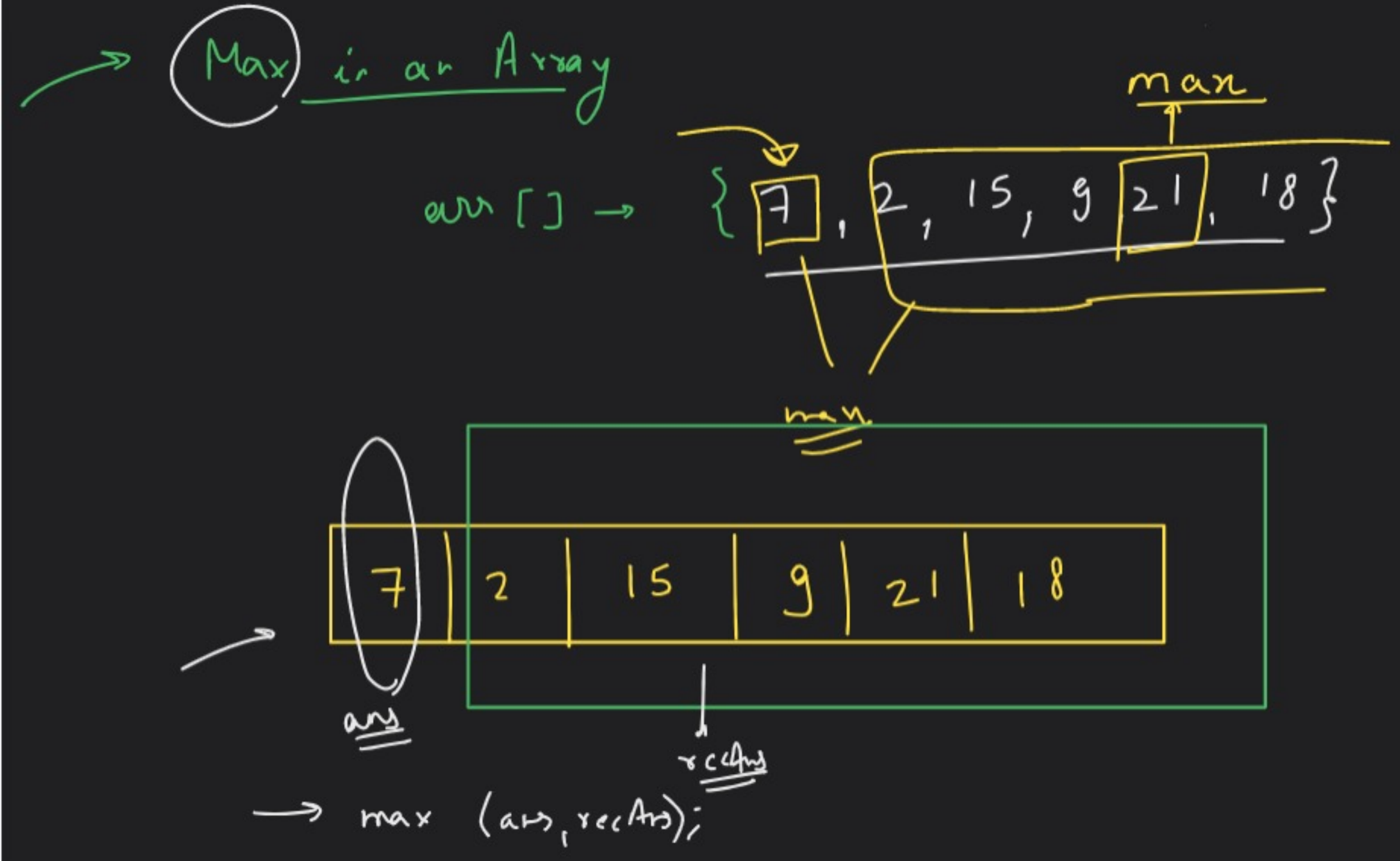
arr

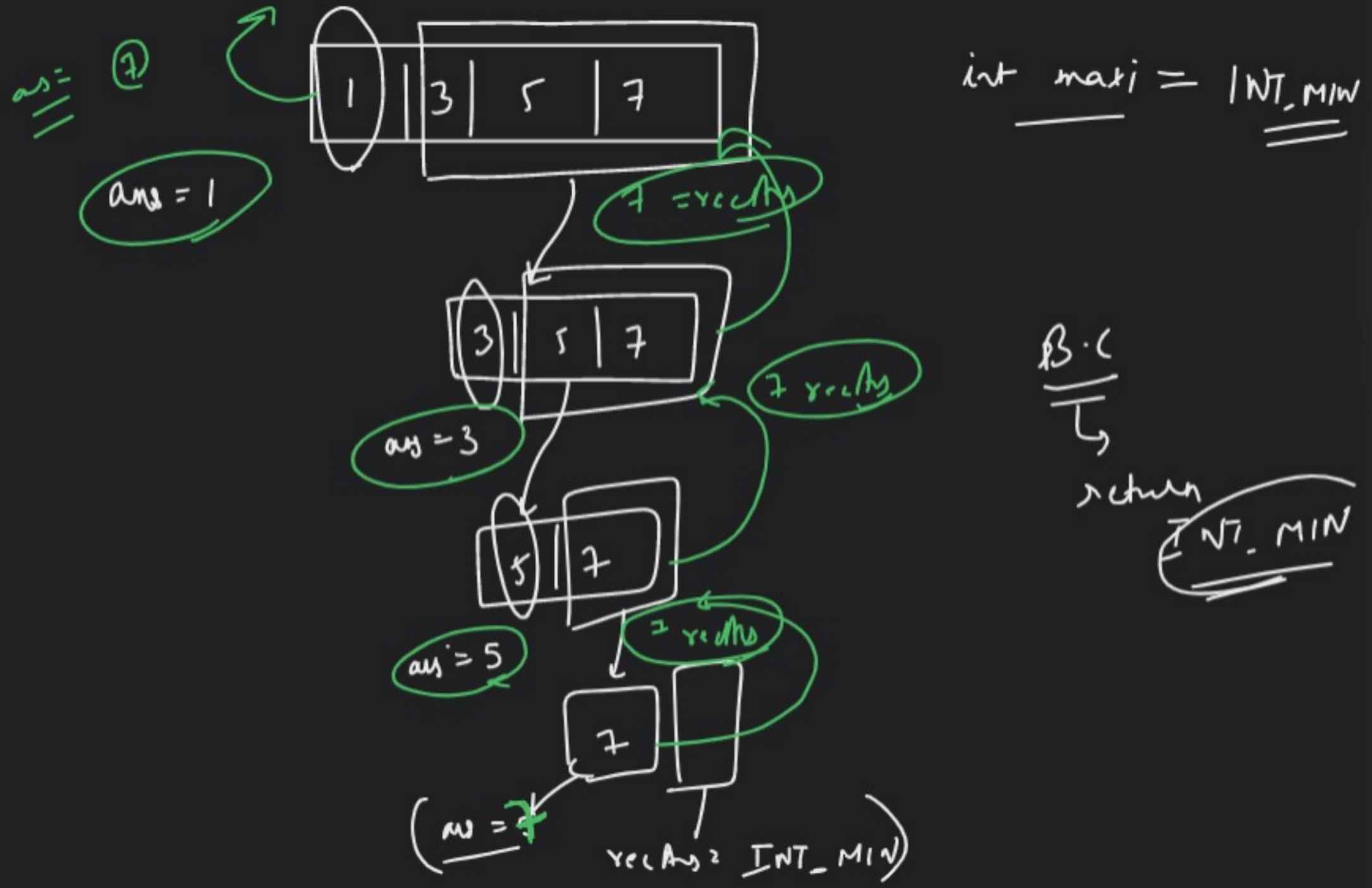
arr + 1

arr + 1 + 1

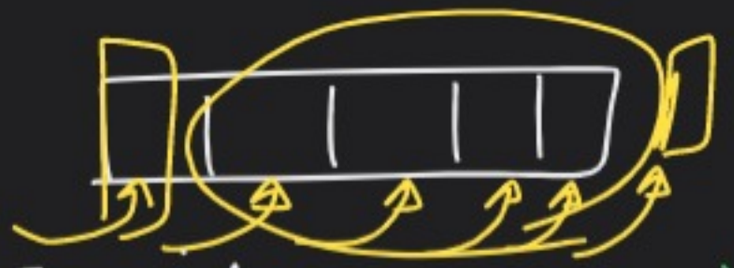


Max in an Array

