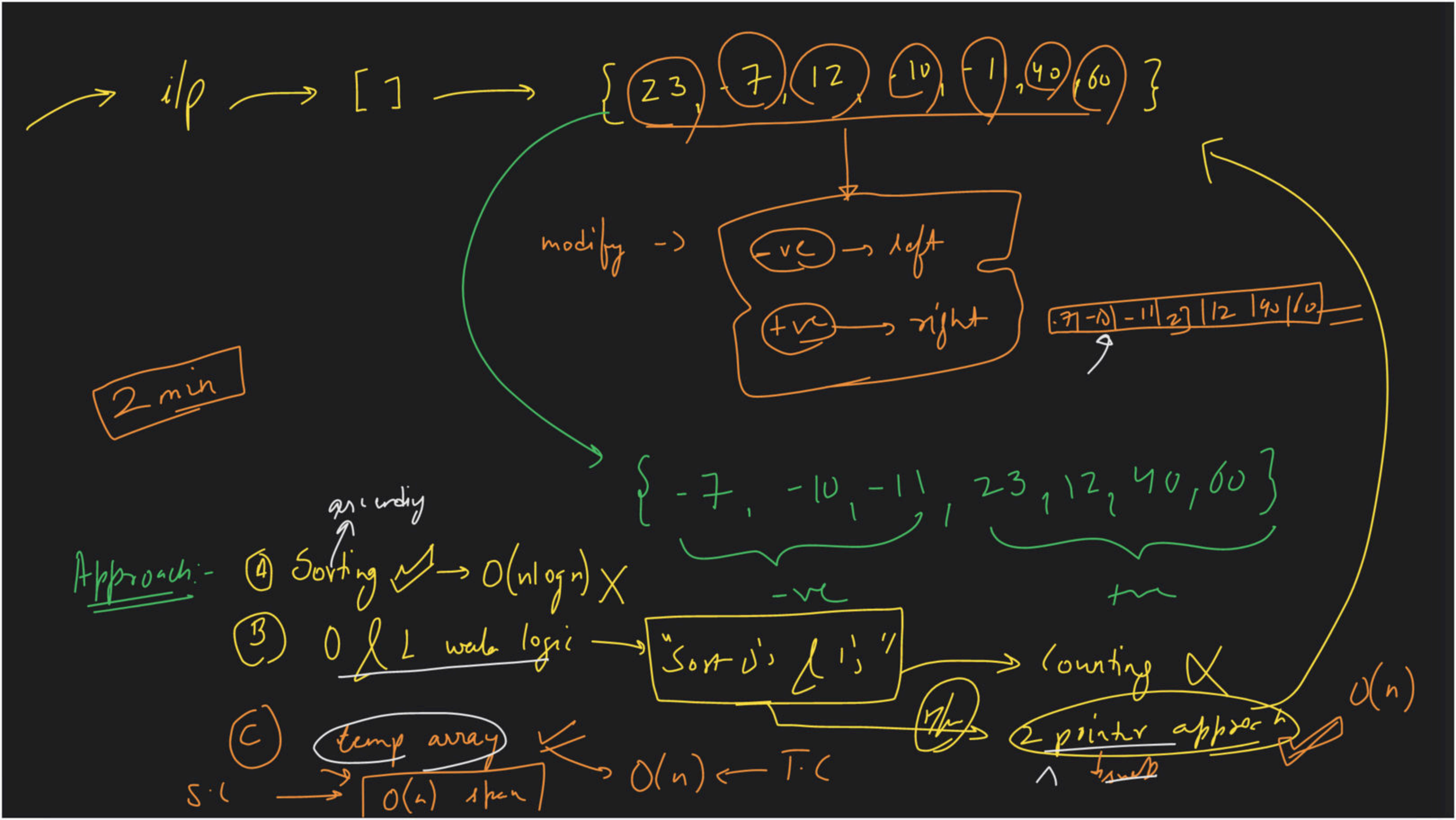
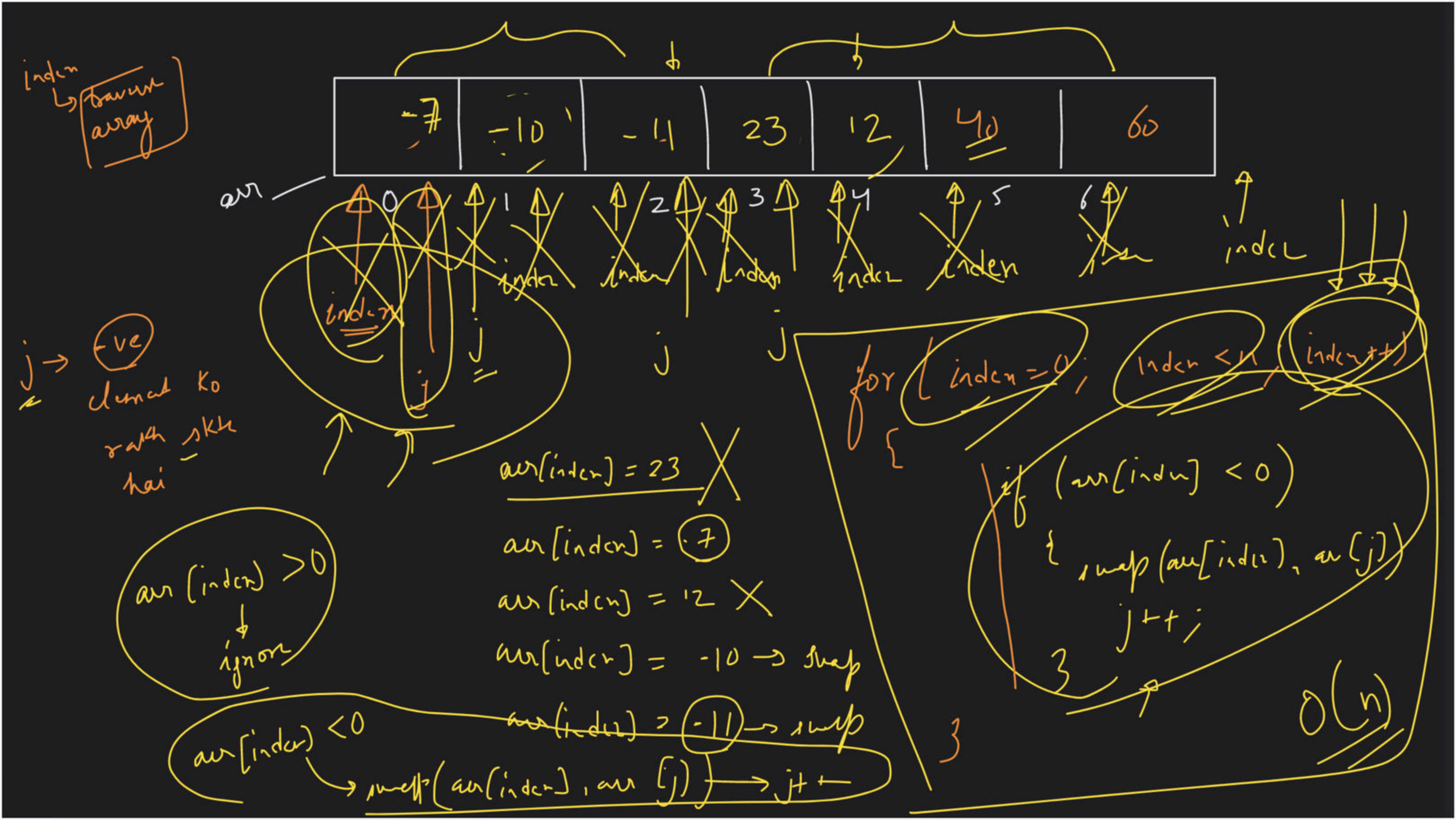
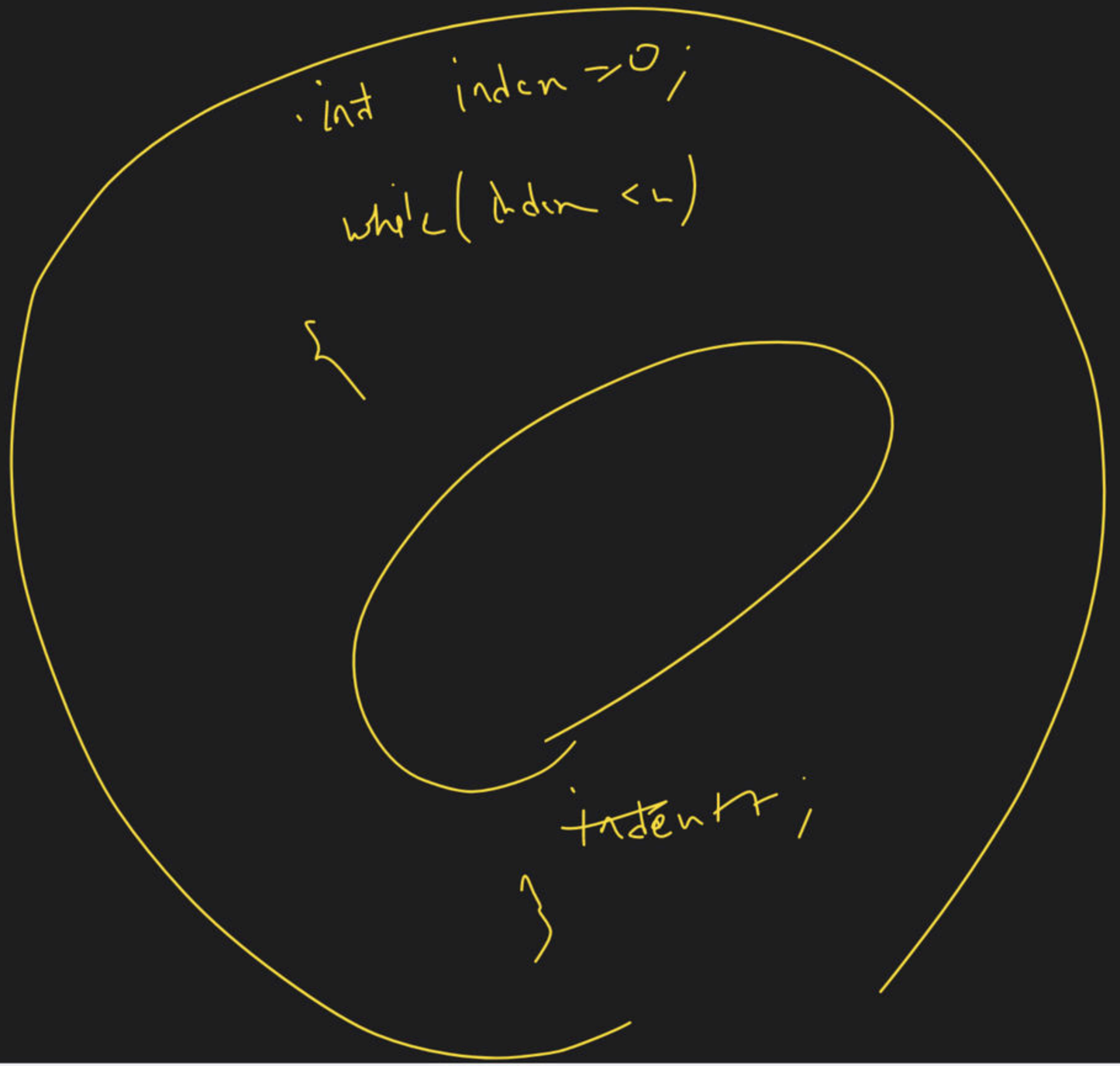
Arrays - [Extra Class]

