Searching and Sorting Class - 3

Special class

Love Babbar • May 30, 2024

i/p →

Divisor = 2

dividend = 10

Divisor

Dividend

Quotient

Remainder

2 * 5 + 0 = 10

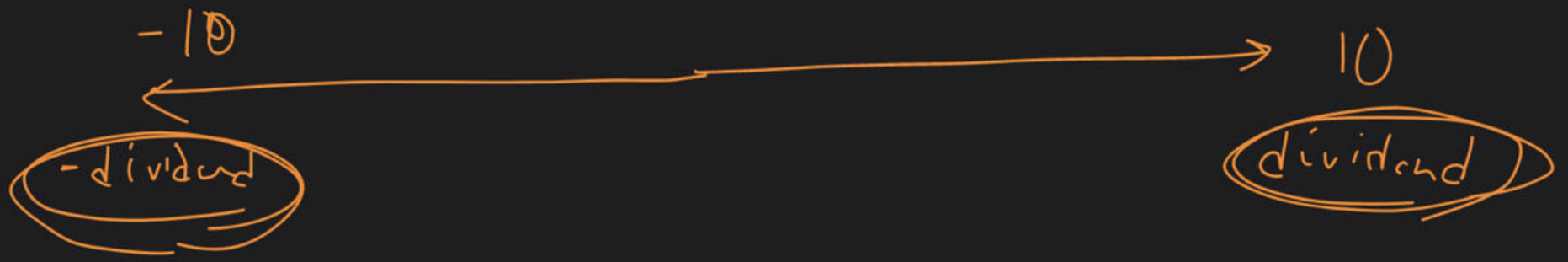
divisor * Quotient + Remainder = Dividend

dividend = 10

divisor = 2

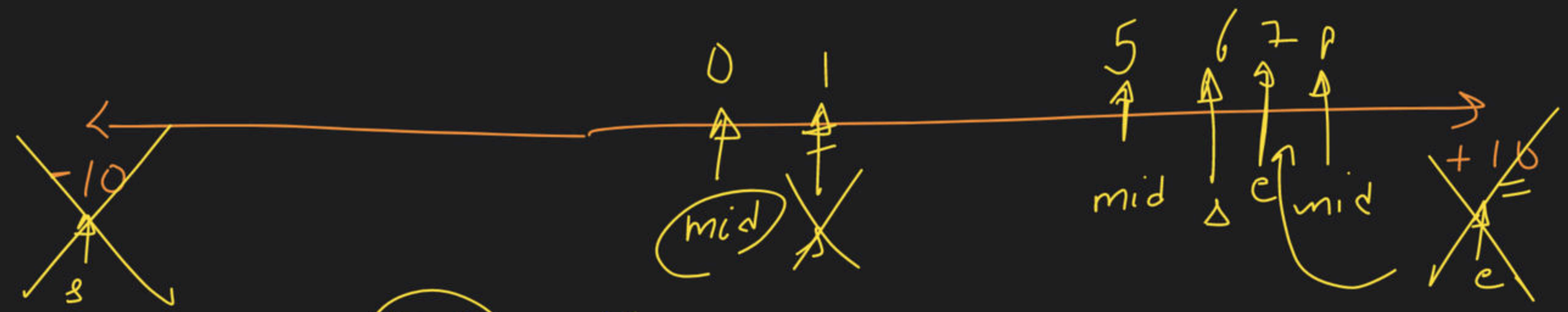
10

-1



dividend = 10 , divisor = 2

Quotient → mid



any

5

mid → 0

divisor * Quotient <= Dividend

2 * 0 <= dividend

0 <= dividend

↳ 0

2 * 5 <= 10

10 <= 10

↳ T

2 * 6 <= 10

12 <= 10

↳ F

2 * 8 <= 10

16 <= 10

↳ F

0, +ve

x + y = dividend

if y = 0

x = dividend

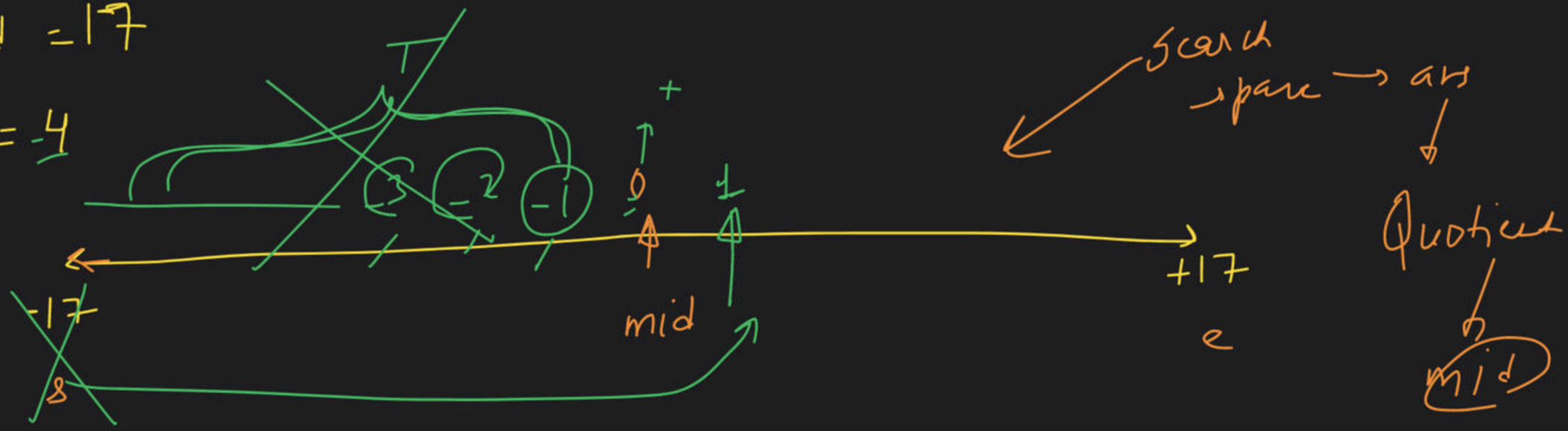
if y = +ve

x <= dividend

x < dividend

dividend = 17

divisor = -4



ans

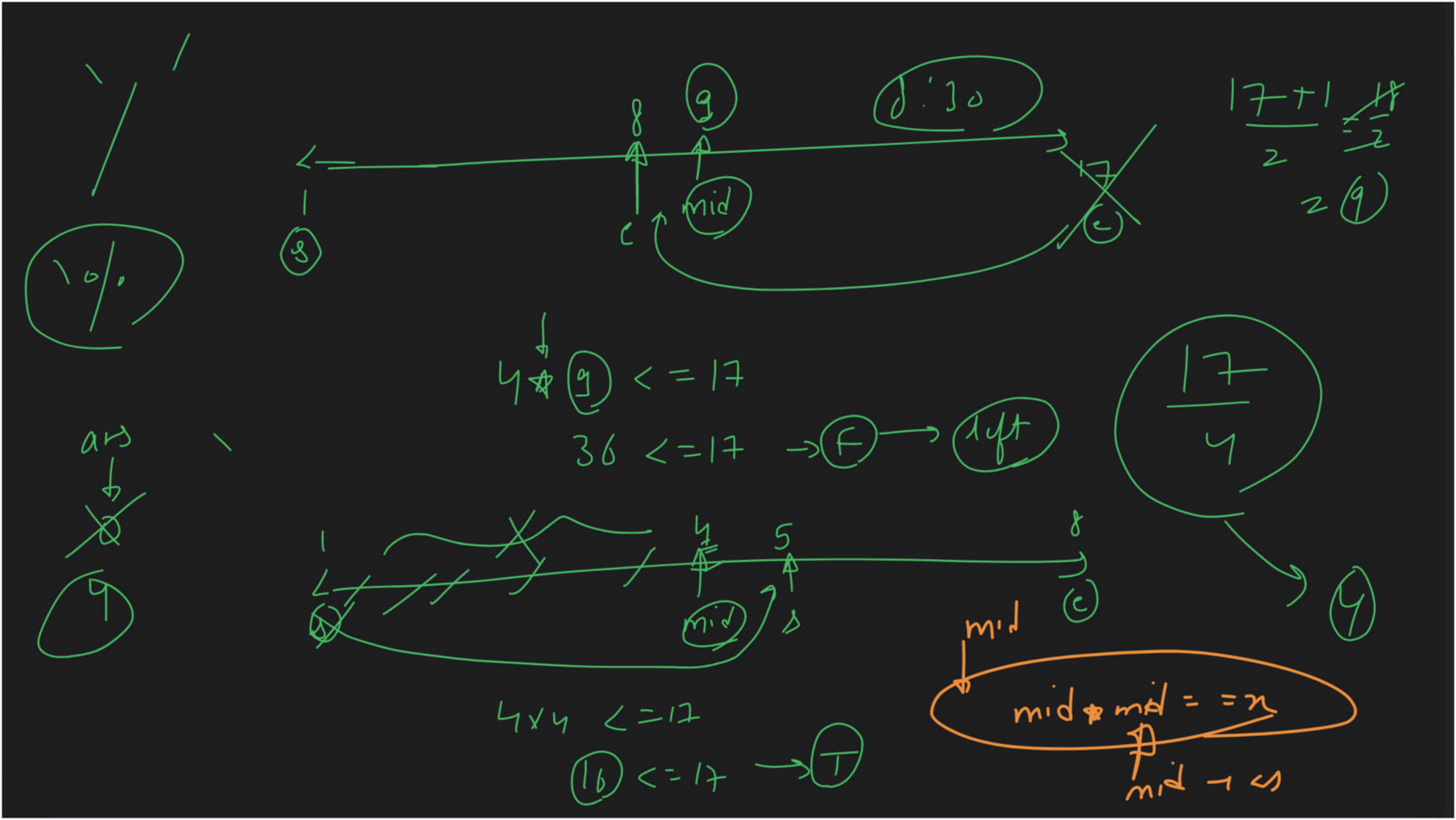
0

divisor * Quotient + ~~rem~~ = dividend

divisor * Quotient <= dividend

4 * 0 <= 17

0 <= 17 → (T)