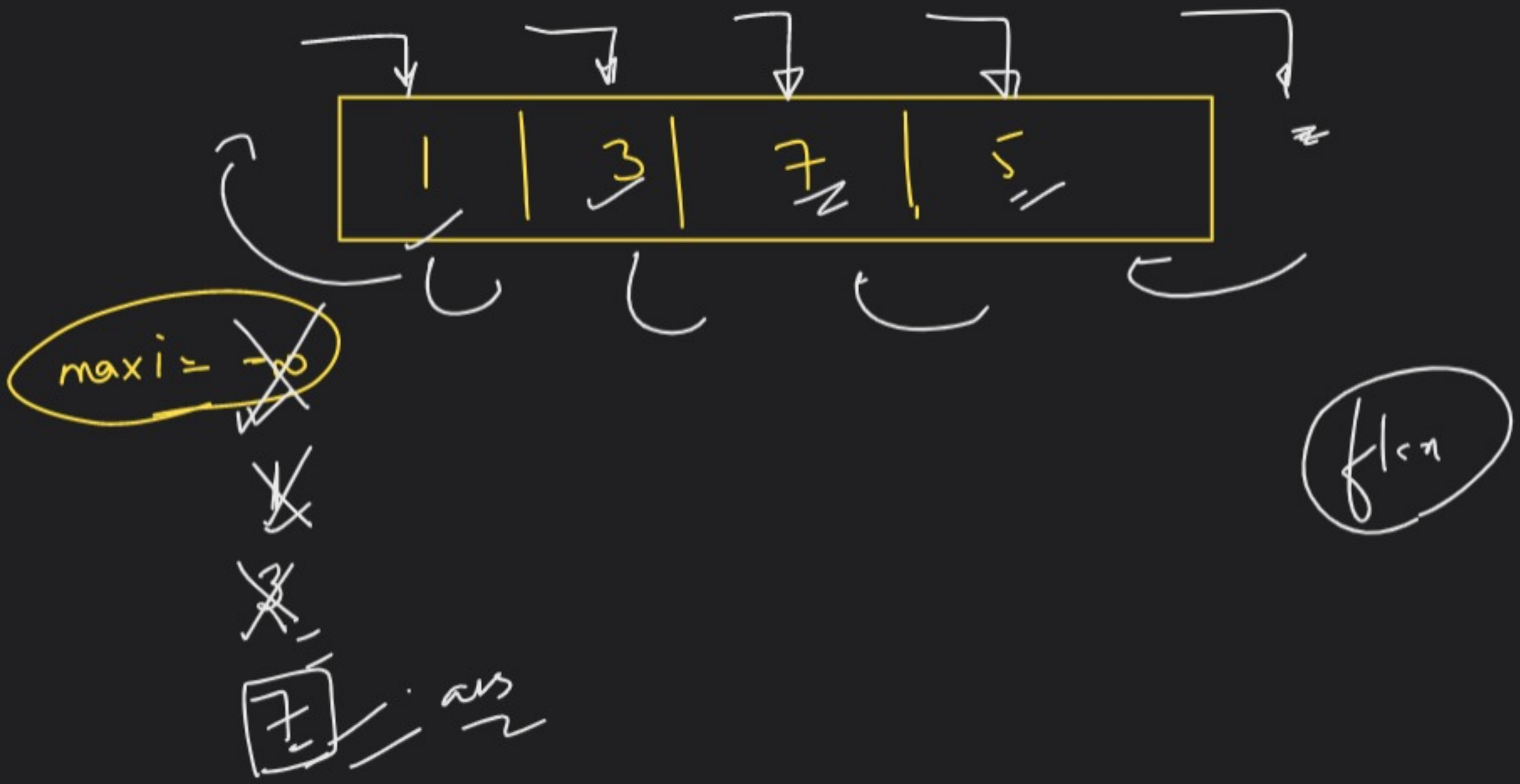



ans = ?
1 | 3 | 5 | 7
ans = 1
? = recAns
3 | 5 | 7
? recAns
ans = 3
5 | 7
ans = 5
? recAns
7
(ans = 7)
recAns = INT_MIN
int maxi = INT_MIN
B.C
return
INT_MIN