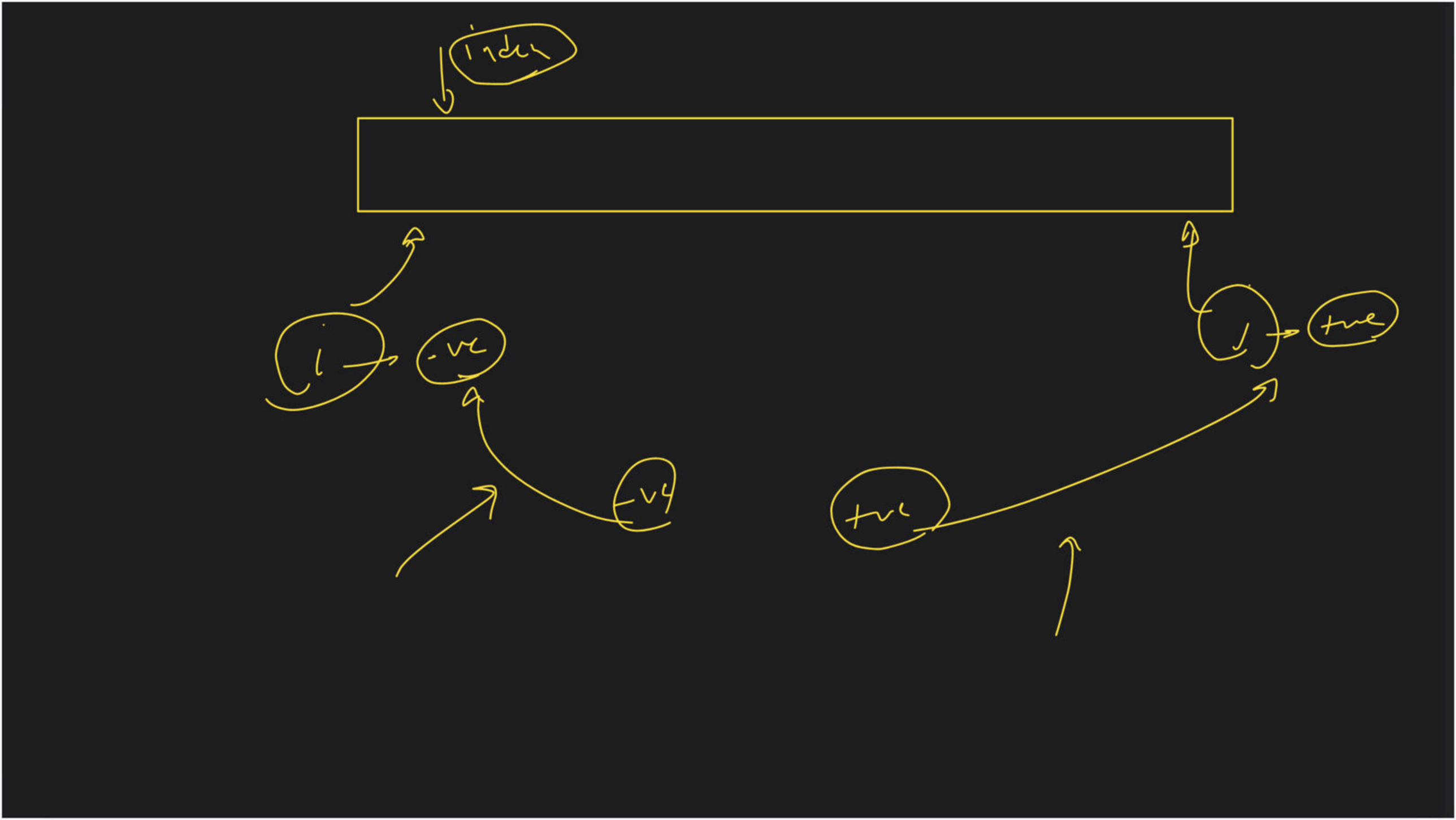
Special class

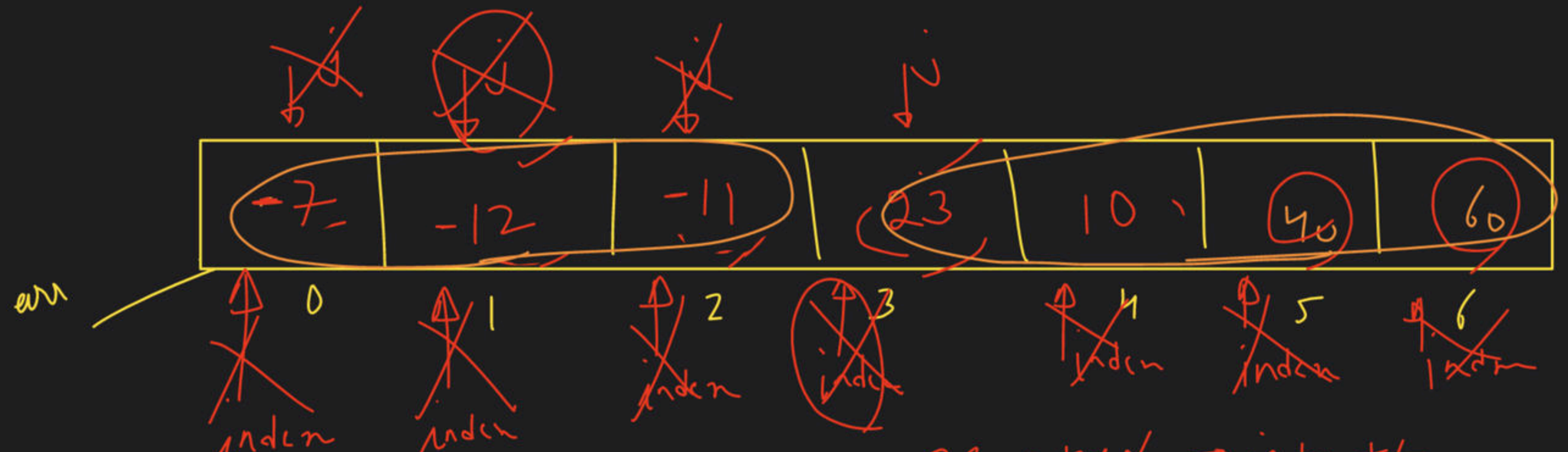
Love Babbar • Sept 12, 2023


i/p
[ ]
{ 23, -7, 12, -10, -1, 40, 60 }
modify ->
-ve → left
+ve → right
-7 | -10 | -11 | 23 | 12 | 40 | 60
2 min
{ -7, -10, -11, 23, 12, 40, 60 }
-ve
+ve
Approach:-
ascending
④ Sorting → O(nlogn) X
③ O(L) work logic → "Sort 0's & 1's"
Counting
2 pointer approach
O(n)
Two
⑤ temp array
S.C → O(n) span
O(n) ← T.C


index → [sorted array]
-7
-10
-11
23
12
40
60
arr
0
1
2
3
4
5
6
index
i
j
j → -ve element ko swap nahi karna hai
arr[index] = 23 X
arr[index] = -7
arr[index] = 12 X
arr[index] = -10 → swap
arr[index] = -11 → swap
arr[index] < 0 → swap(arr[index], arr[j]) → j++
arr[index] > 0 → ignore
for (index = 0; index < n; index++)
{
if (arr[index] < 0)
{ swap(arr[index], arr[j])
j++;
}
}
O(n)