10/0
8
9
10
mid
17+1/2 = 18/2 = 9
4 * 9 <= 17
36 <= 17
F
left
ans
9
1
4
5
8
mid
4 x 4 <= 17
16 <= 17
T
17/4
4
mid
mid * mid = = n
mid - 1

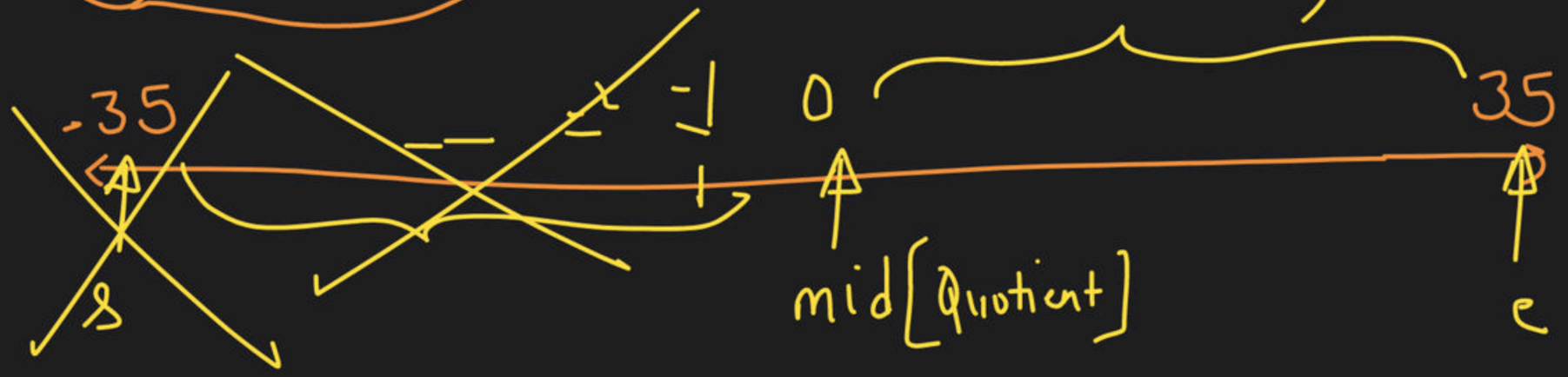
dividend = 35

divisor = 4

divisor * Quotient + Remainder = dividend

! 0

divisor * Quotient <= dividend



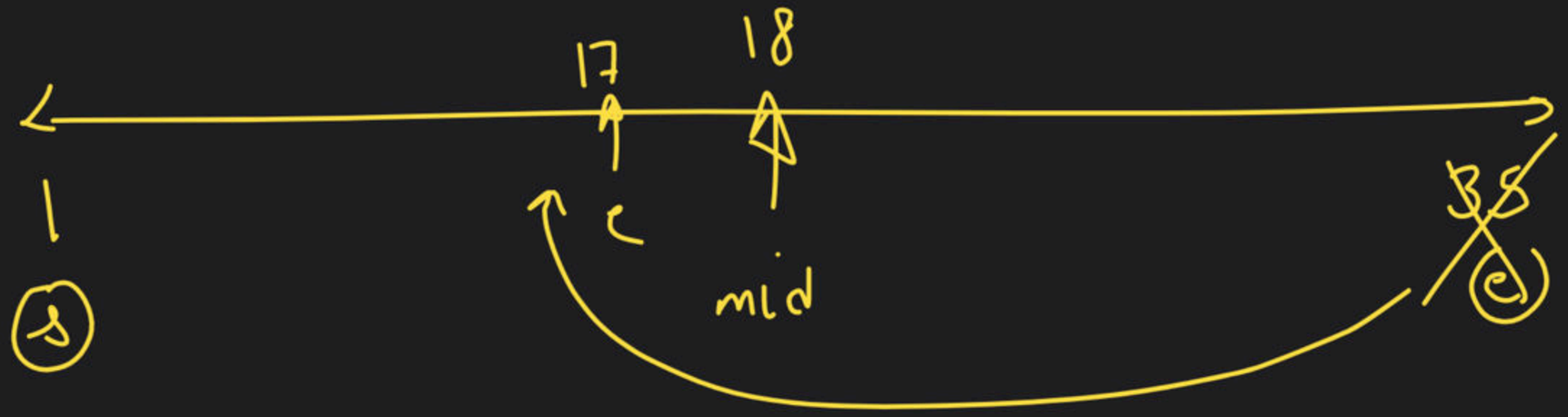
ans

0

4 * 0 = 0 =< 35

0 < 35

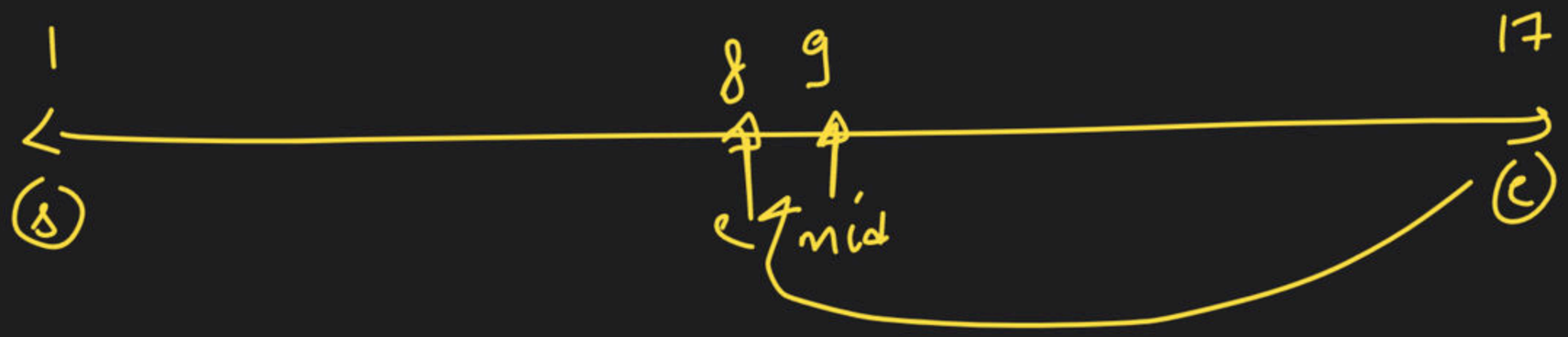