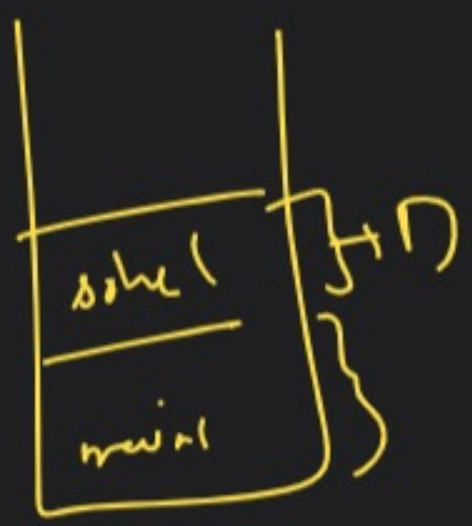
int maxi → INT_MIN

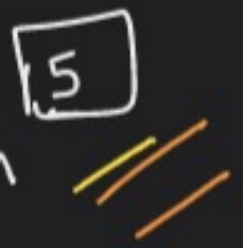


```
void solve ( int arr[], int size, int index, int maxi)
{
    // b.c
    if (index == size)
        return maxi;

    maxi = max (maxi, arr[index])

    solve ( arr, size, index+1, maxi)
}
```


1
3
7
5
maxi = -∞
ans
flcn

→ call by value





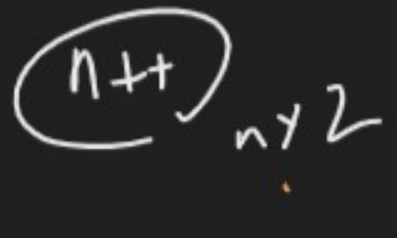

```
int n=5;
solve(n)