int indent = 0;
while (indent < n)
{
indent++;
}


index
i
-ve
-ve
+ve
+ve
i
true



23 → true ✗ → indent++

-7 → -ve → swap → j++ → indent++

10 → true ✗ → indent++

-12 → -ve → swap → j++ → indent++

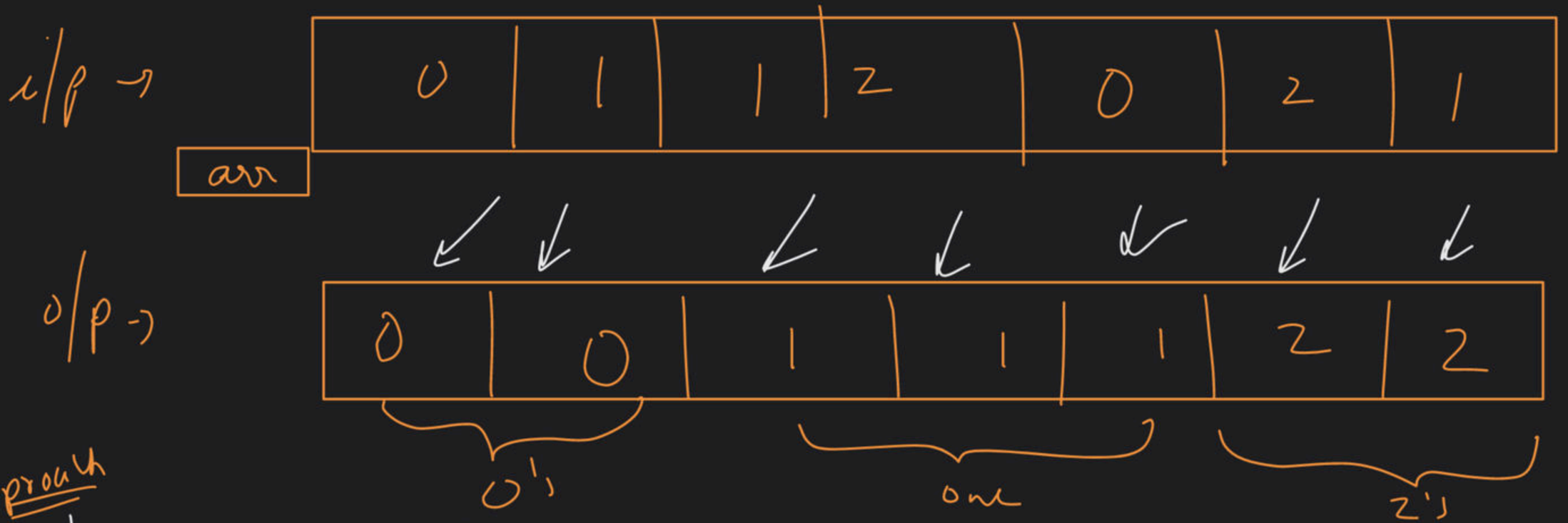
-11 → -ve → swap → j++ → indent++

40 → true ✗ → indent++

60 → true ✗ → indent++

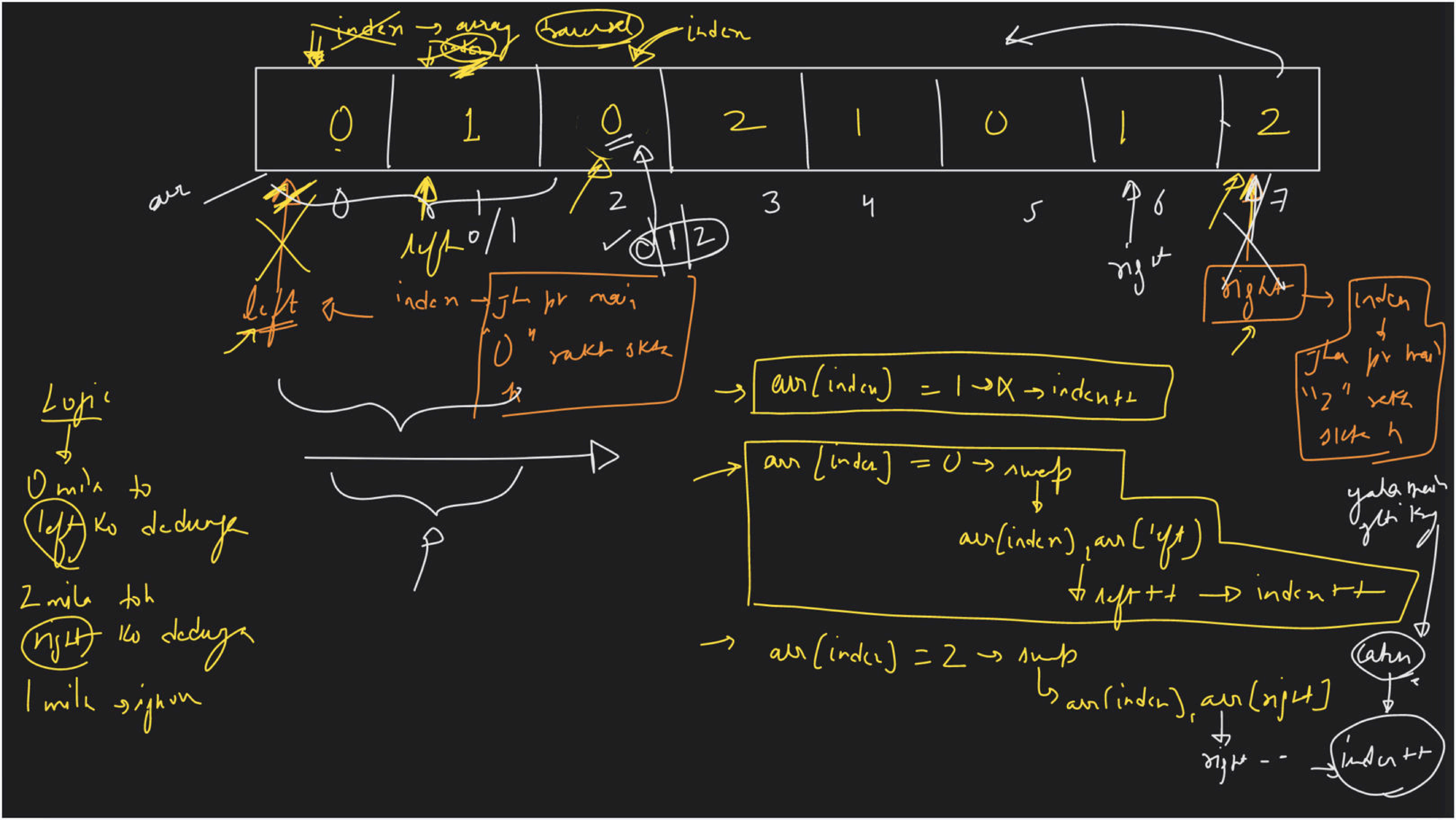
Sort 0's 1's & 2's

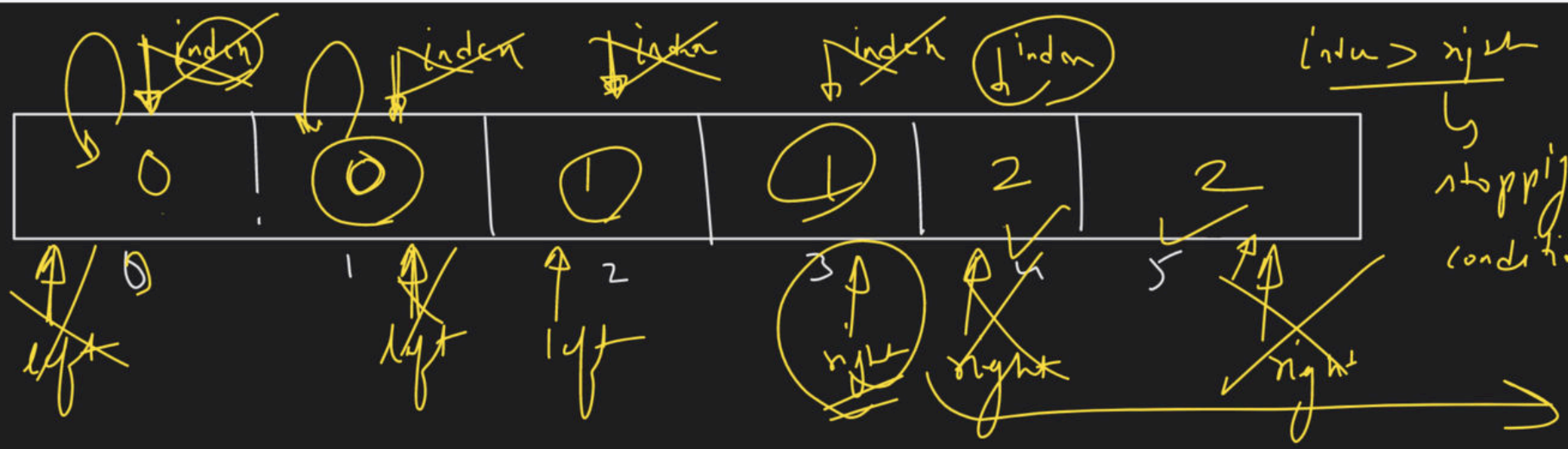
zero count = 2
one count = 3
two count = 2



Approach

- counting → O(n)
- sorting → O(n log n)
- 2 pointer approach


index → array traversal
index
arr
0 1 0 2 1 0 1 2
0 1 2 3 4 5 6 7
left
0/1
0 | 1 | 2
right
index → jha pr hai "0" rakh skte
right → index jha pr hai "2" rakh skte h
Logic
0 mila to left ko dedunga
2 mila toh right ko dedunga
1 mila → ignore
arr(index) = 1 → X → index++
arr[index] = 0 → swap
arr(index), arr(left)
left++ → index++
arr(index) = 2 → swap
arr(index), arr(right)
right --
index++
yaha main galti ki



arr[index] = 2 → swap → index, right → right - -

arr[index] = 0 → swap → index, left → left + + → index + +

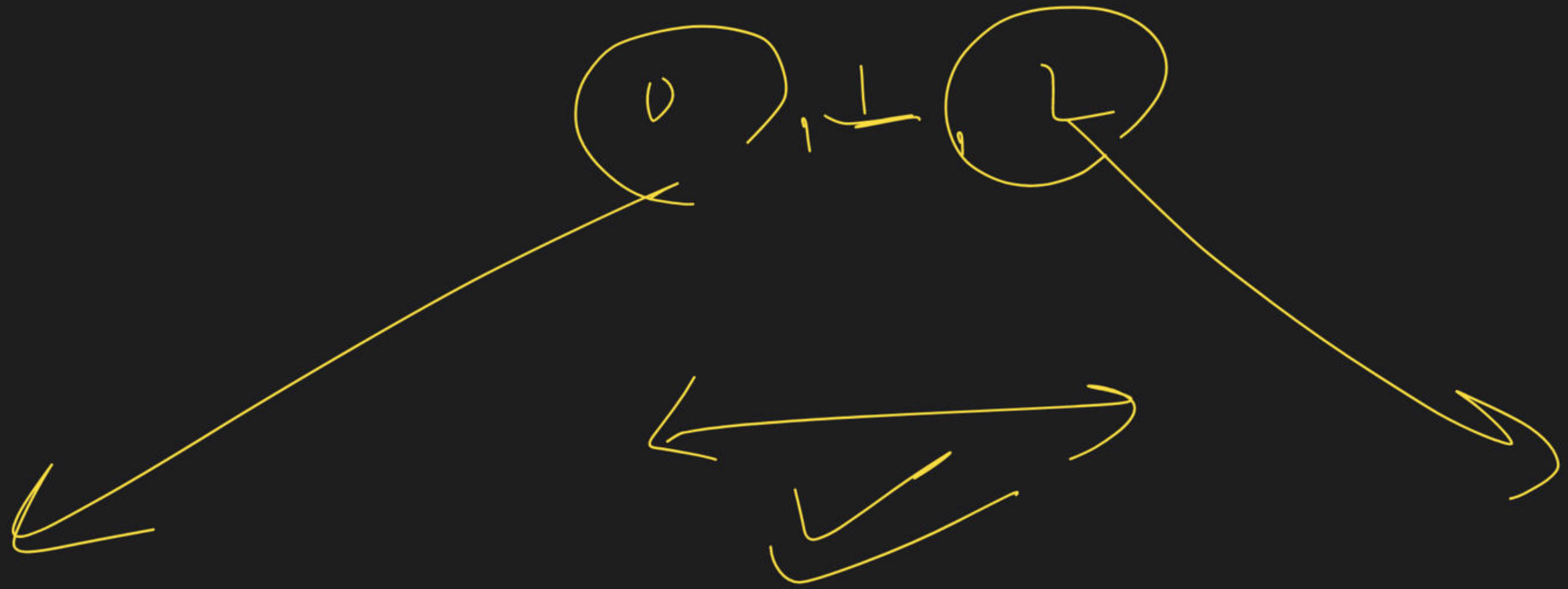
arr[index] = 0 → swap → index, left → left + + → index + +