s = mid + 1



ans
↓
0

$4 \times 18 = 72 = \alpha^{=35}$

72 > 35 → left



$4 * 9 = 36 = 35$

$\geq$ → left → e = mid - 1

$4 * 4 = 16 = = 35$

$\leq \rightarrow s \ell c$

$\rightarrow mid+1$

$s = mid + 1$

4 * 6 = = 24 = = 35

ans

→ sbc

↳ right

4 * 7 = 28 < 35

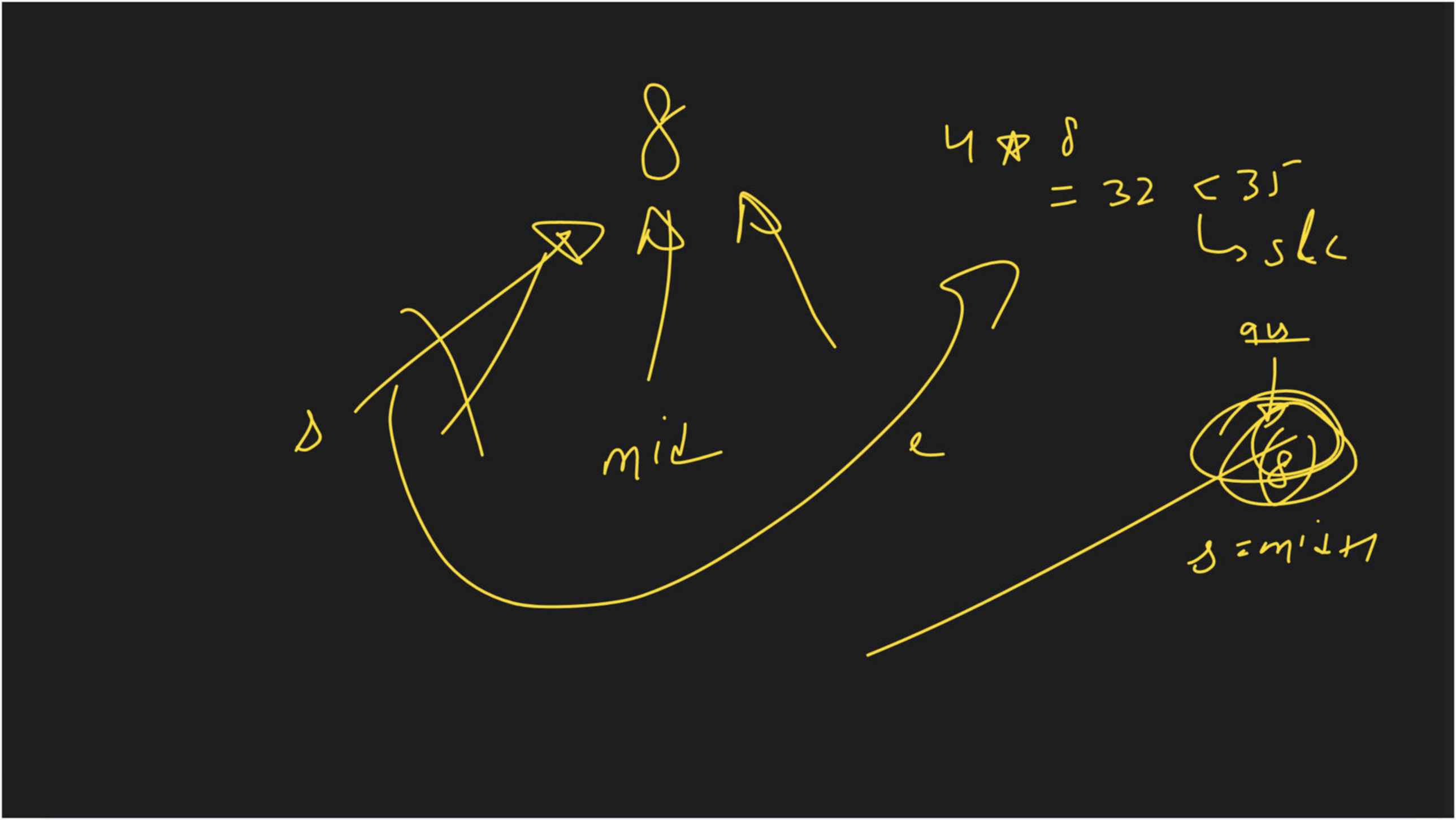
↳ s & c

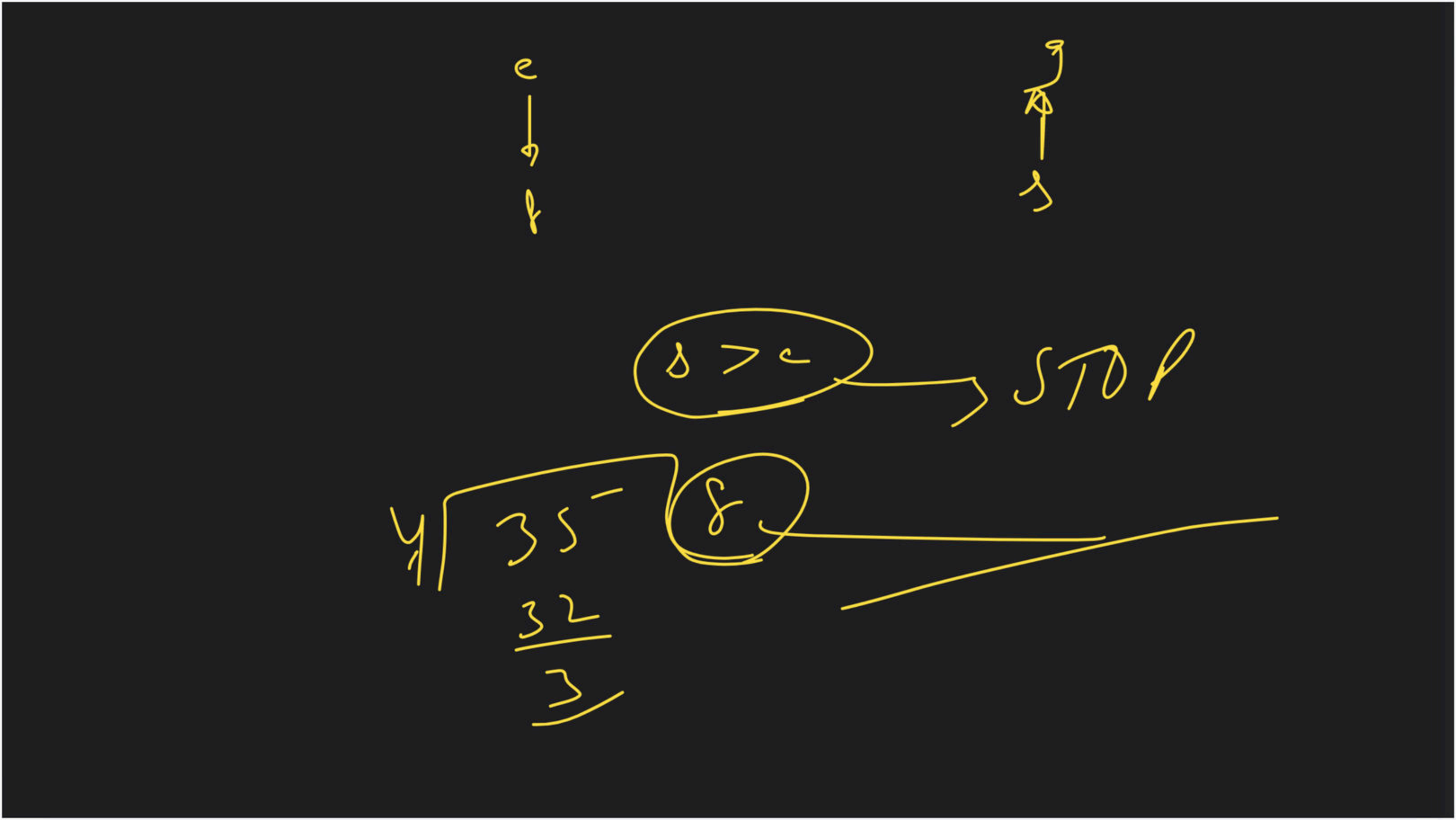
↳ s = mid + 1

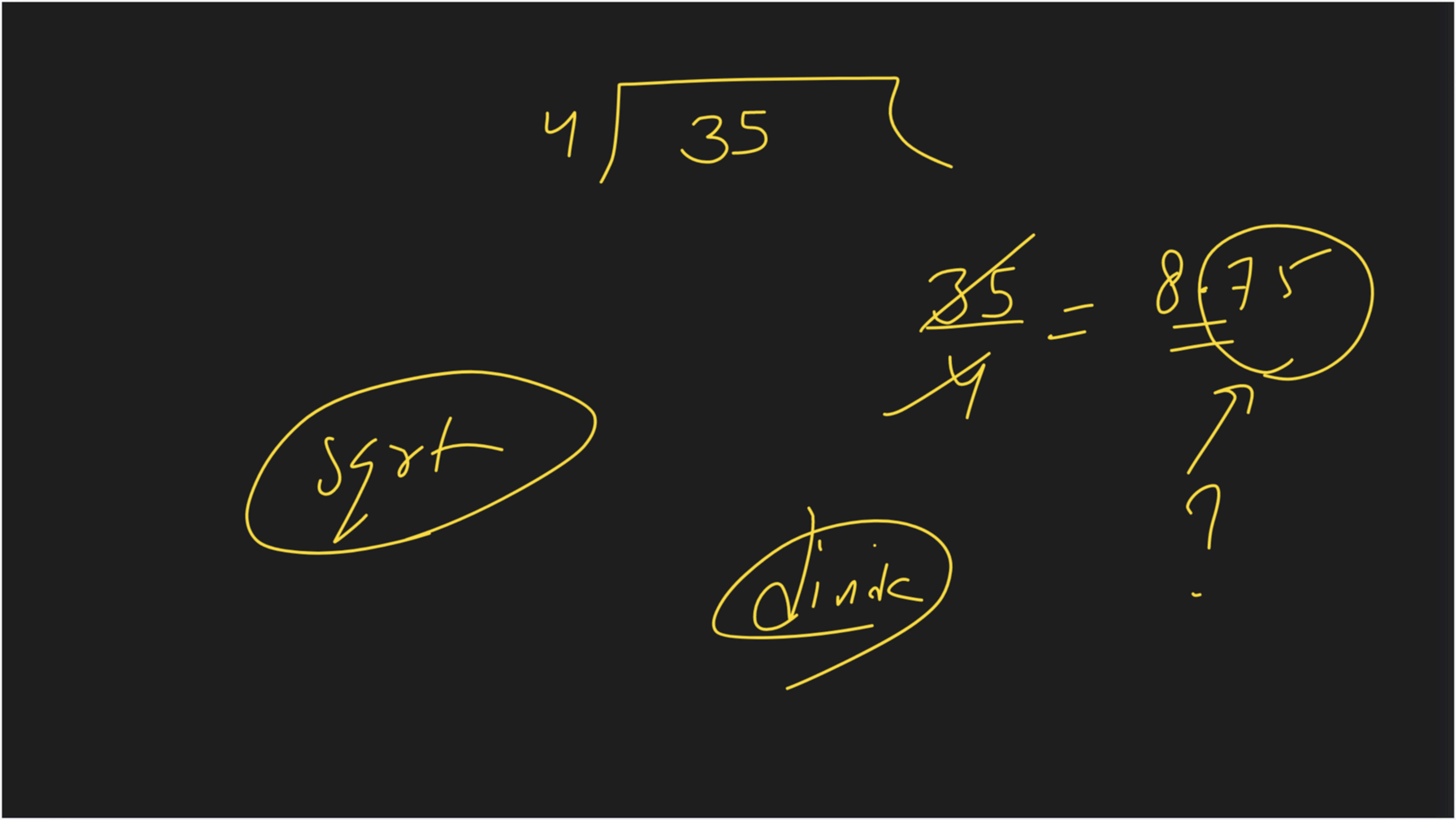