```

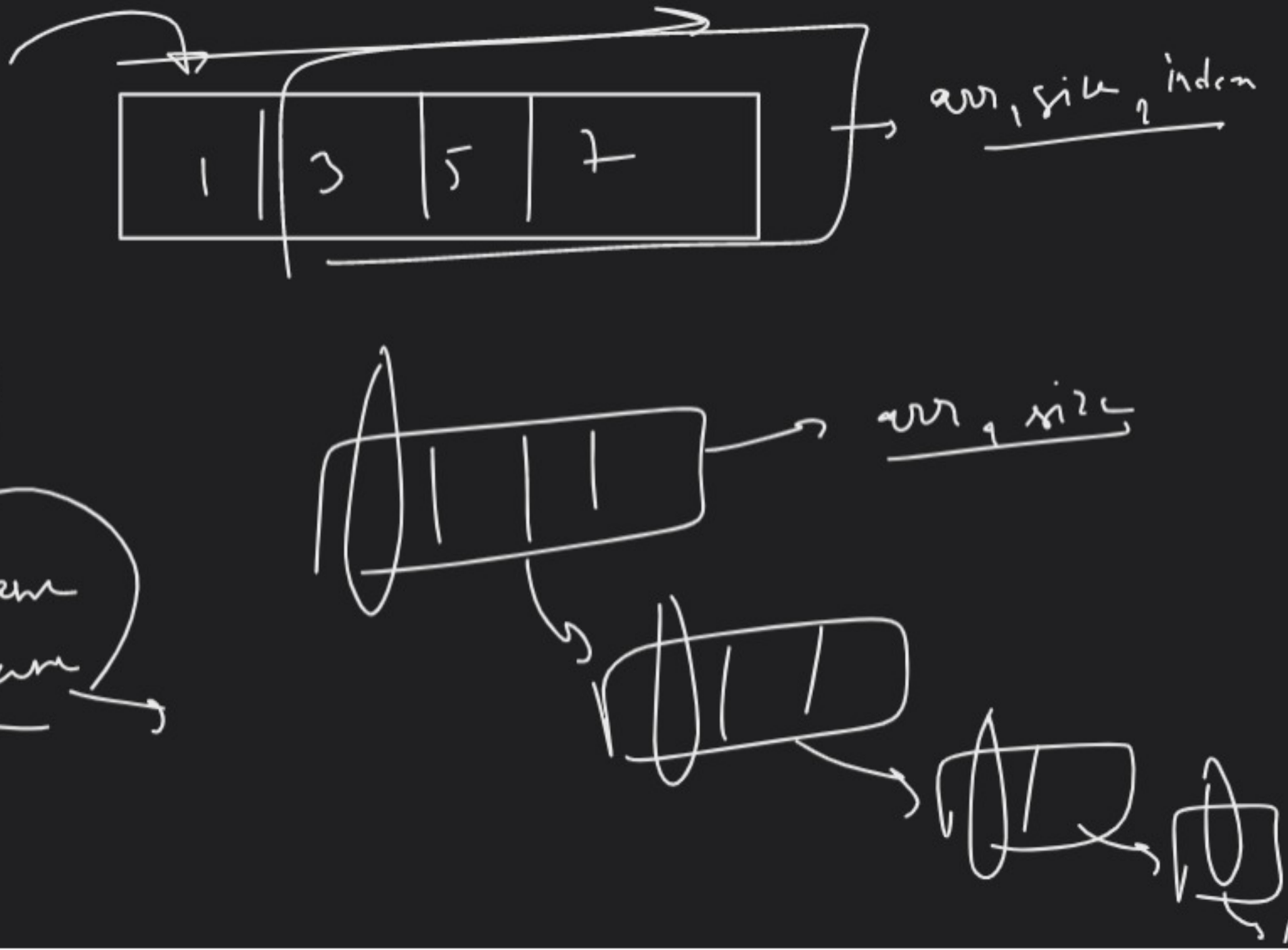
```
solve(int n)
{
}
```

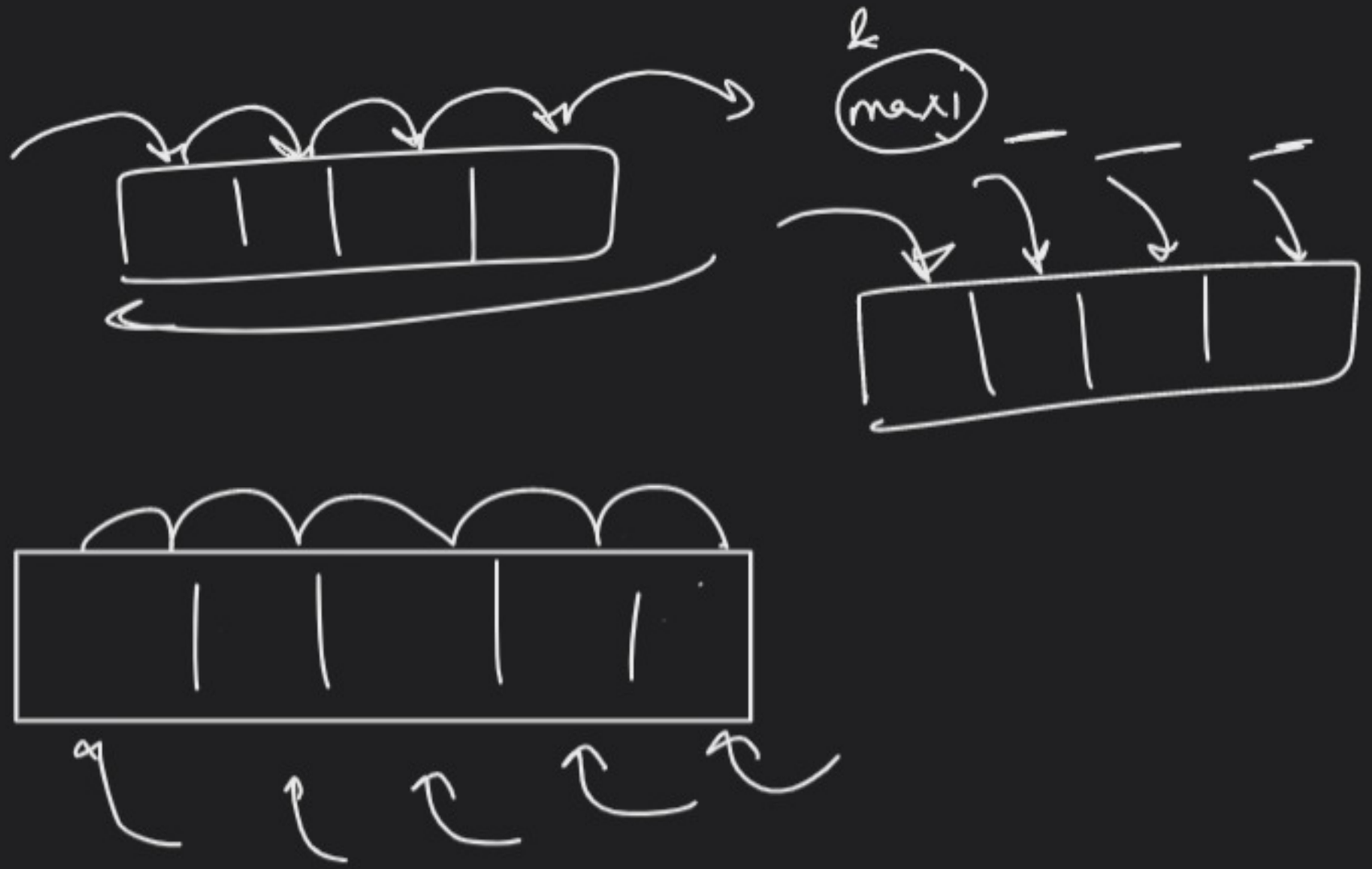
→ call by reference


```
main()
int n
solve(n)
```

```
solve(int& m)
{
}
```




1
3
5
7
arr, size, index
17
Classes
Blam
gum
arr, size


k
maxi

i/p → 1 | 3 | 2 | 3 | 6 | 4 | 3

0 1 2 3 4 5 6

target = 3

find the first index of target

ans → 1

first occurrence of target

Binary Search ?