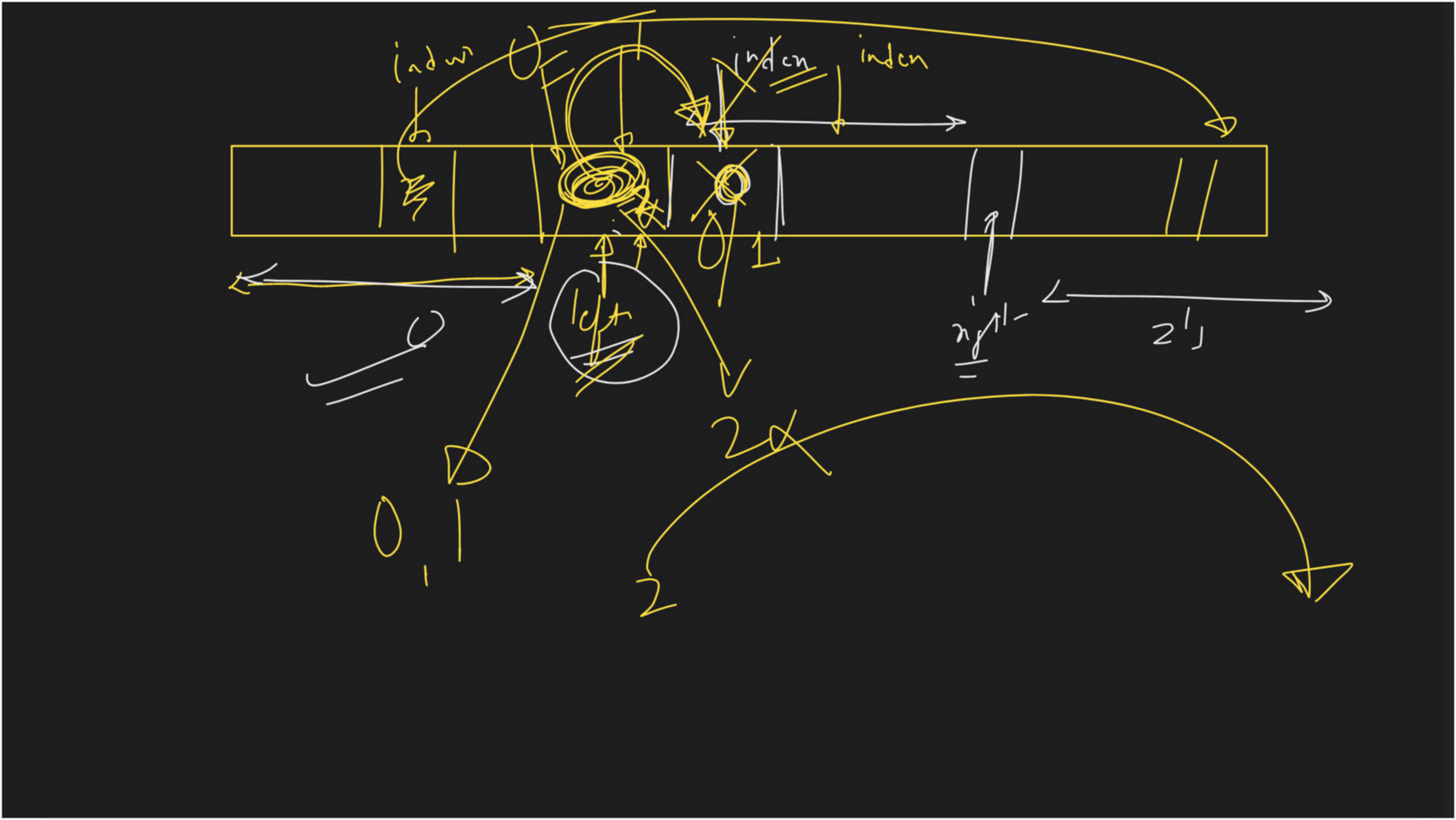
arr[index] = 2 → swap → index, right → right - -