qb

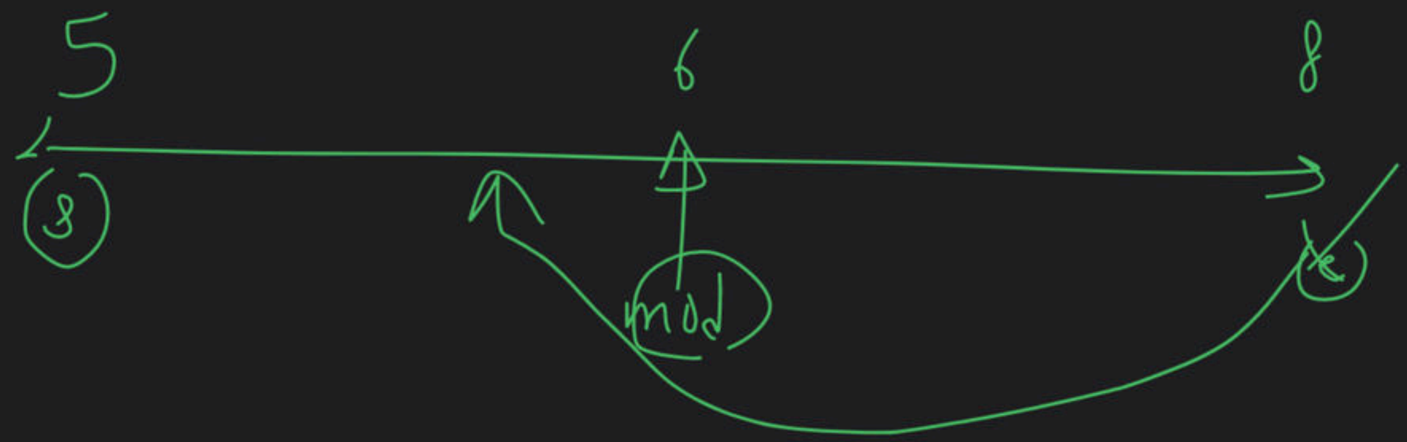
↓

7


8
4 * 8
= 32 < 35
s
mid
e
qu
s = mid+1


e
f
g
s
s > c
STOP
4 | 35
8
32
3


4 | 35
35/4 = 8.75
sqrt
divide



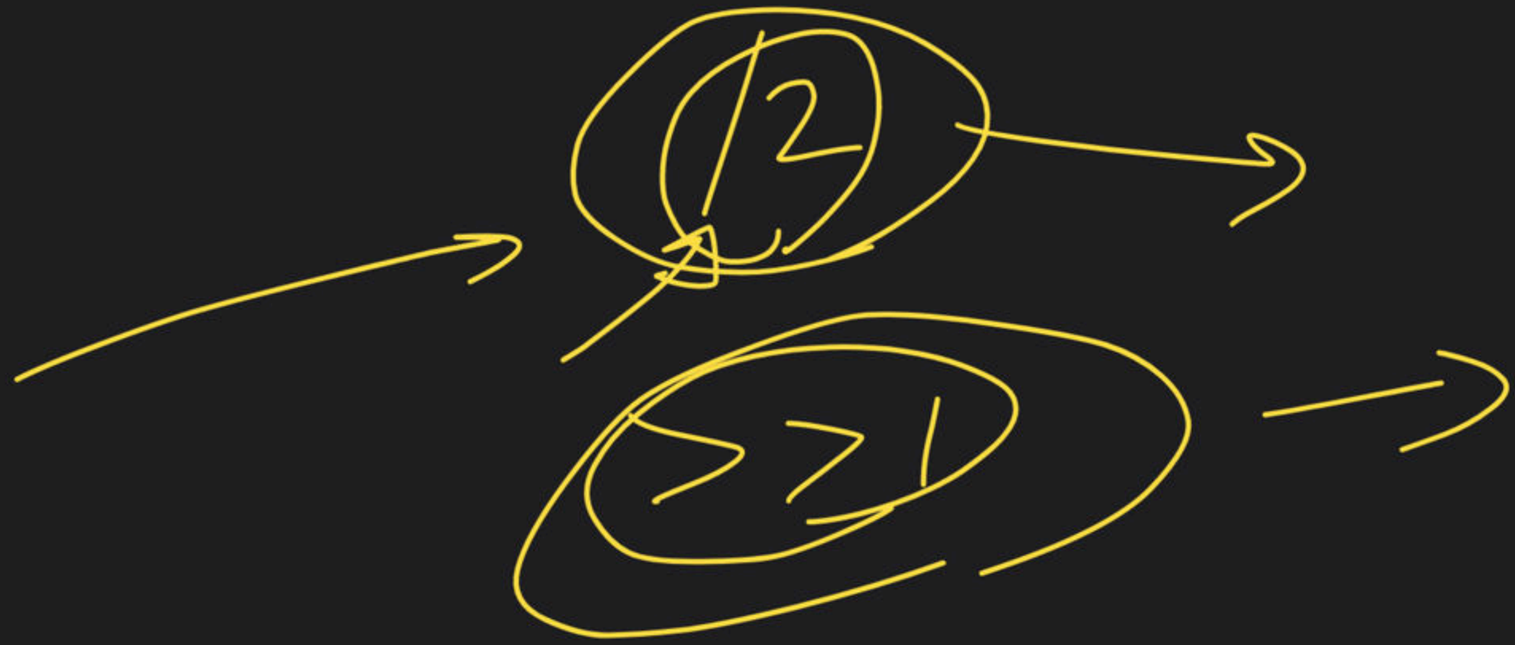
$4 * (6) <= 17$

$24 <= 17 \rightarrow$ (F)

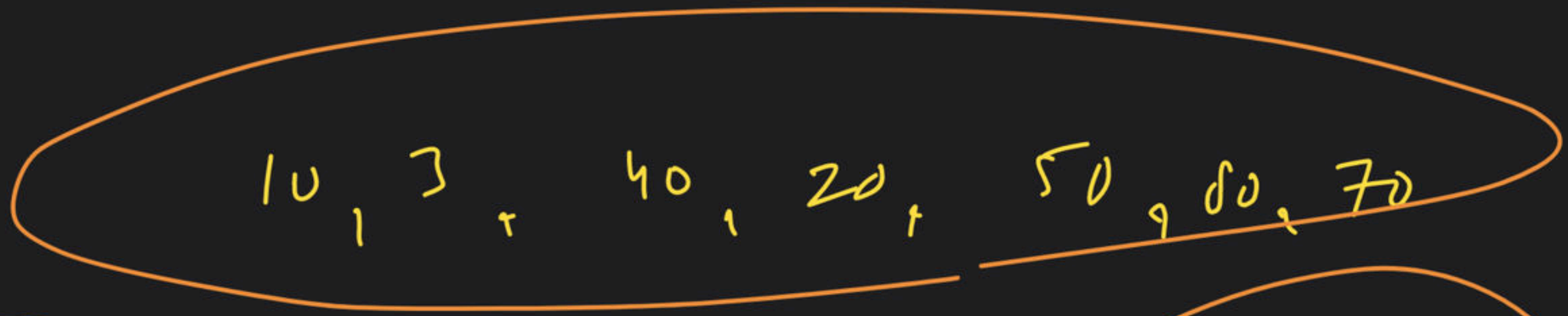
$4 * 5 <= 17$

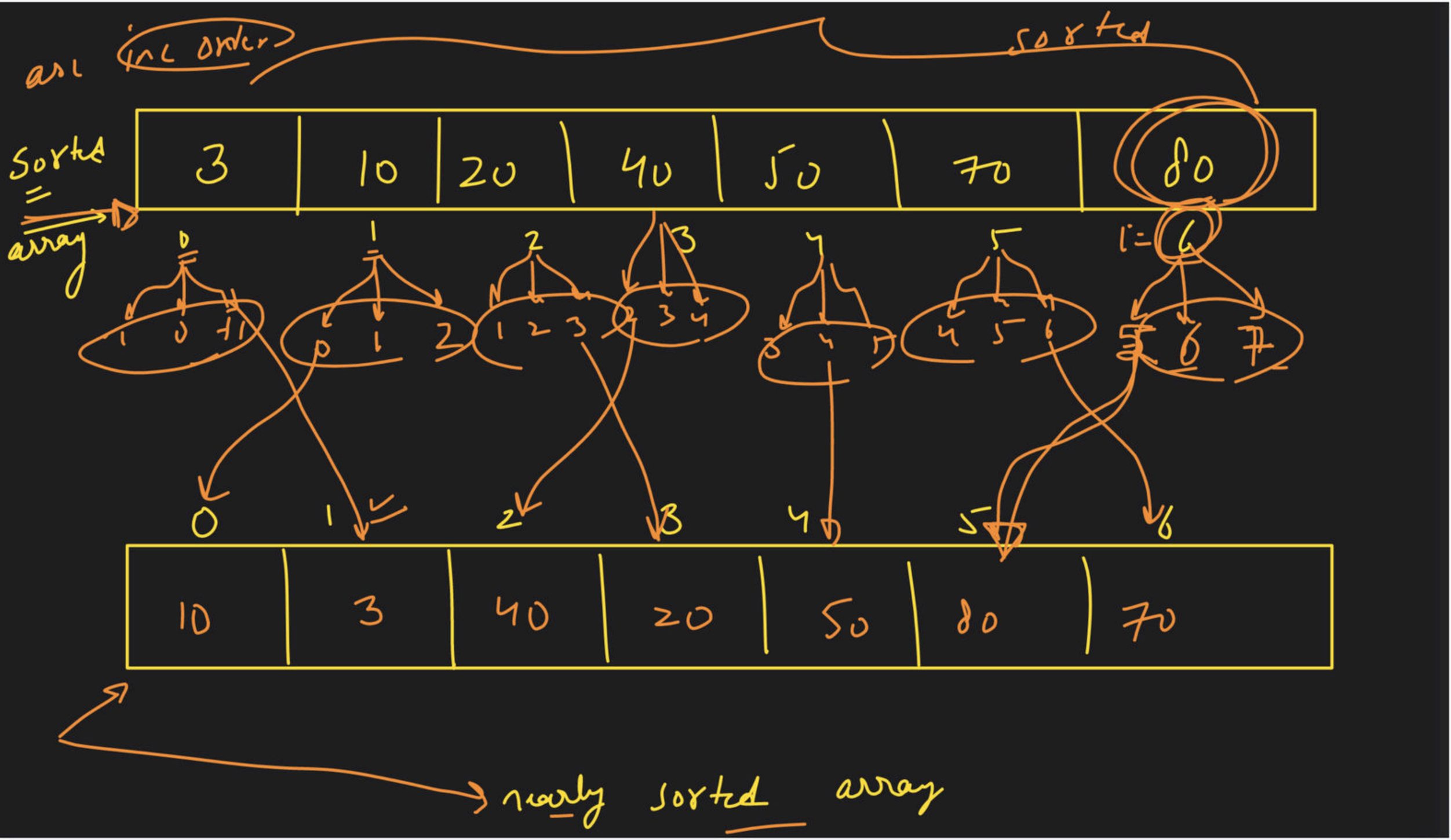
$20 <= 17 \rightarrow$ (F) $\rightarrow$ 반복