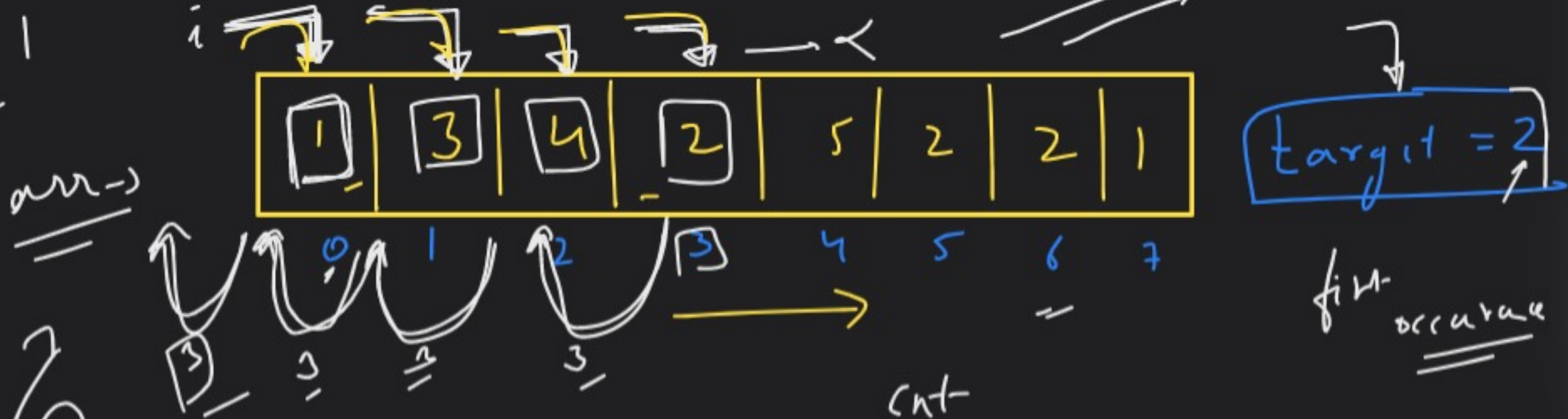
{2, 4, 6, 8, 4, 4, 2, 6}

0 1 2

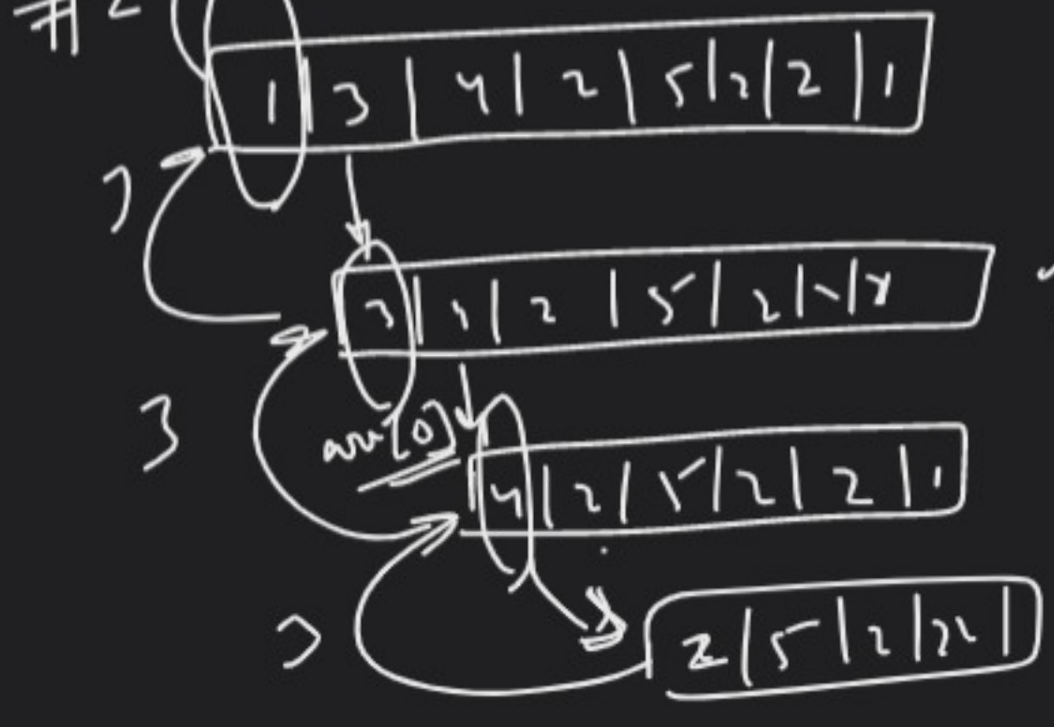
target = 4

ans = 6

#1



#2

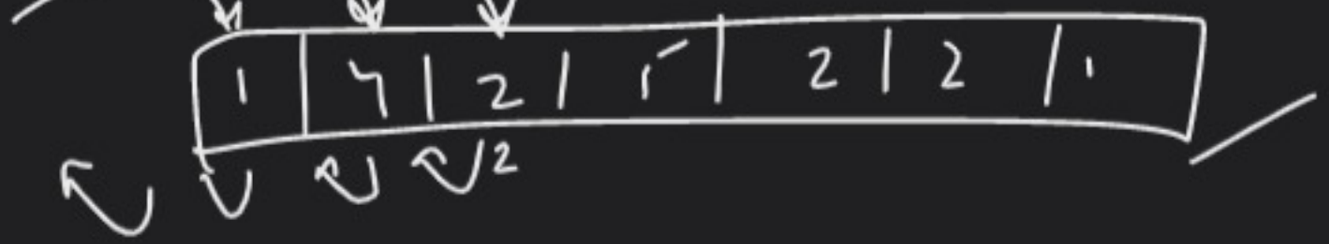


arr, size	index	target	cnt
			1
			2
			3

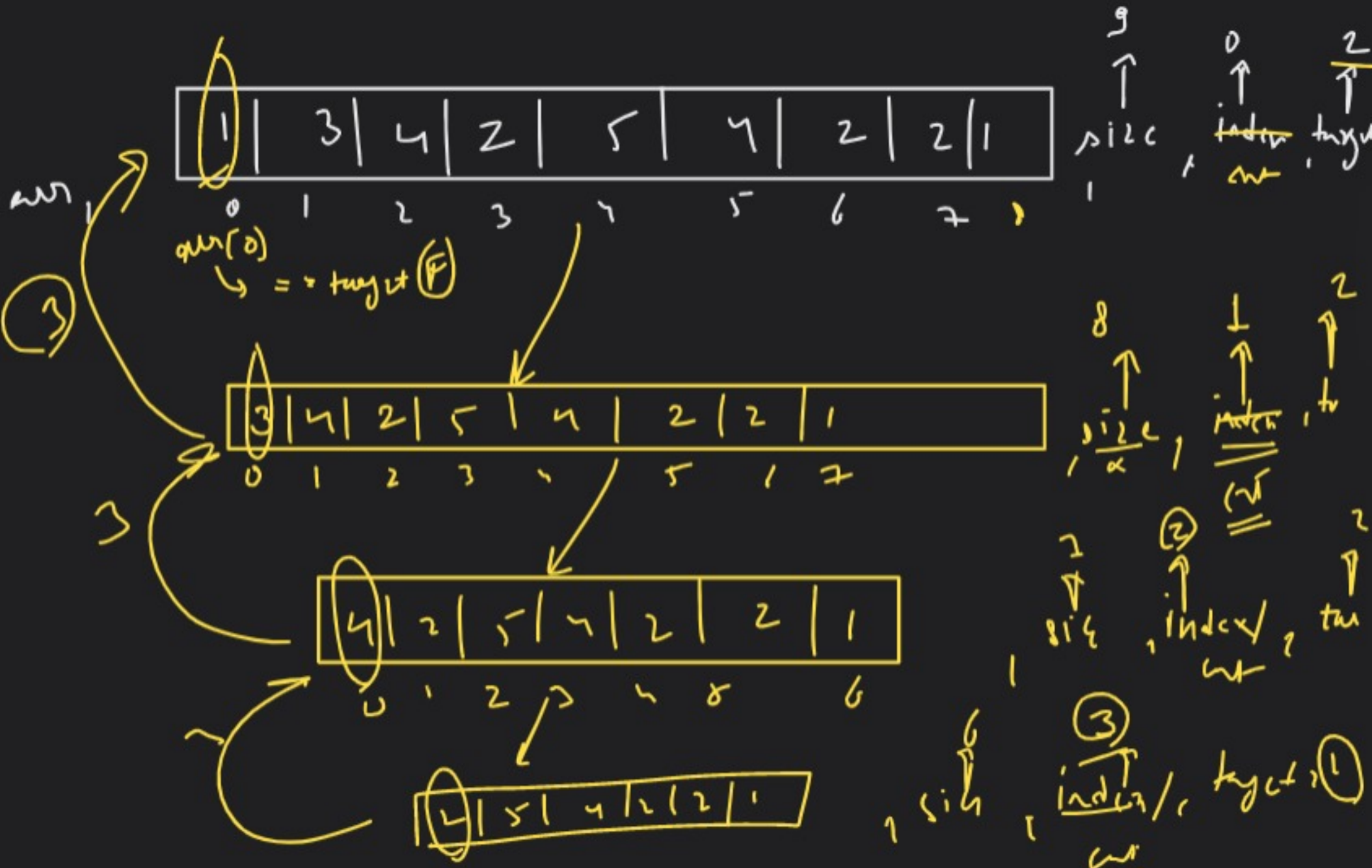
```
if (index == size)
    return -1;
```