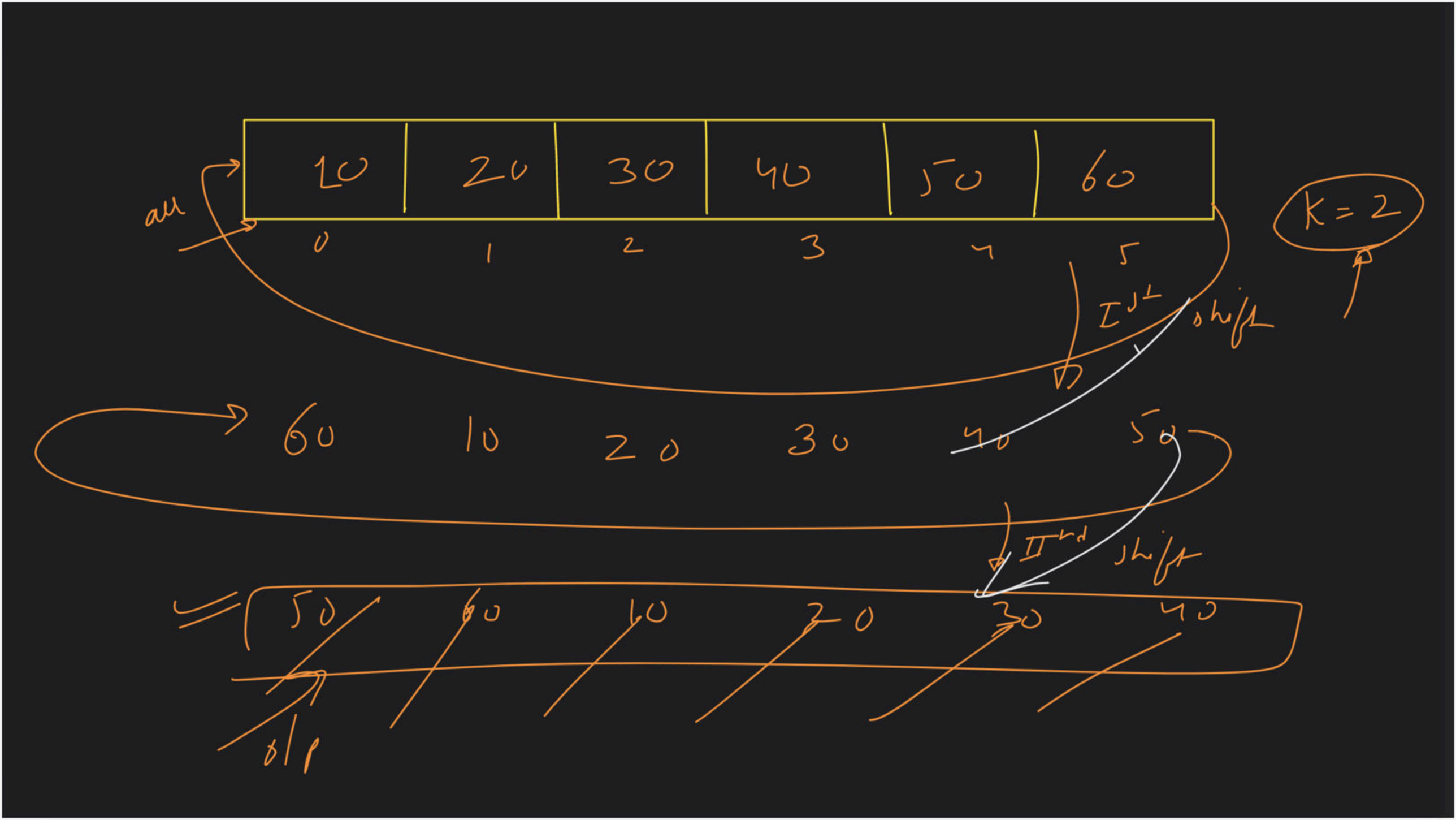
arr[index] = 1 → index + +

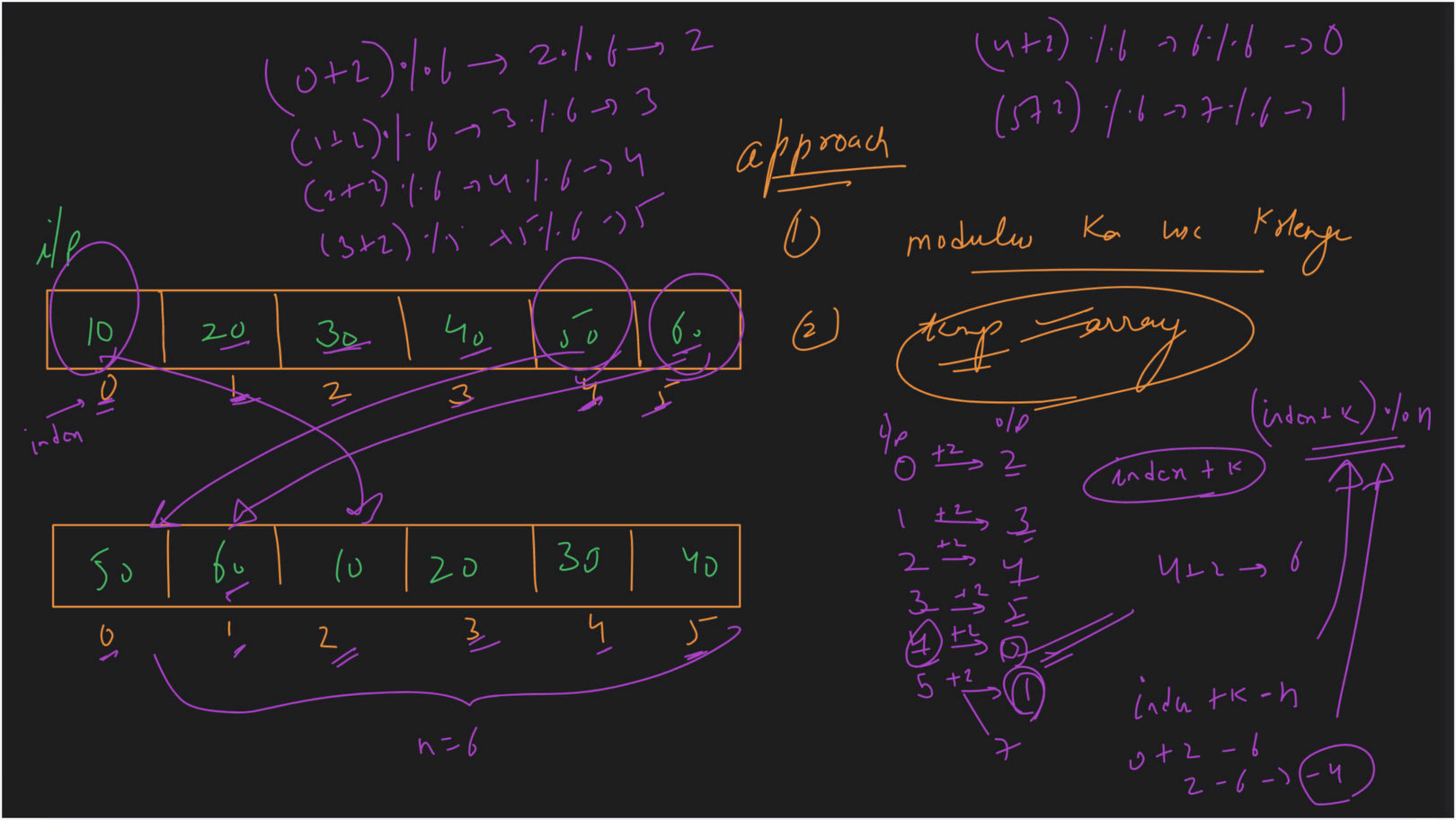
arr[index] = 1 → index + +

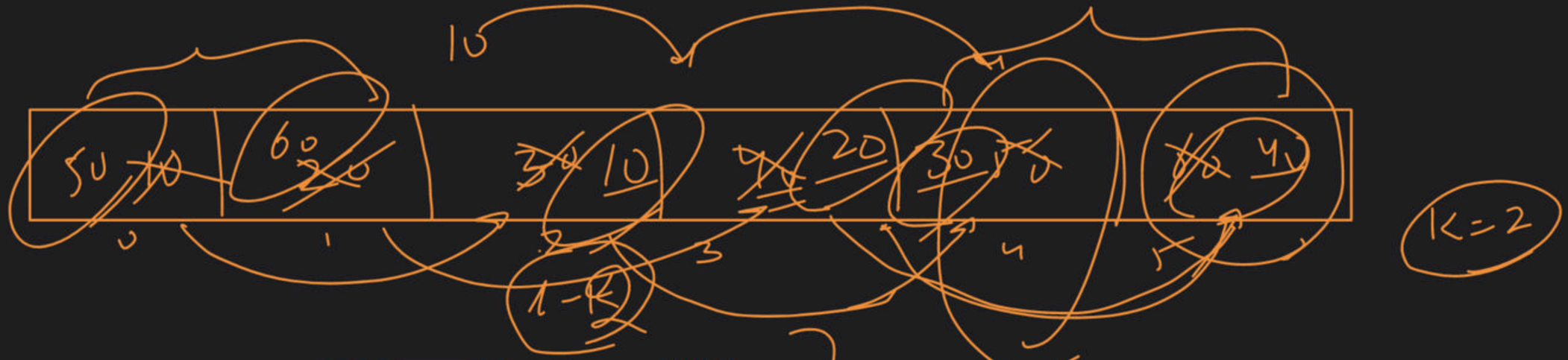
left → 1 → right → 2


0
1
2


indow
indcn
indcn
0
1
left
right
0
1
2
2


10
20
30
40
50
60
arr
0
1
2
3
4
5
k = 2
I^st shift
60
10
20
30
40
50
II^nd shift
50
60
10
20
30
40
o/p


(0+2)%6 → 2%6 → 2
(1+2)%6 → 3%6 → 3
(2+3)%6 → 4%6 → 4
(3+2)%6 → 5%6 → 5
(4+2)%6 → 6%6 → 0
(5+2)%6 → 7%6 → 1
approach
1)
modulus Ka use Krenge
2)
temp array
i/p
10 20 30 40 50 60
0 1 2 3 4 5
index
50 60 10 20 30 40
0 1 2 3 4 5
n=6
o/p
0 +2 → 2
1 +2 → 3
2 +2 → 4
3 +2 → 5
4 +2 → 0
5 +2 → 1
7
index + k
(index+k)%n
4+2 → 6
index+k-n
0+2-6
2-6 → -4



(A) 2 size → 50 | 60 ← temp array → O(K)

(B) for (n-1 → >=0 → wrong

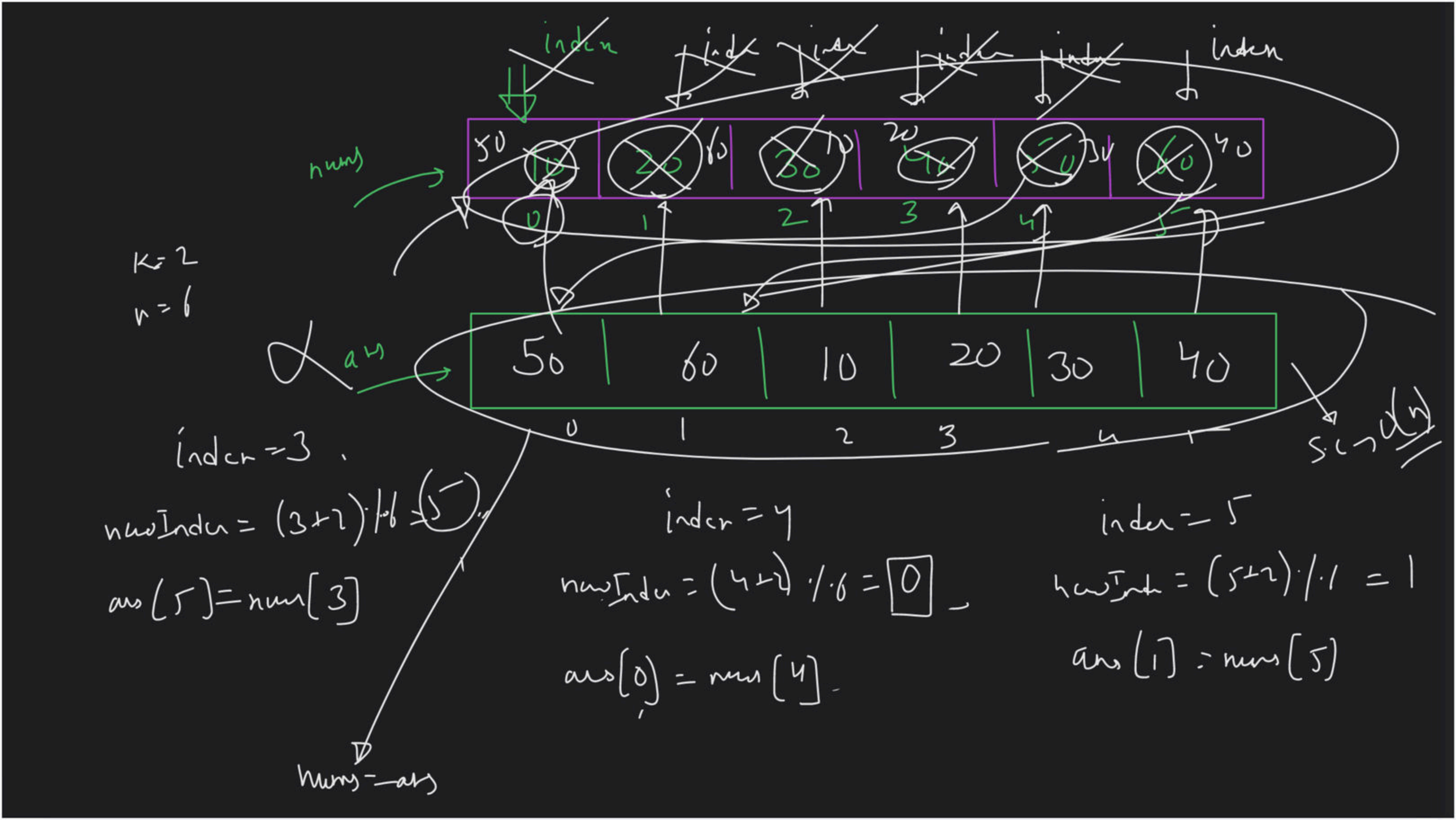
{

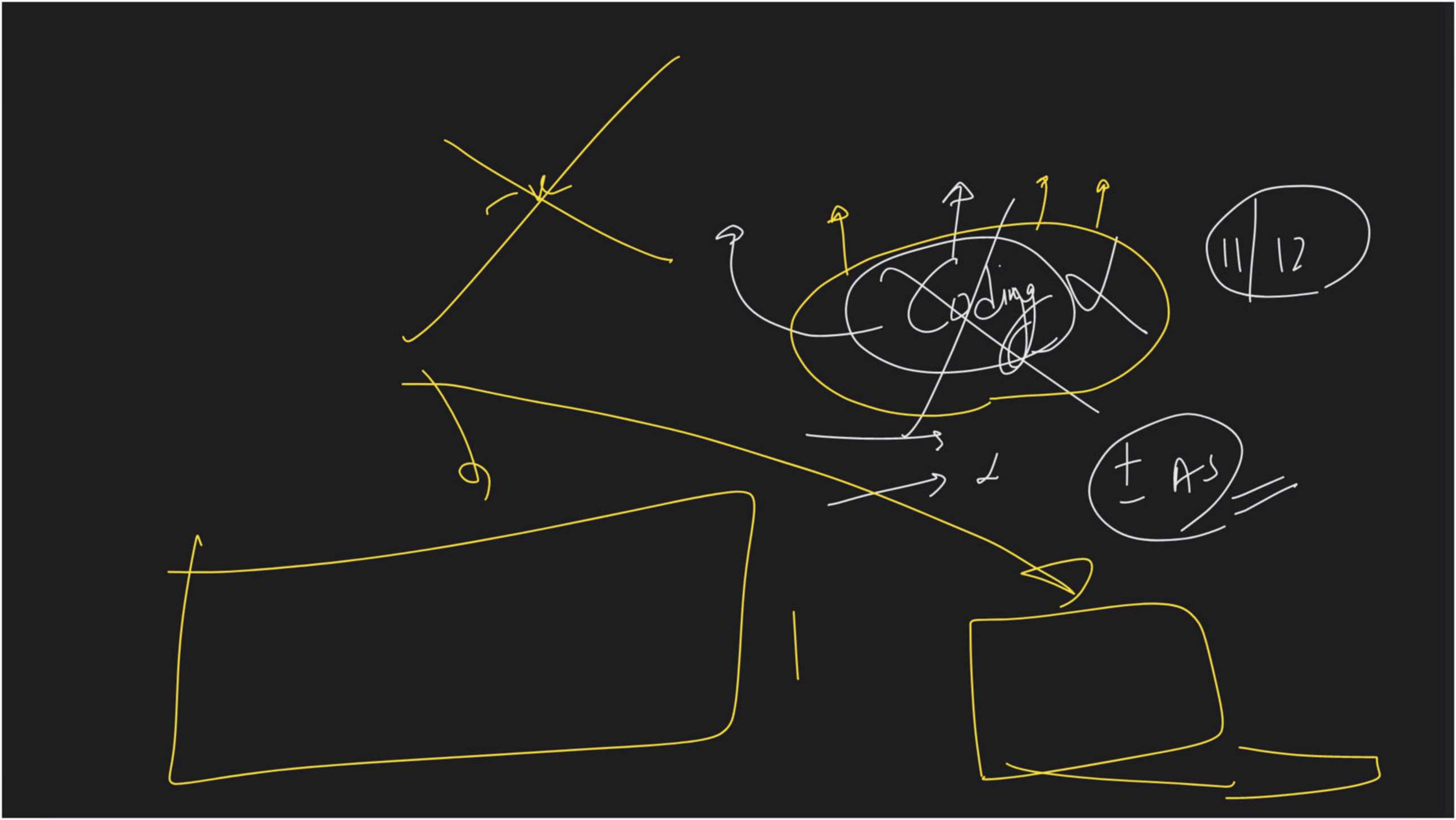
arr[i] = arr[i-K]