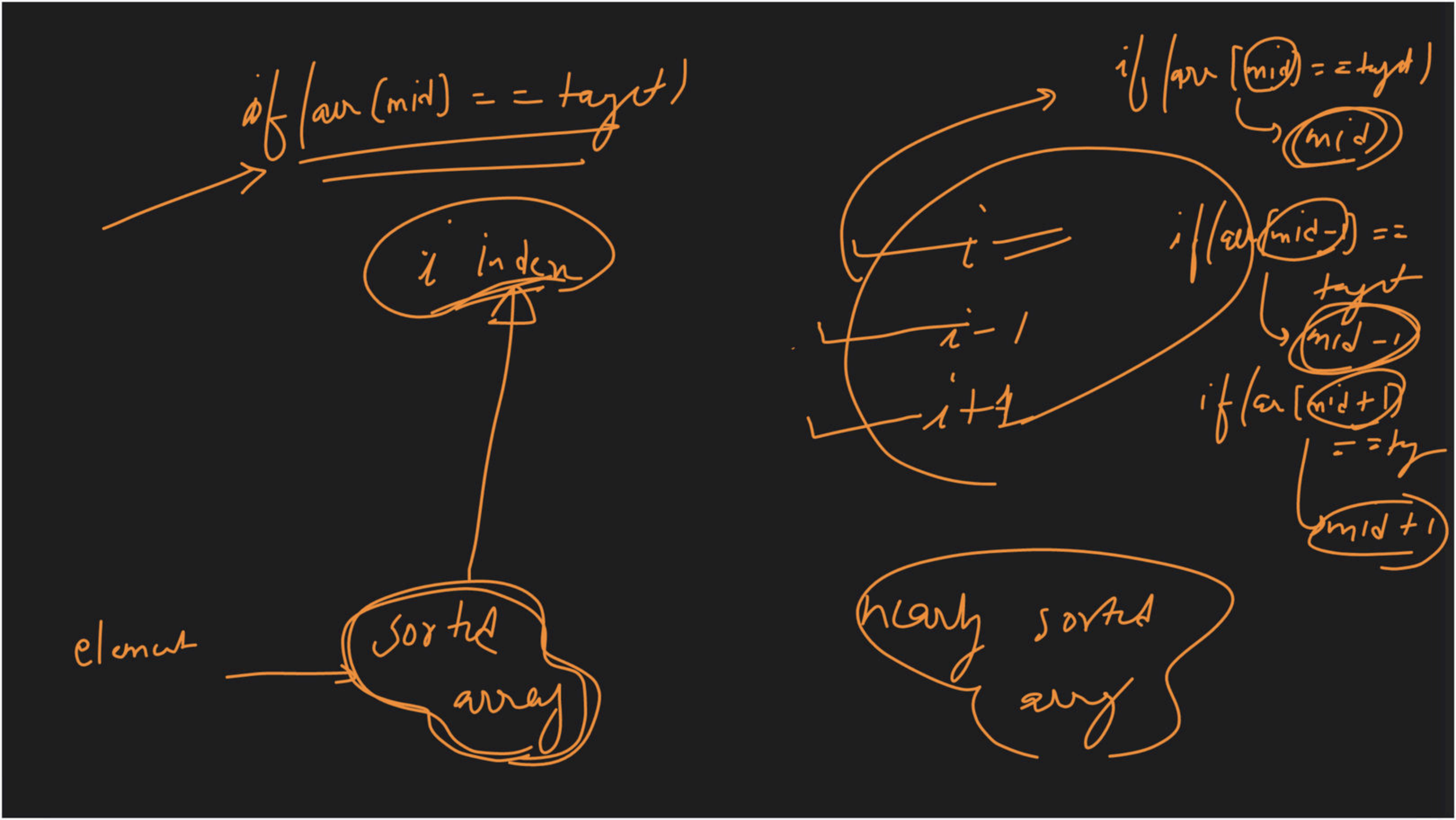
$c = 4$ $s = 5 \rightarrow s > c \rightarrow$ 종료

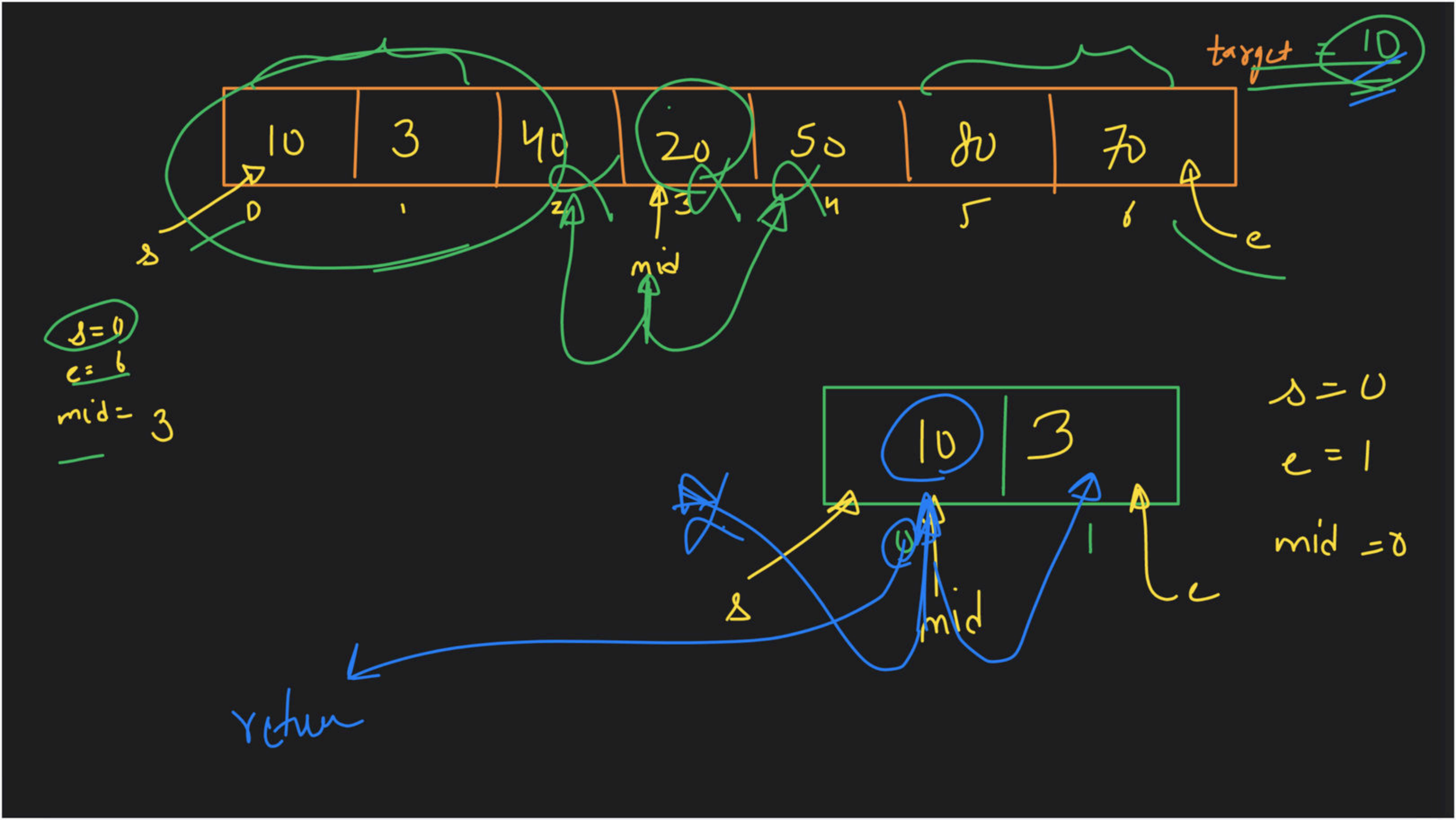

2
>>1

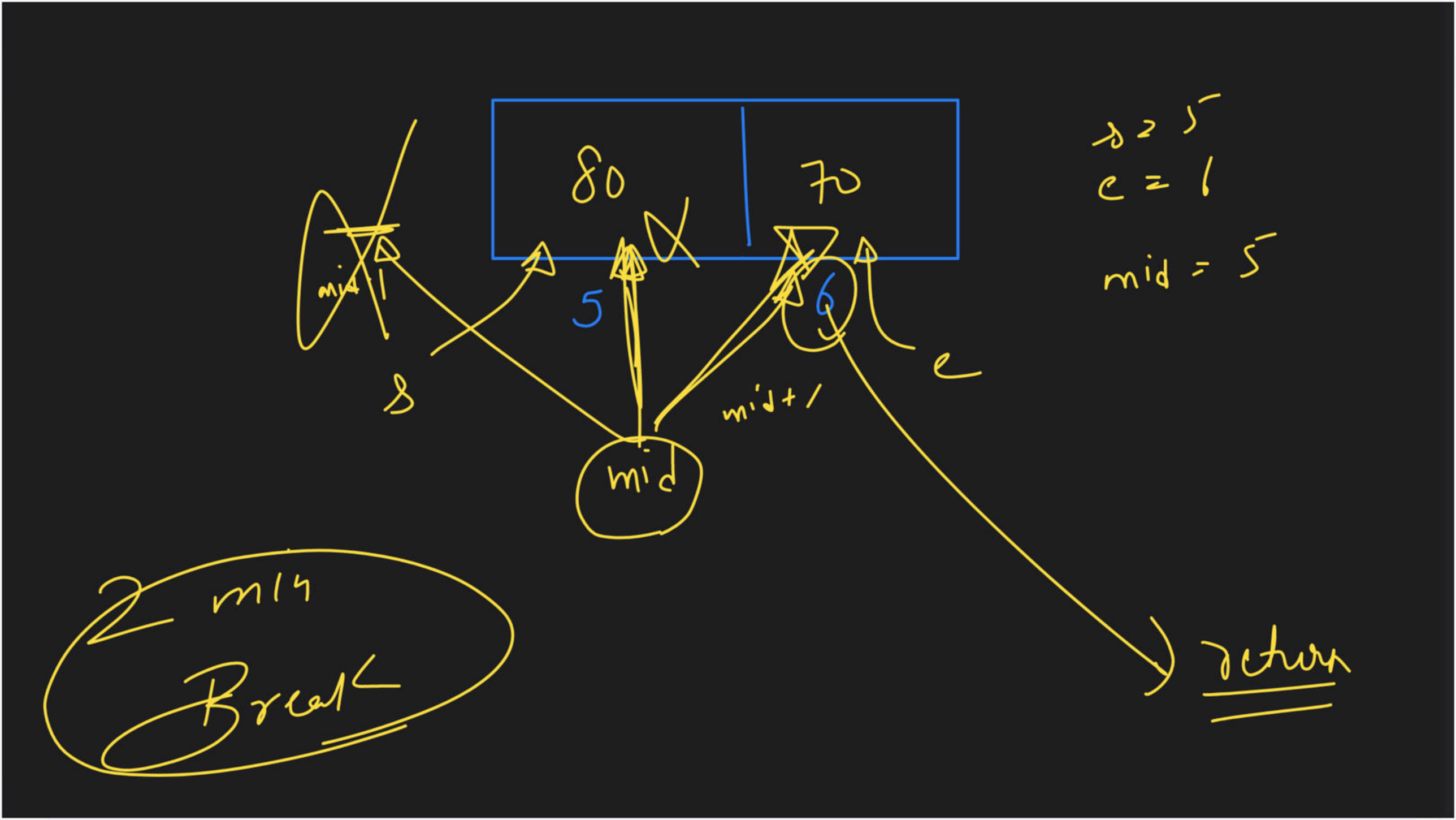
→ Search in a nearly / almost sorted array