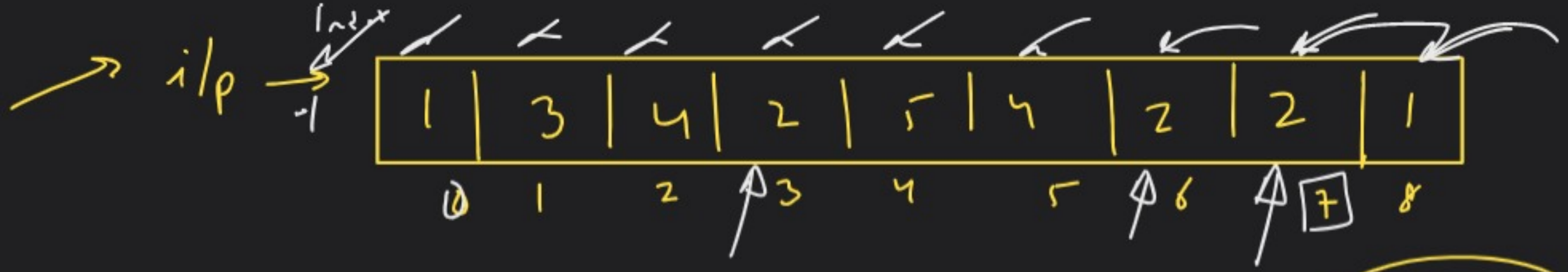
#3

arr = 2 — by reference



int size = sizeof(arr) / sizeof(int)

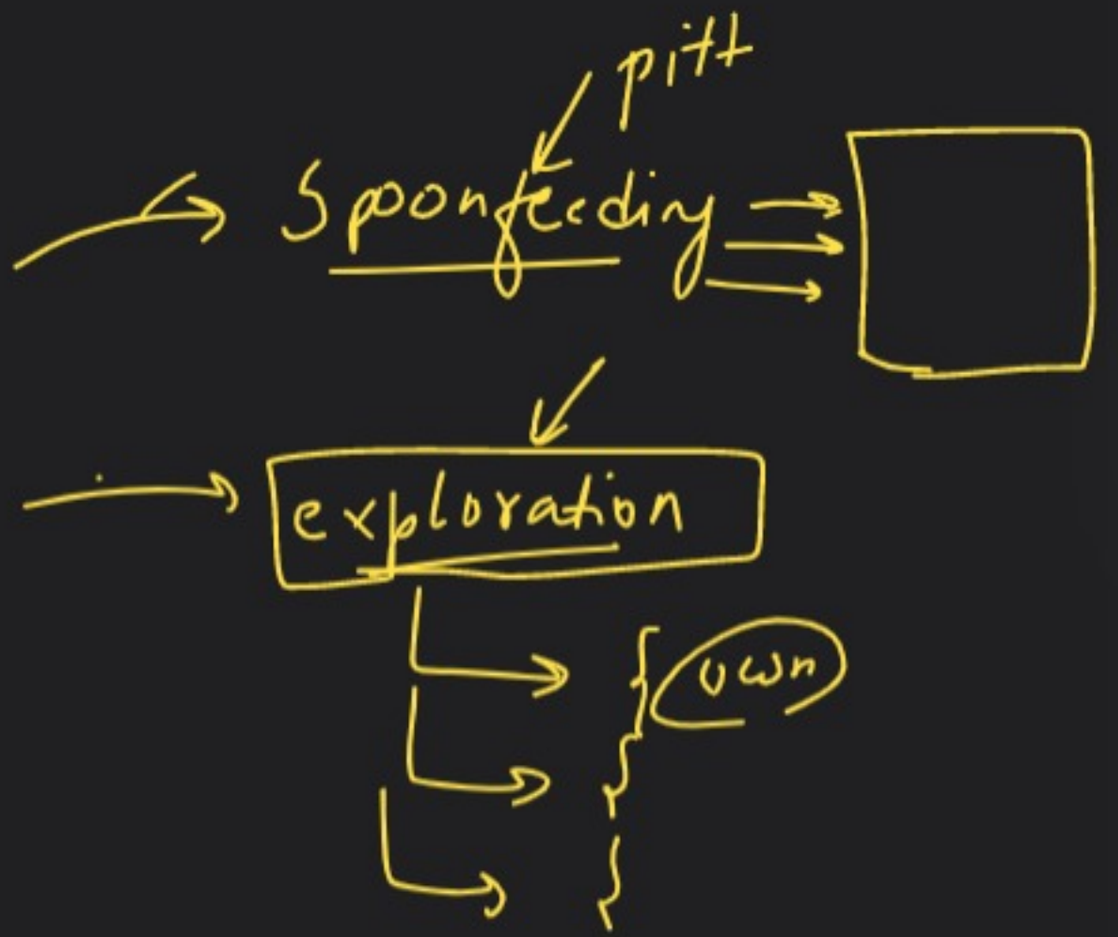

1 3 4 2 5 4 2 2 1
0 1 2 3 4 5 6 7
arr
arr[0] == target (F)
3
3 4 2 5 4 2 2 1
0 1 2 3 4 5 6 7
3
4 2 5 4 2 2 1
0 1 2 3 4 5 6
2
4 5 4 2 2 1
size
index
target
9
0
2
8
1
2
2
2
3
1