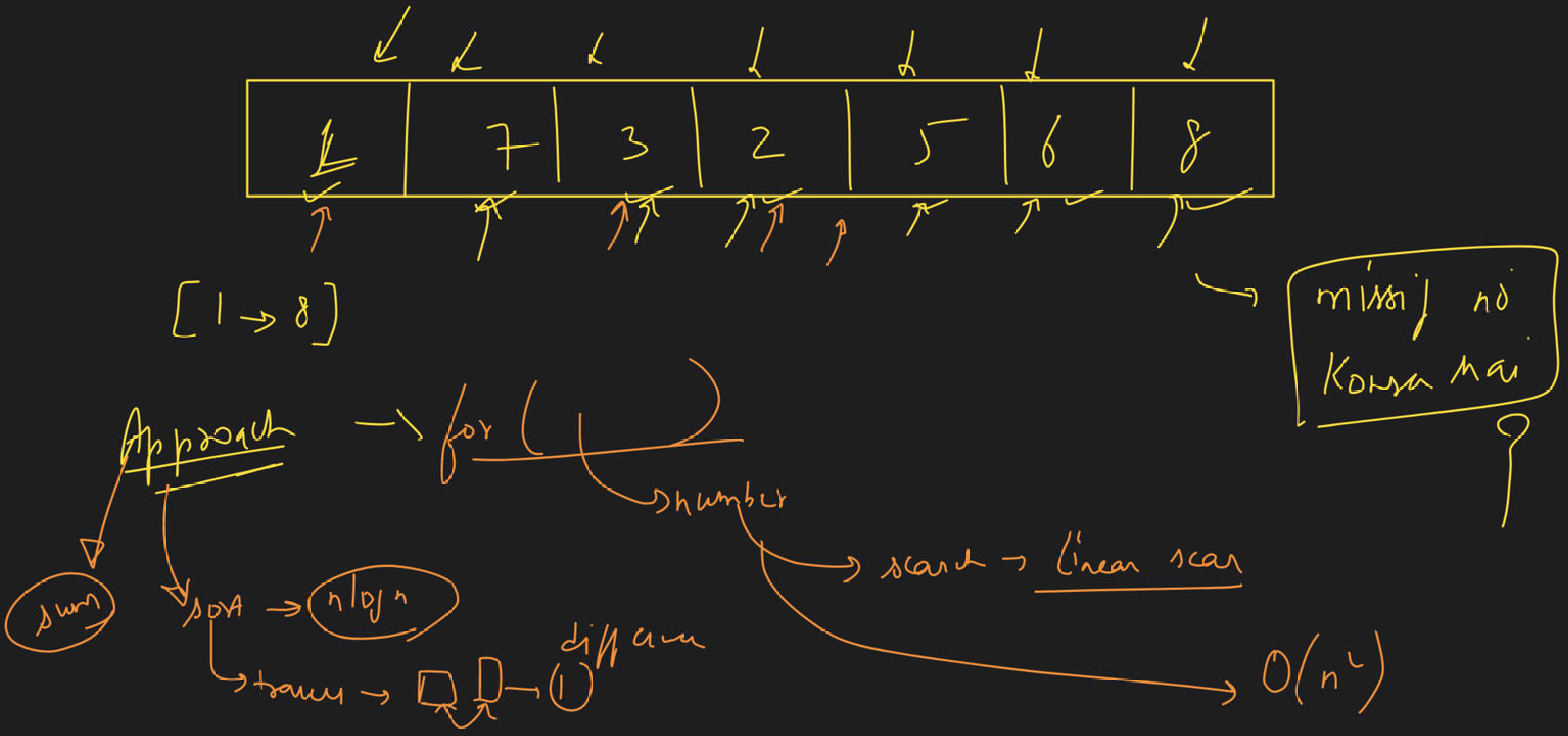
}

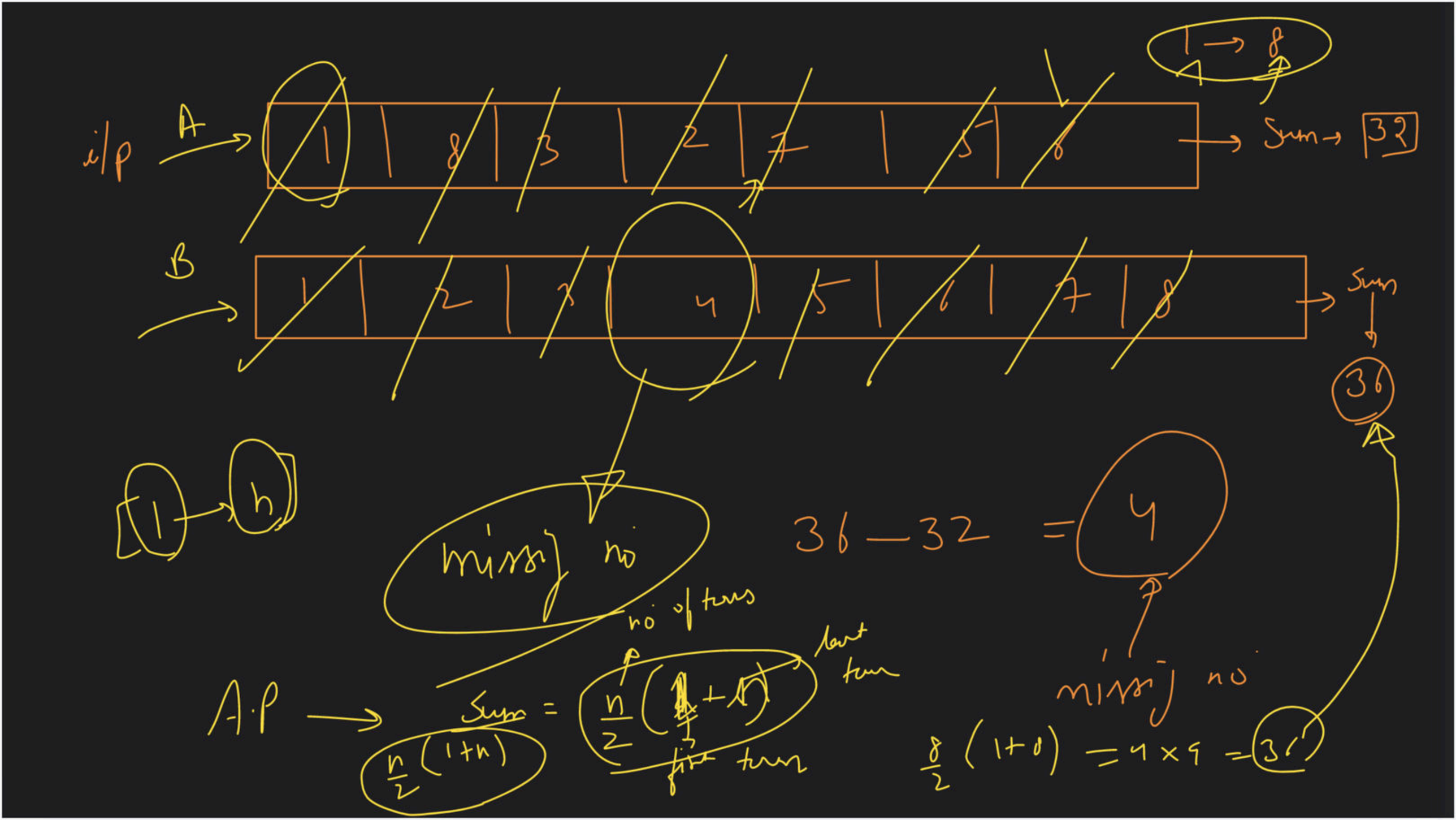
(C) temp → paste → temp arr

O(1) → ?

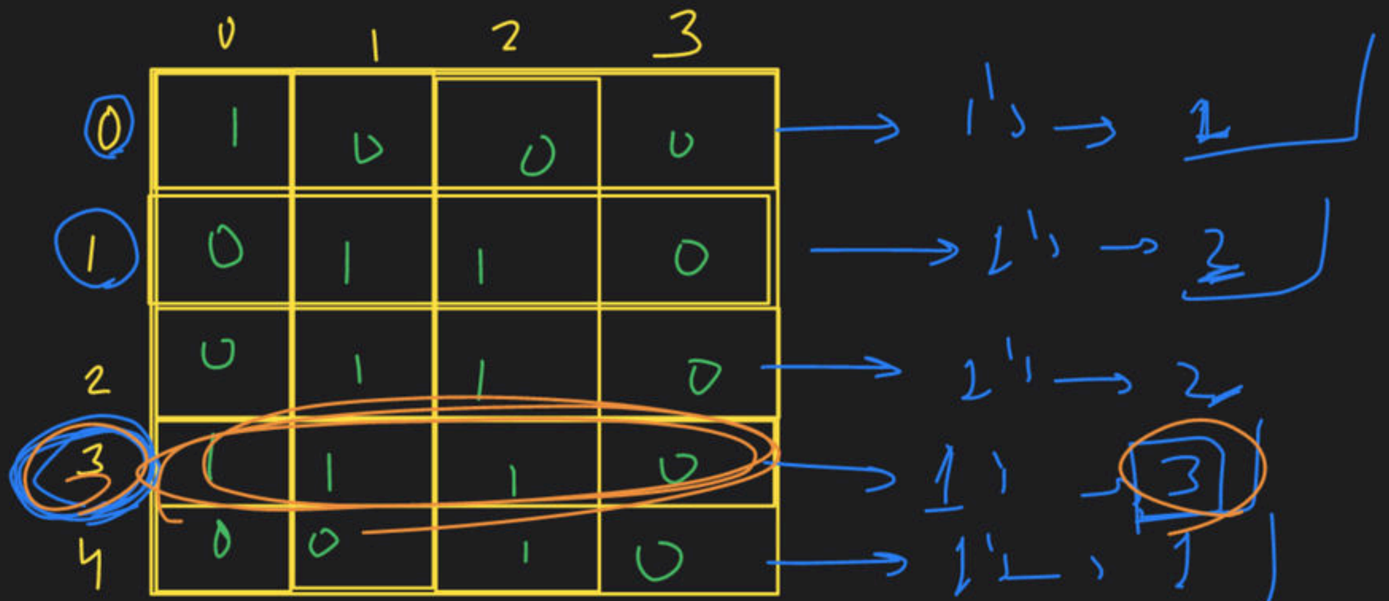

index
nums
50
60
10
20
30
40
0
1
2
3
4
5
K = 2
n = 6
ans
50
60
10
20
30
40
0
1
2
3
4
5
index = 3
newIndex = (3+2) % 6 = 5
ans[5] = nums[3]
index = 4
newIndex = (4+2) % 6 = 0
ans[0] = nums[4]
index = 5
newIndex = (5+2) % 1 = 1
ans[1] = nums[5]
nums = ans


Coding
11/12
± AS


1 7 3 2 5 6 8
[1 → 8]
missing no
Konsa hai
?
Approach
for
number
search → linear scan
O(n^2)
sum
sort → nlogn
traverse
1
difference


i/p
A
1 | 8 | 3 | 2 | 7 | 1 | 5 | 6
1 → 8
Sum → 32
B
1 | 2 | 3 | 4 | 5 | 6 | 7 | 8
Sum
36
1 → 6
missing no
36 − 32 = 4
missing no
A.P
Sum = n/2 (1st + last)
n/2 (1+n)
no of terms
last term
first term
8/2 (1+8) = 4 × 9 = 36