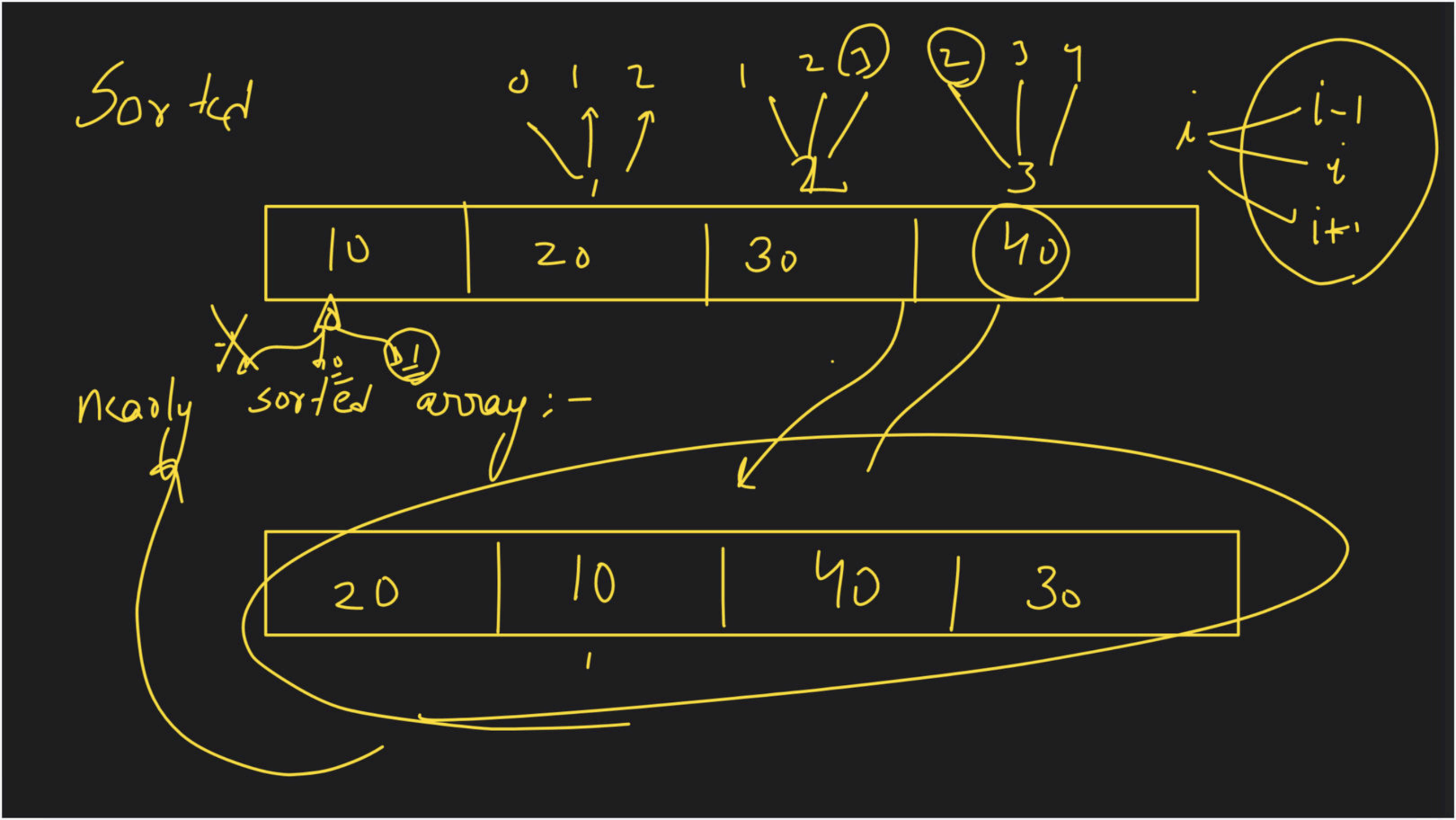



asc
inc order
sorted
SortA
=
array
3
10
20
40
50
70
80
i=6
0
1
2
3
4
5
6
7
10
3
40
20
50
80
70
nearly sorted array


if (arr (mid) == target)
i index
element
sorted array
if (arr [mid] == target)
mid
i ==
i - 1
i + 1
if (arr [mid - 1] ==
target
mid - 1
if (arr [mid + 1]
== tg
mid + 1
nearly sorted
arr


target = 10
10
3
40
20
50
80
70
0
1
2
3
4
5
6
s
e
mid
s = 0
e = 6
mid = 3
10
3
0
1
s
e
mid
s = 0
e = 1
mid = 0
return


80
70
5
6
mid+1
s
mid
e
s = 5
e = 1
mid = 5
2 min
Break
return


Sorted
0 1 2
1 2 3
2 3 4
1
2
3
i
i-1
i
i+1
10
20
30
40
0
1
nearly sorted array :-
20
10
40
30
1

Odd occuring element :-

sort() → O(n log n)