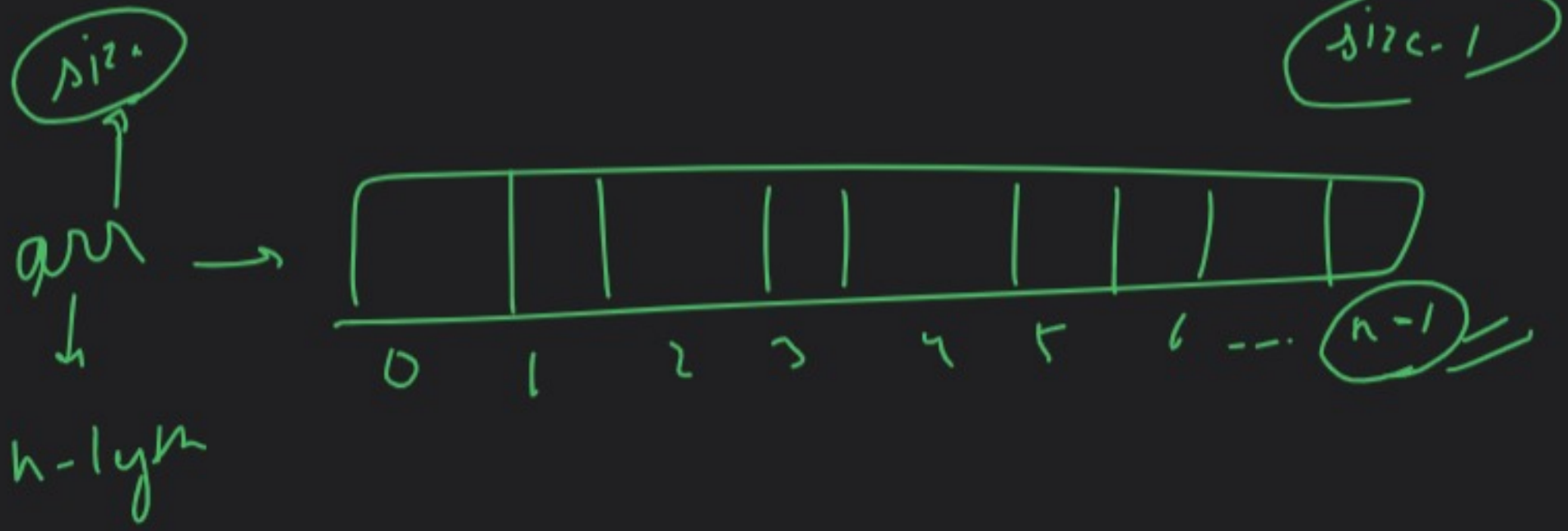
target = 2

find Last occurence

o/p -> ans = 7


pitt
Spoonfeeding
exploration
own


arr


size
size-1
arr
n-1
0 1 2 3 4 5 6 ... n-1
n-1 ya

① find first occ

② find last occ

③ find all occurrences

arr

1	3	4	2	5	2	2	1