Max no of 1's wali row



{3, 3}

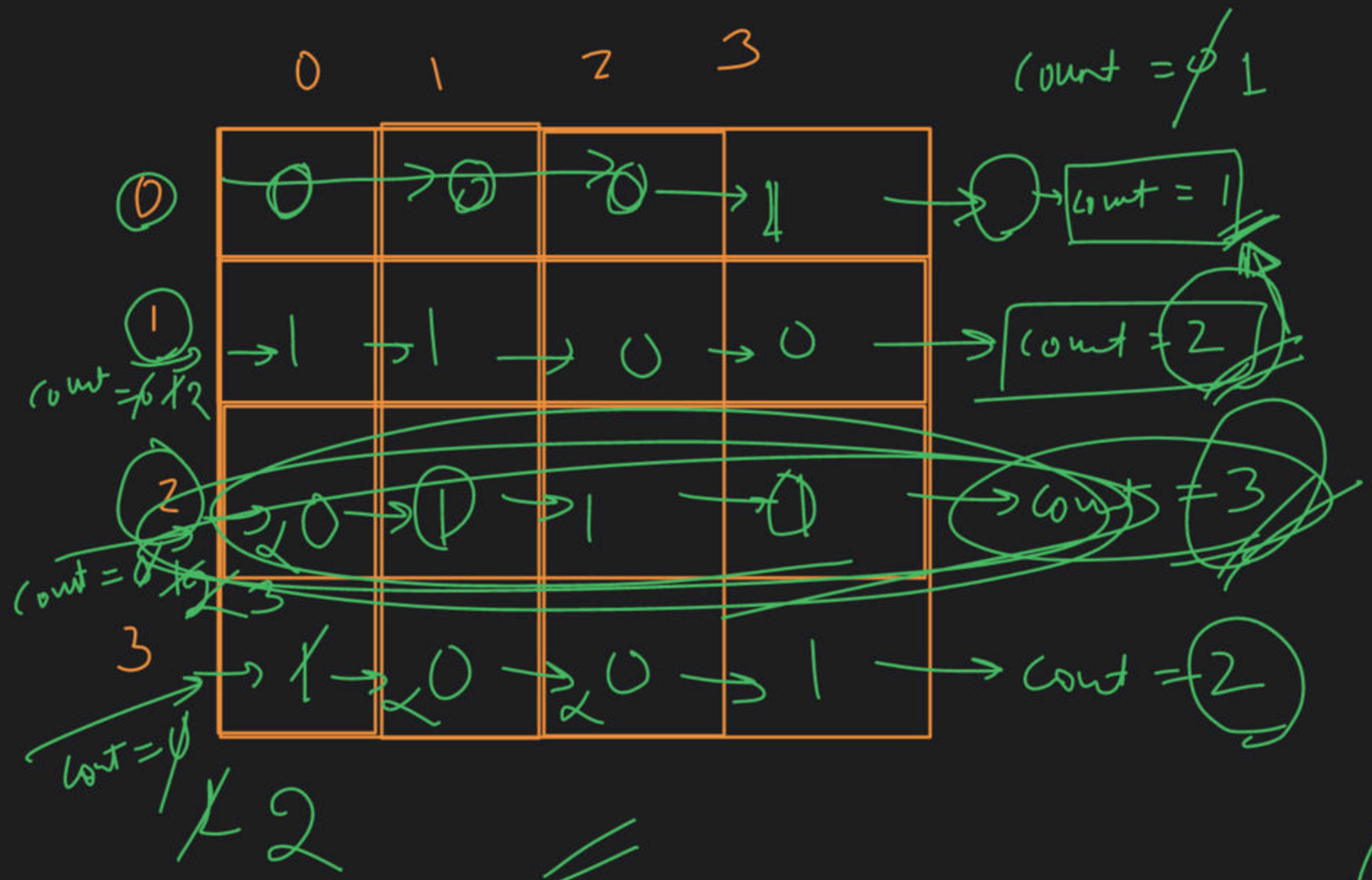
rowNo, Oncount

Oncount = INT_MIN

3

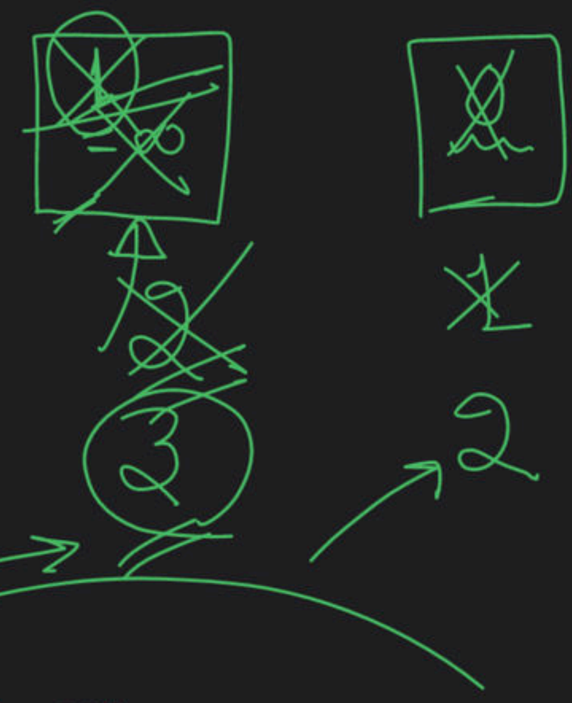
rowNo = -1

Jab bhi naya max Oncount milega, tab rowNo store krenge

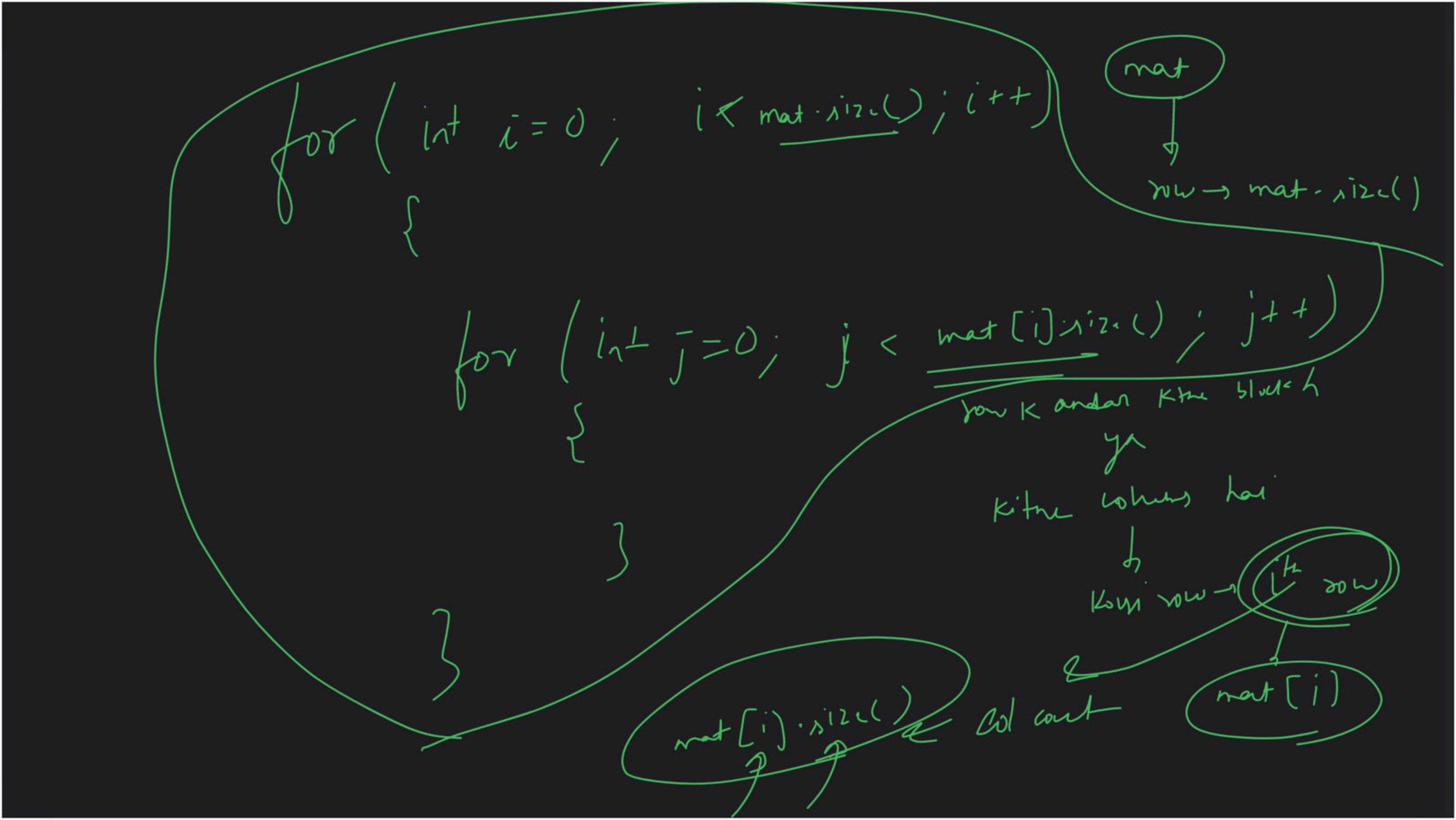


mat[i].size()