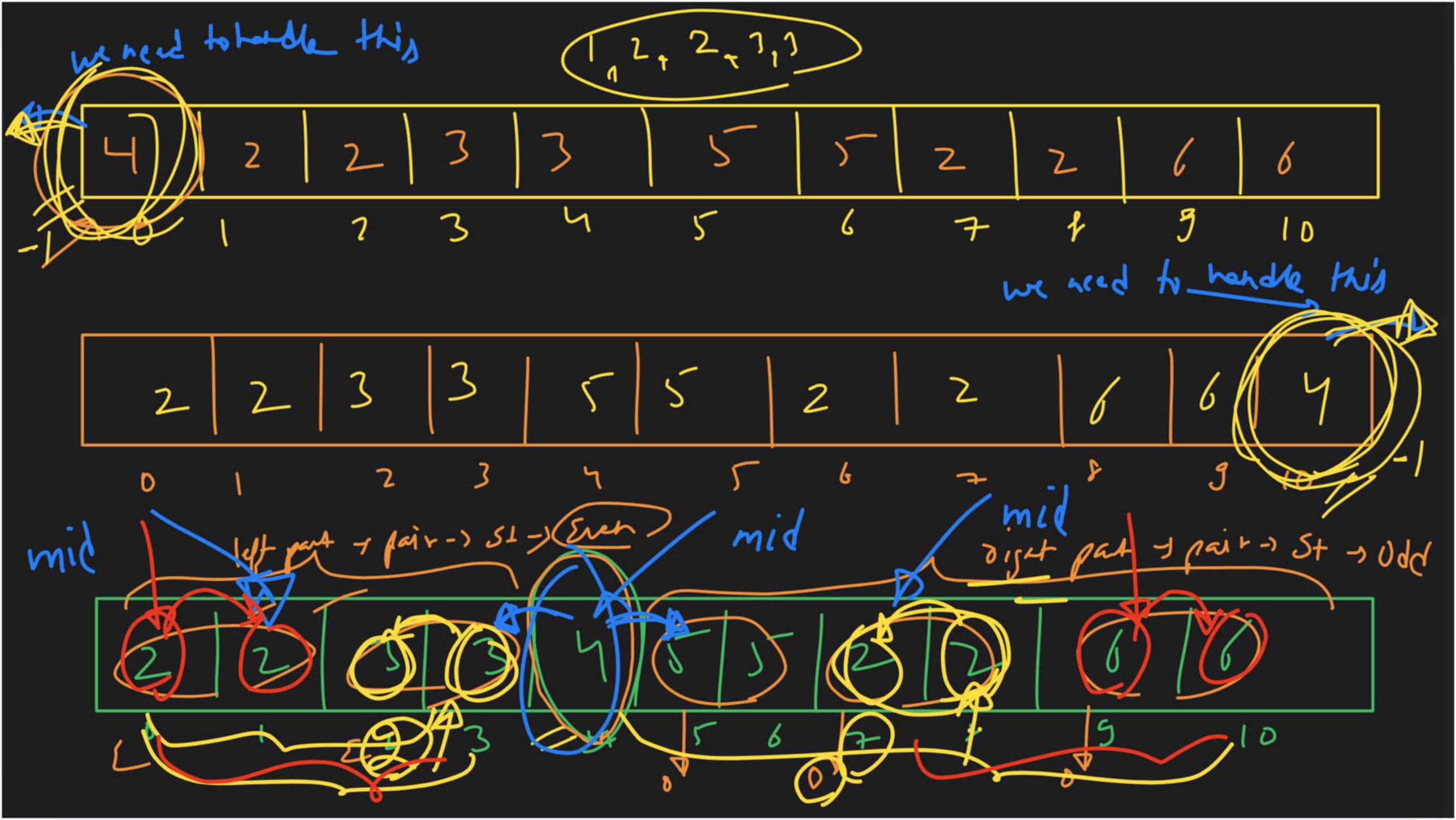
O(log n) → B.S

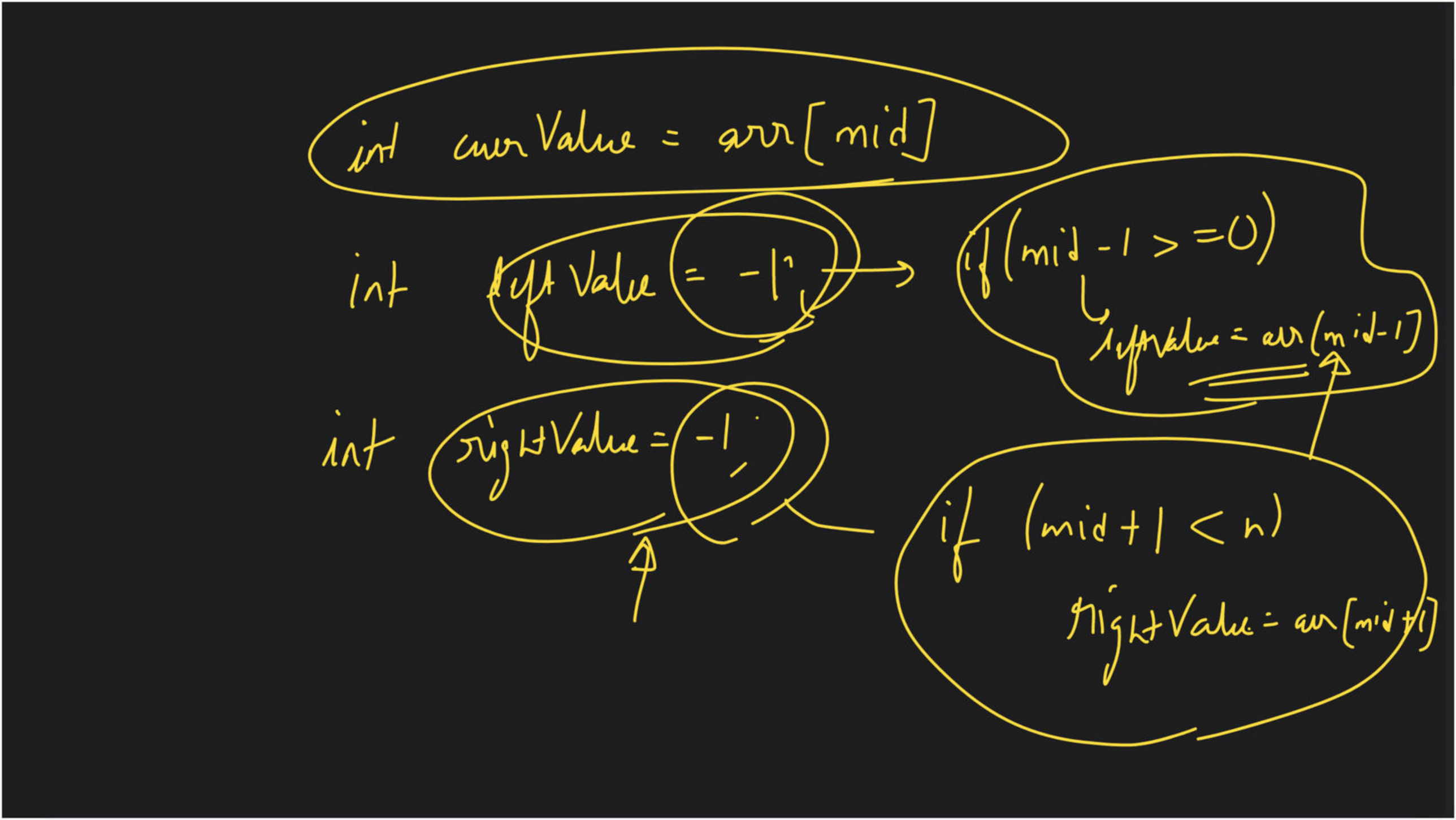
(A) all element occurs even number of times except one

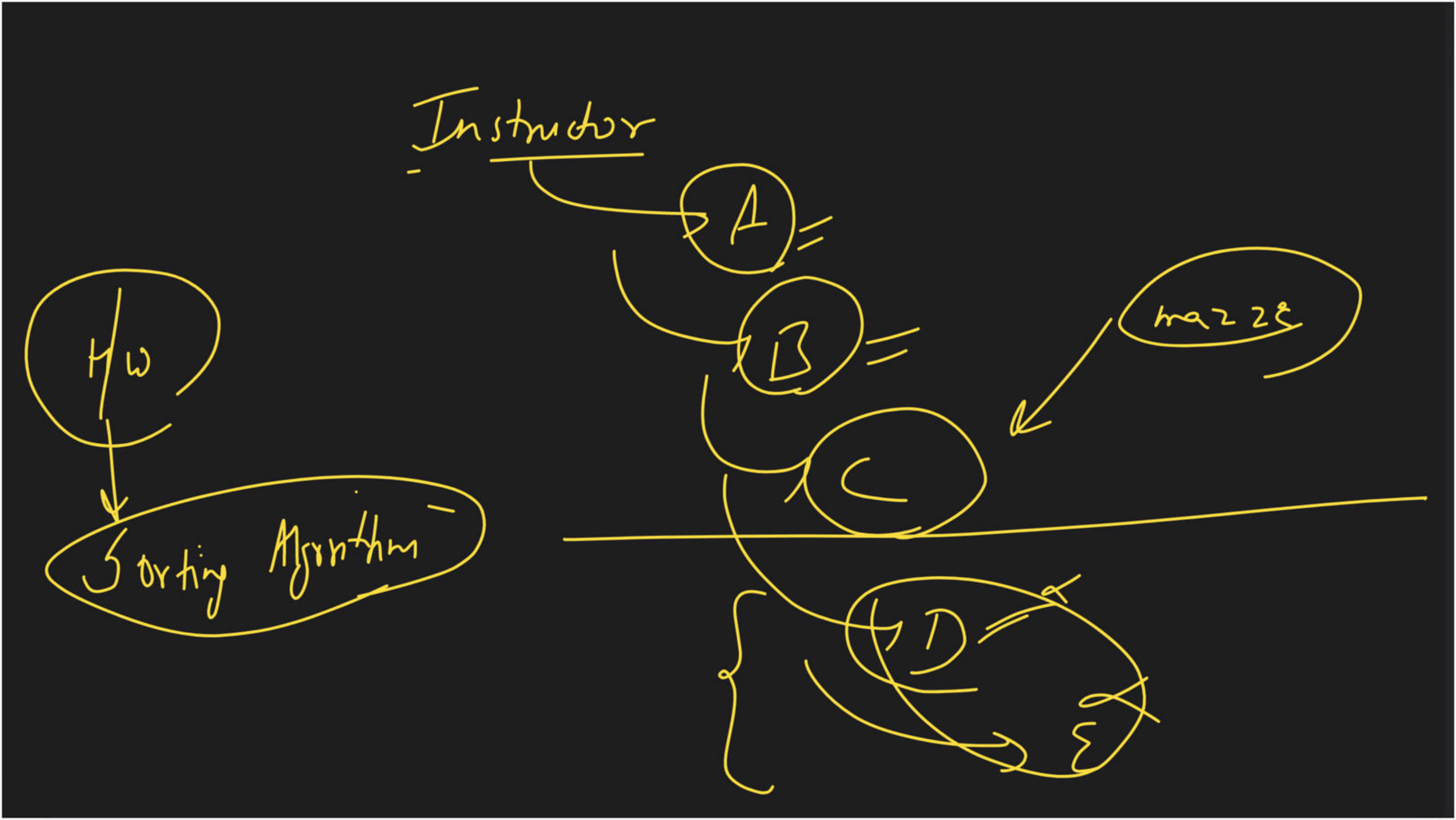
(B) element repeats itself in pairs

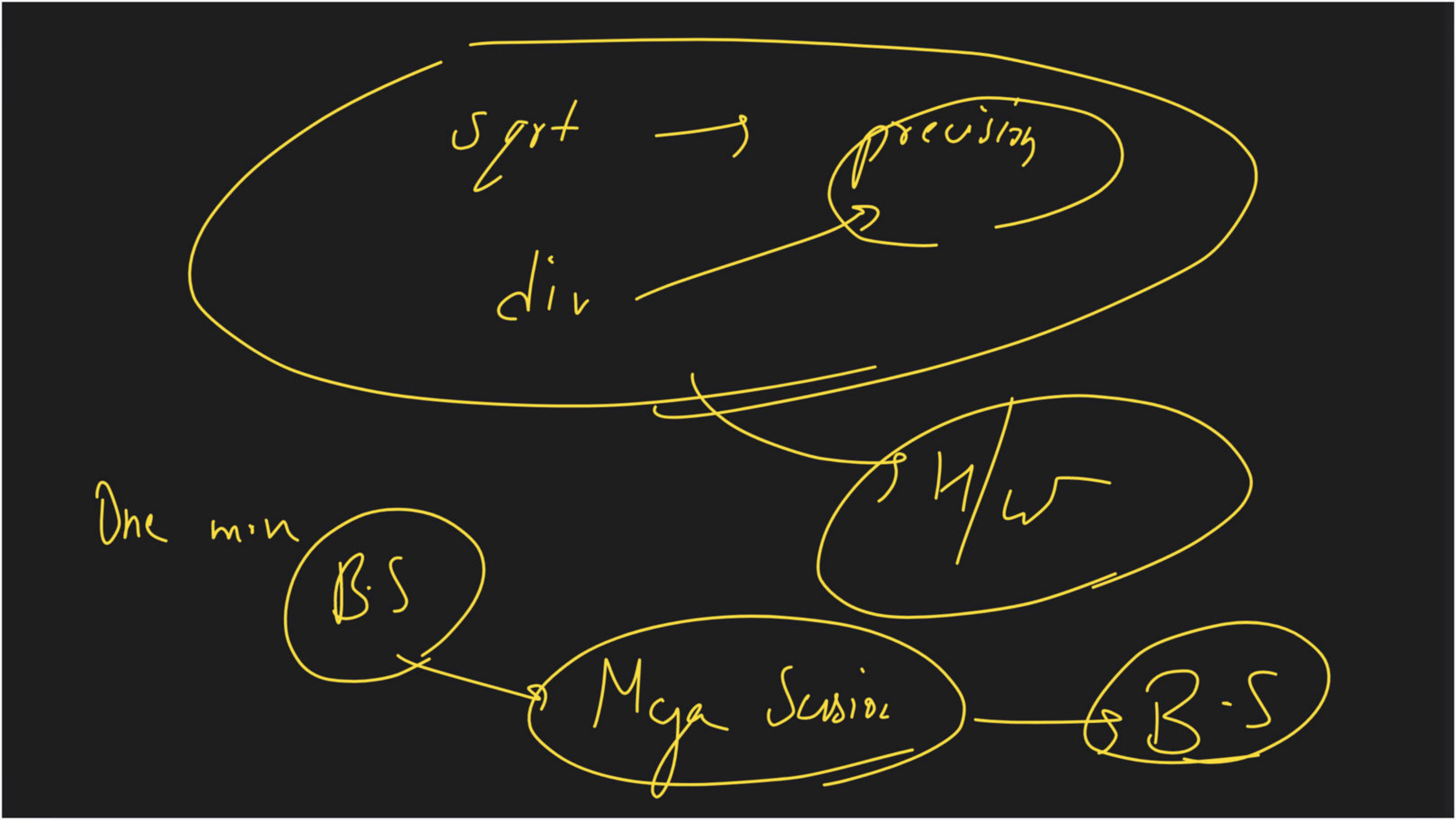
(C) no pair repeats itself, no number can occur more than 2 times in a single stretch