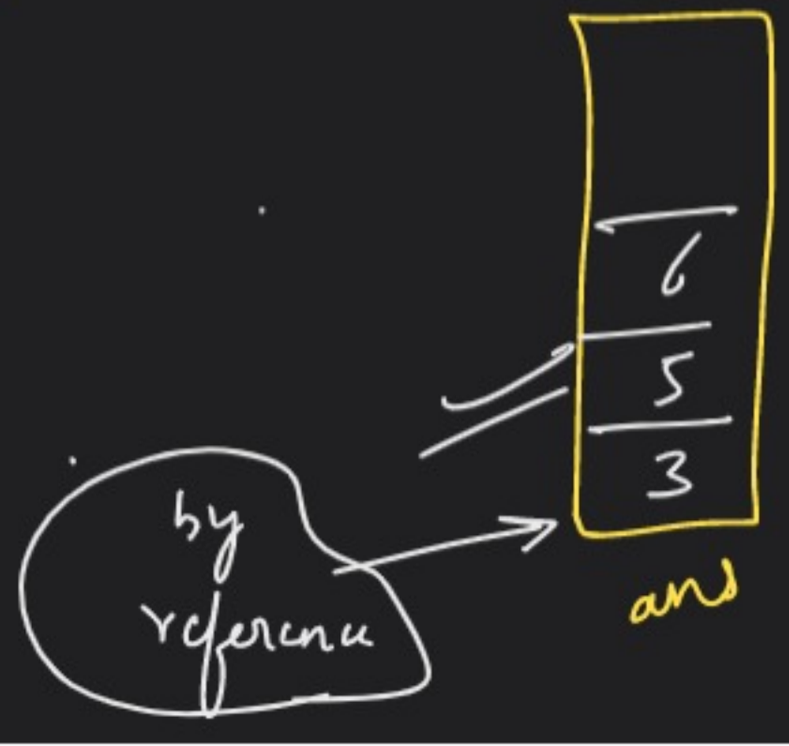
target = 2

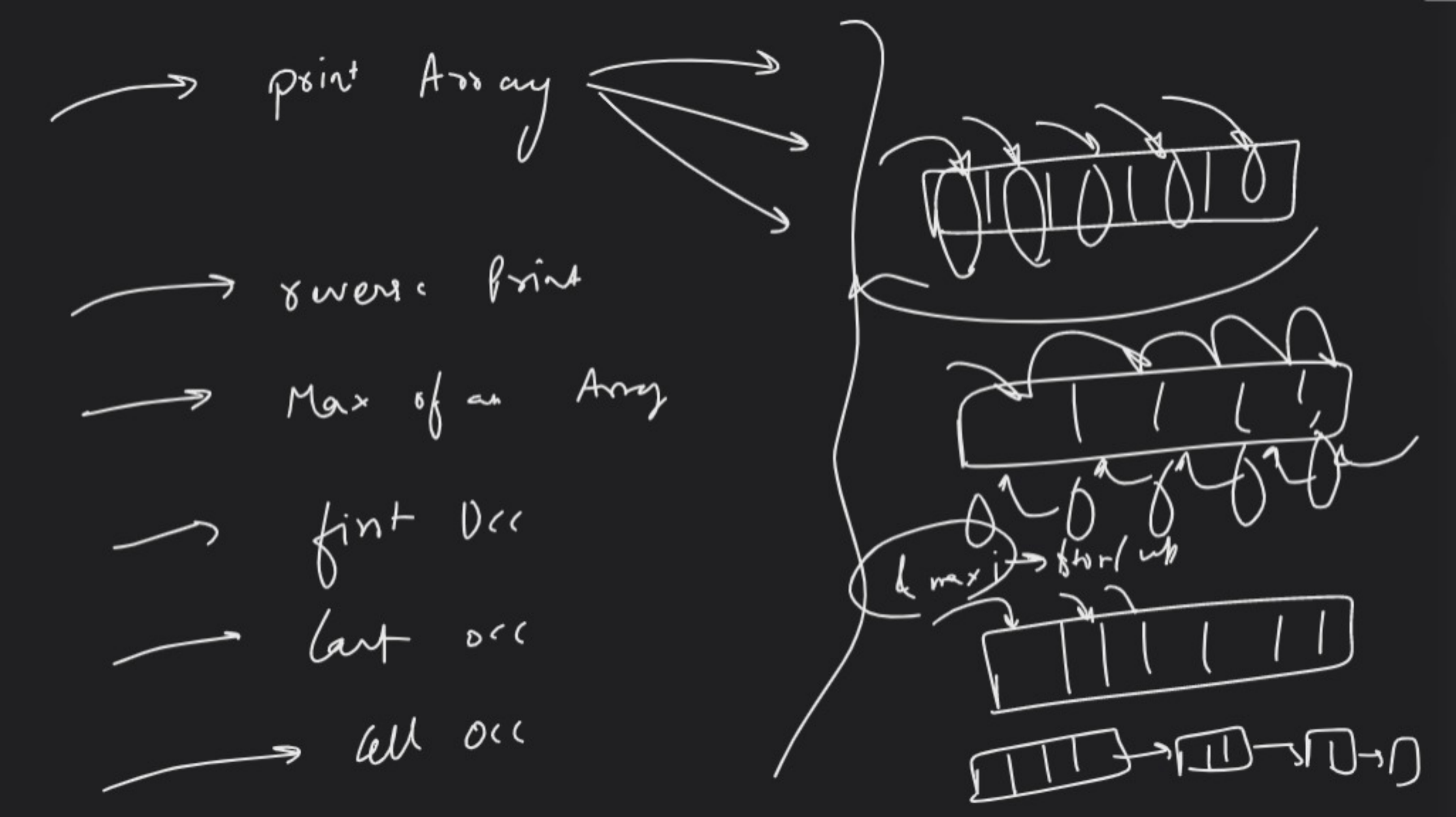
find all occurrences

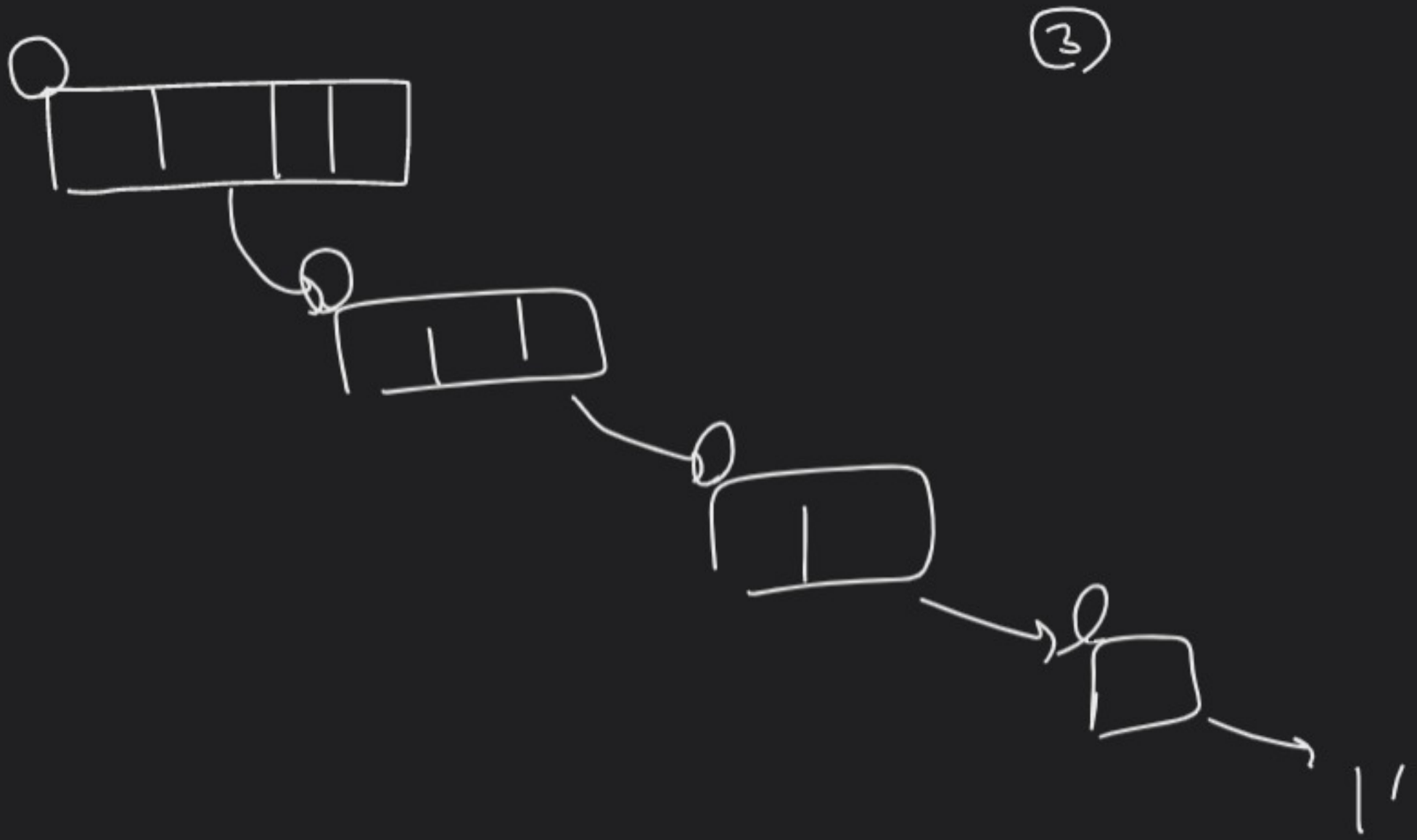
0 1 2 3 4 5 6 7

approach

-1 → {3, 5, 6}




print Array
reverse Print
Max of an Array
first Occ
last occ
all Occ
&max i
short up


3