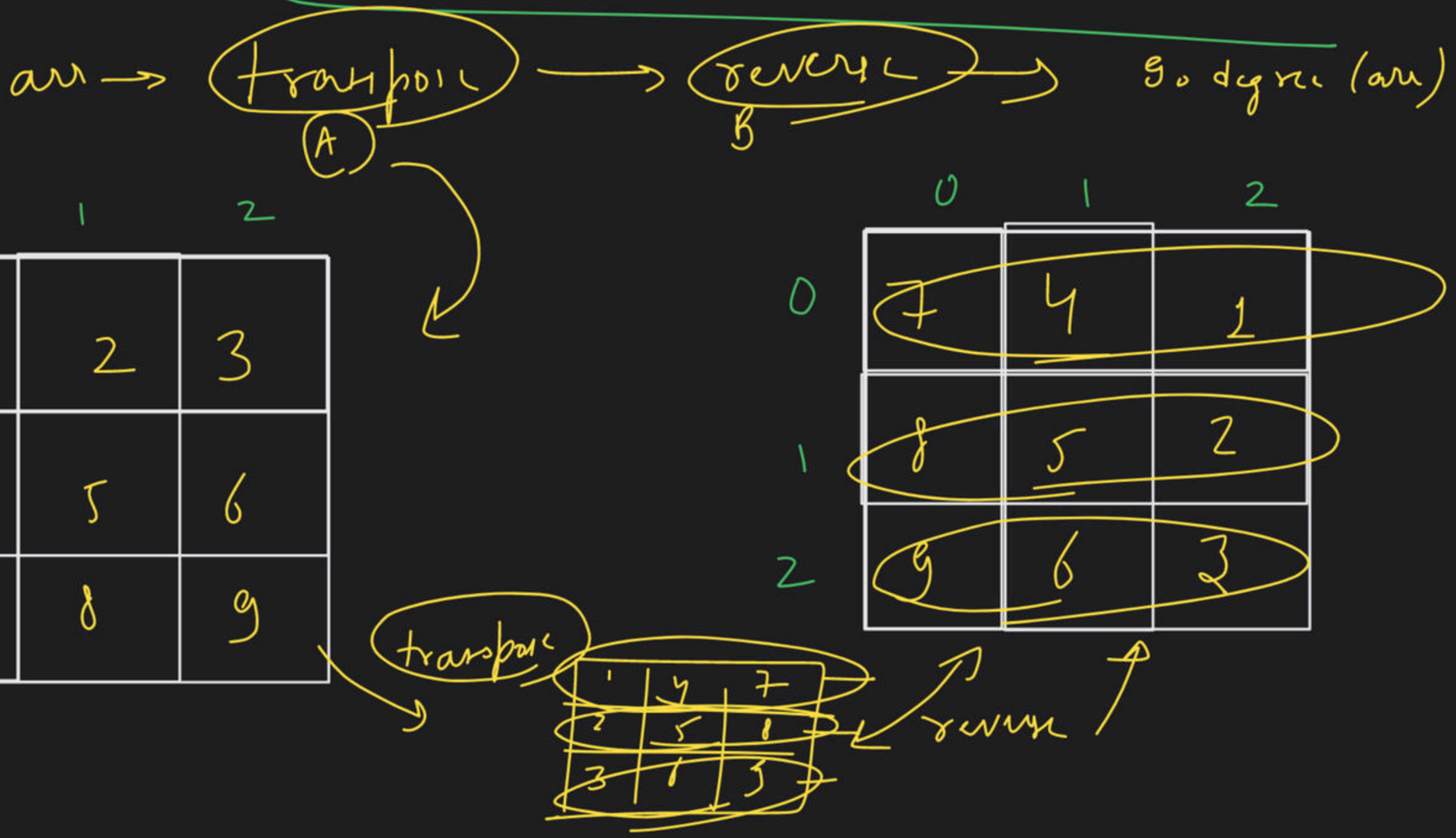
① oneCount ② rowNo



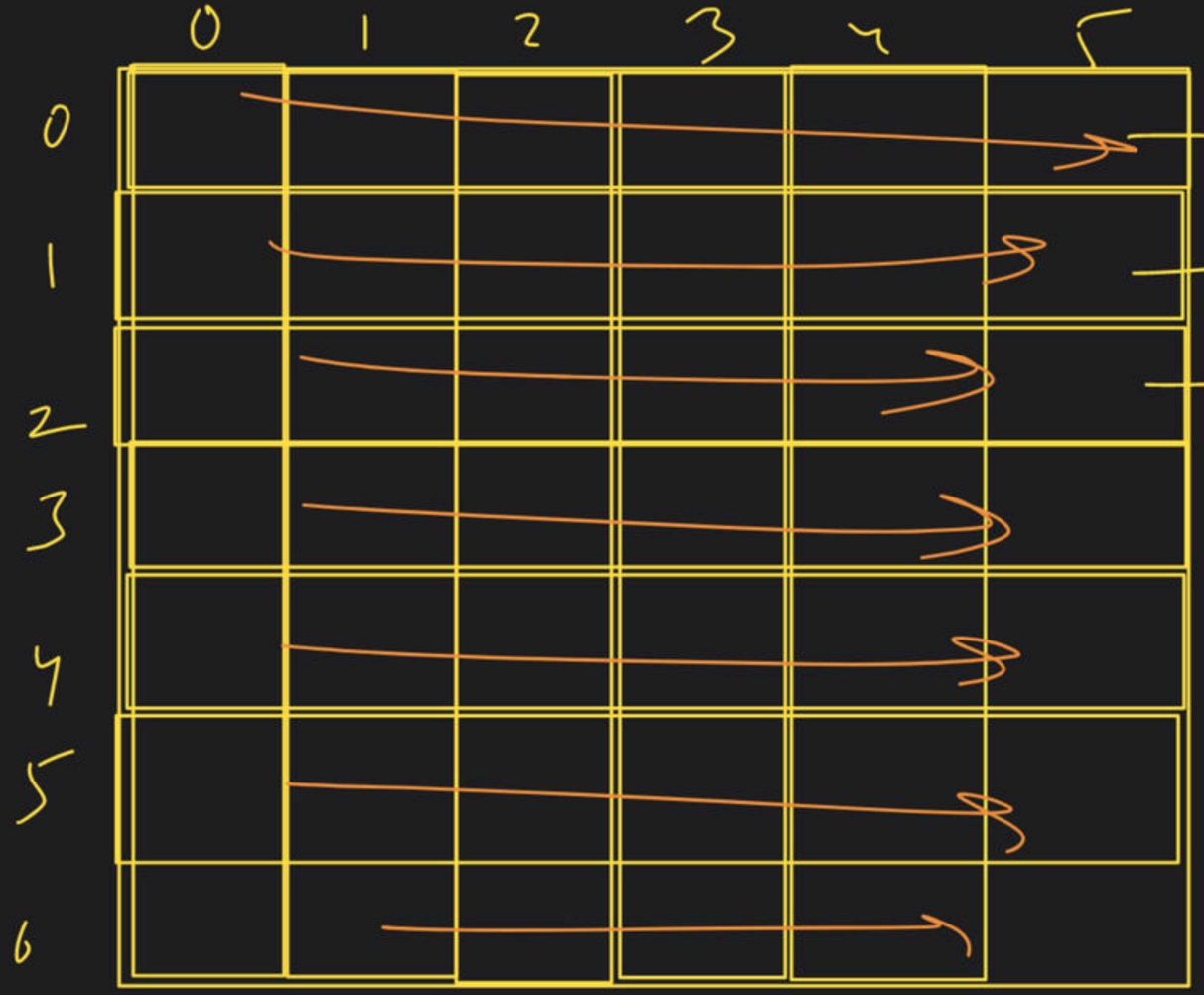
vector<int> ans
ans.pushback(rowNo)
ans.pushback(oneCount)
return ans;


for (int i = 0 ; i < mat.size() ; i++)
{
for (int j=0 ; j < mat[i].size() ; j++)
{
}
}
mat
row → mat.size()
row k andar kitne block h
ya
kitne columns hai
kisi row → ith row
mat[i]
mat[i].size()
col count

→ Rotate a 2D Array by 90 degree



matrix



row number → matrix(0)

→ → matrix[1]

→ row → matrix[2]

i^{th} row →

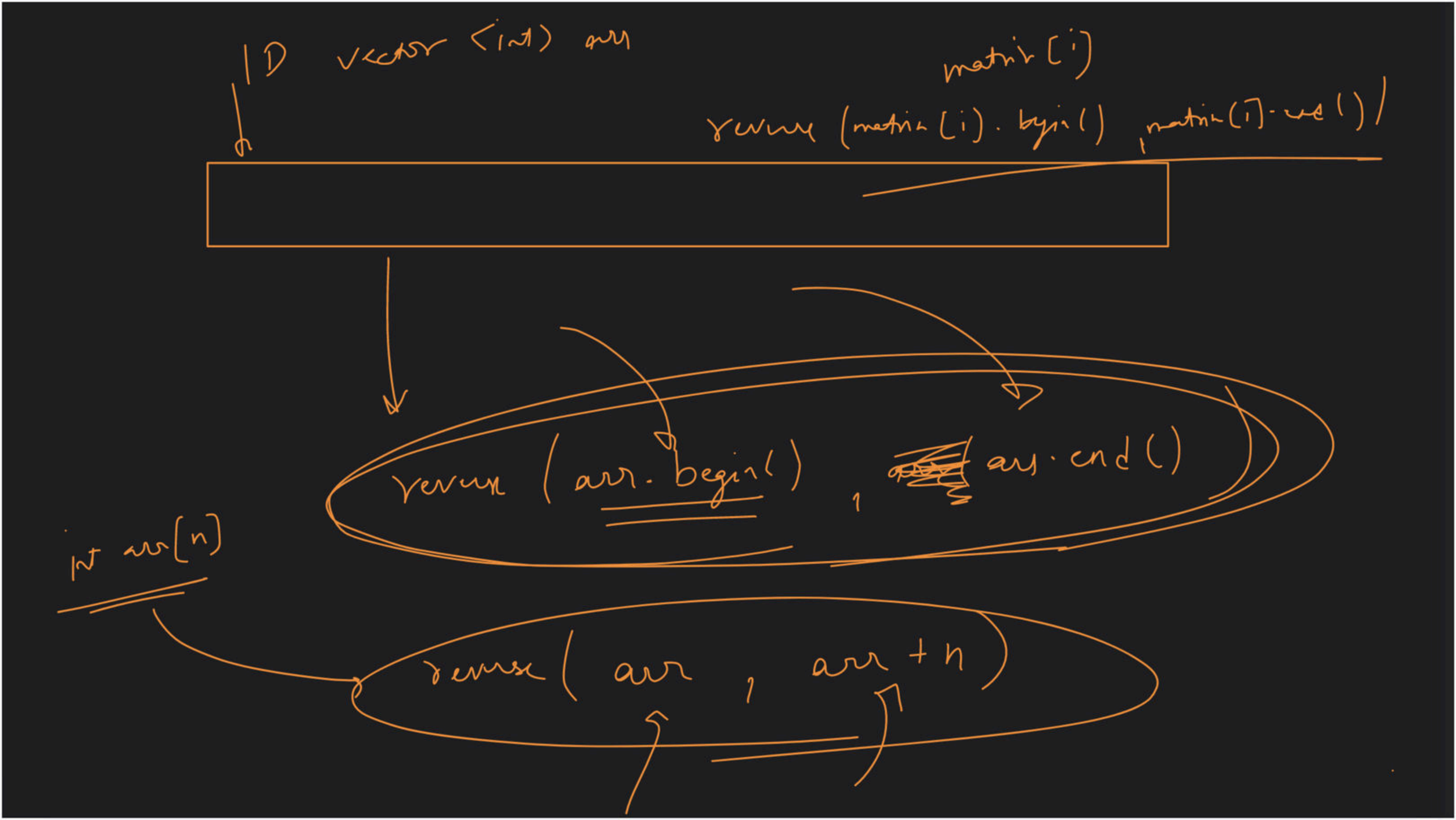
matrix[i]

for (i=0 → i<n)

{

recurse → matrix[i]

}


1D vector<int> arr
matrix[i]
reverse(matrix[i].begin(), matrix[i].end());
reverse(arr.begin(), arr.end())
int arr[n]
reverse(arr, arr+n)