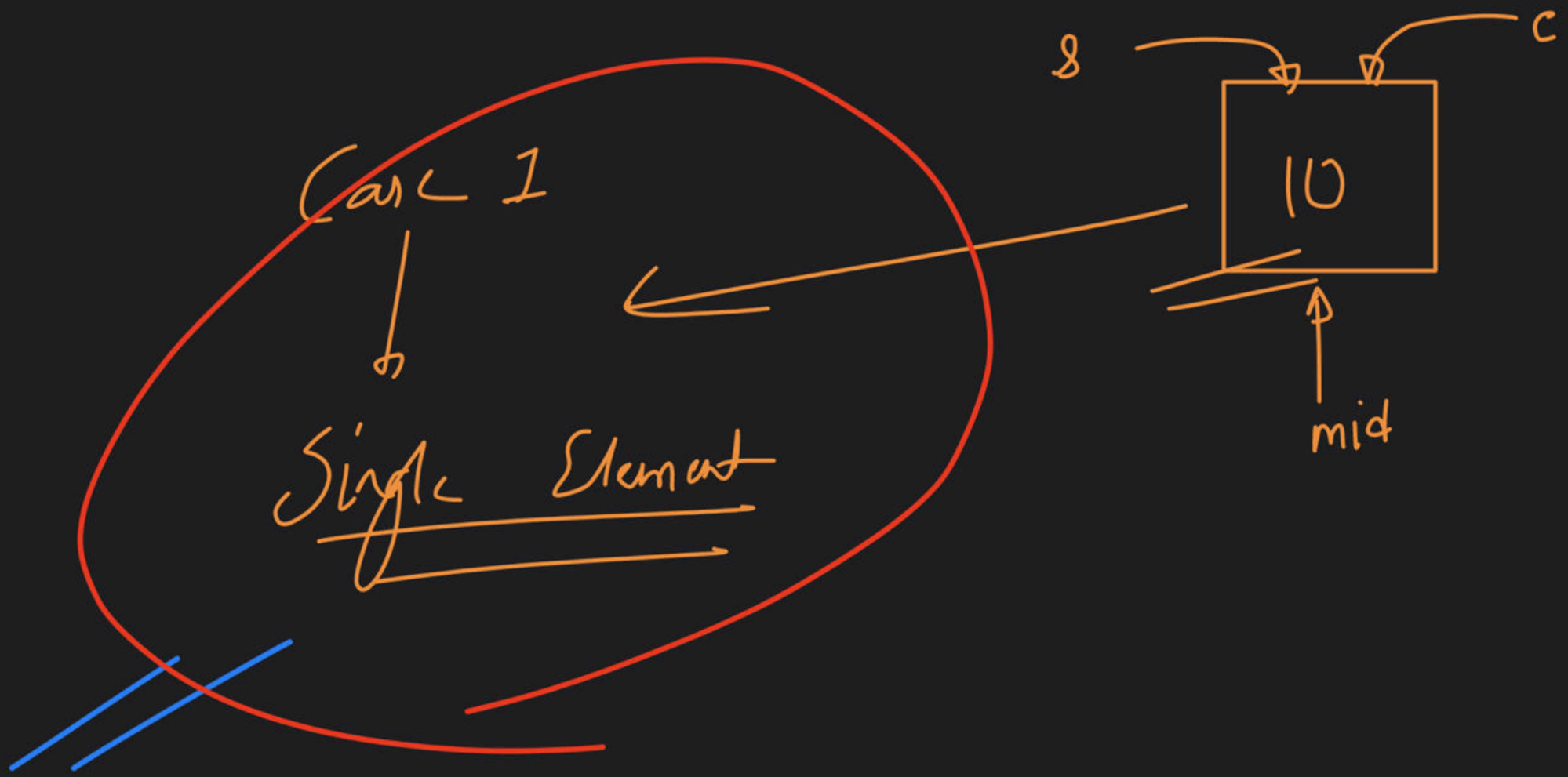
(D) find element that occur Odd times

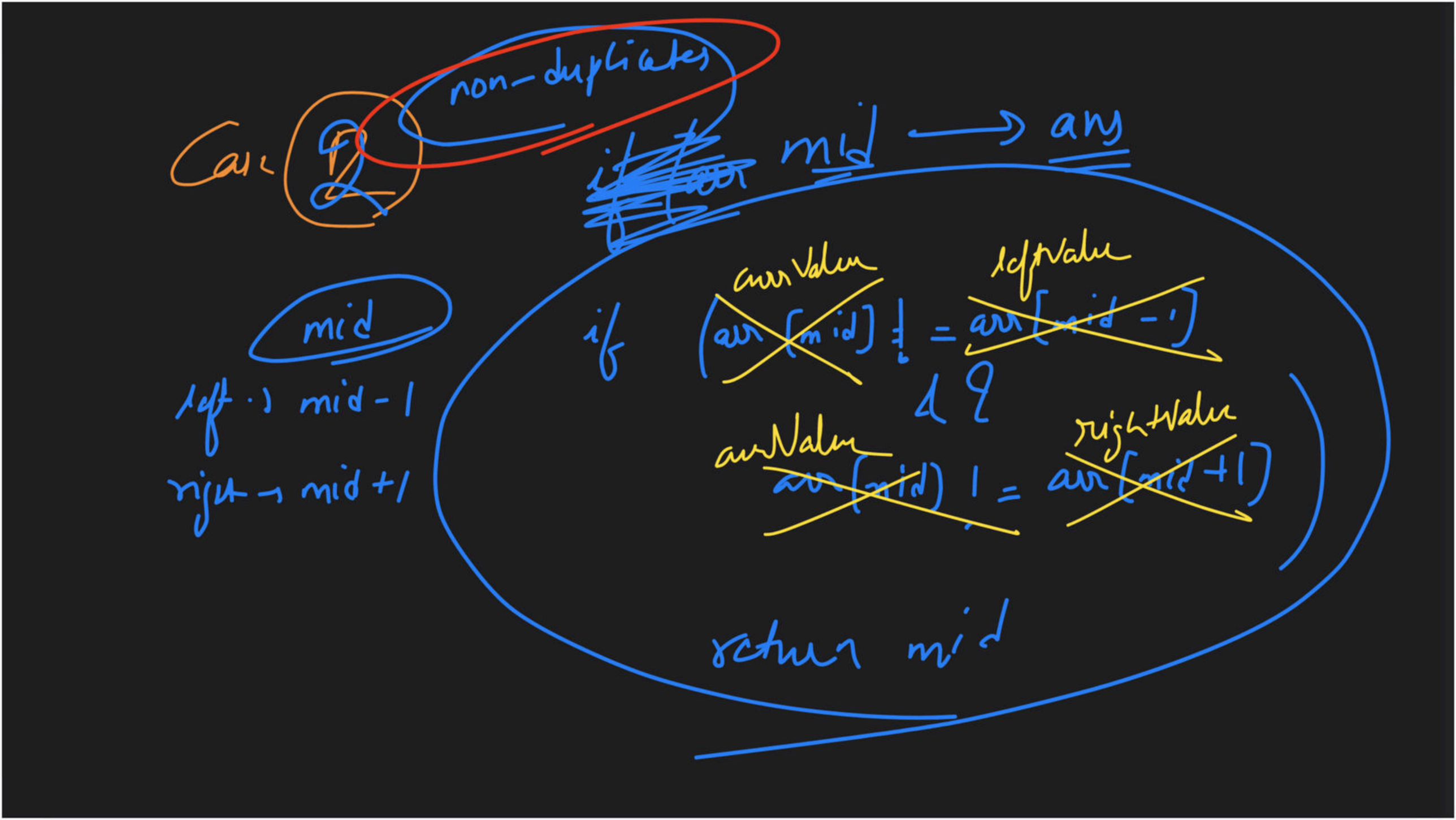

we need to handle this
4 2 2 3 3 5 5 2 2 6 6
0 1 2 3 4 5 6 7 8 9 10
-1
we need to handle this
2 2 3 3 5 5 2 2 6 6 4
0 1 2 3 4 5 6 7 8 9 10
-1
mid
left part → pair -> st -> Even
mid
mid
right part → pair -> st -> Odd

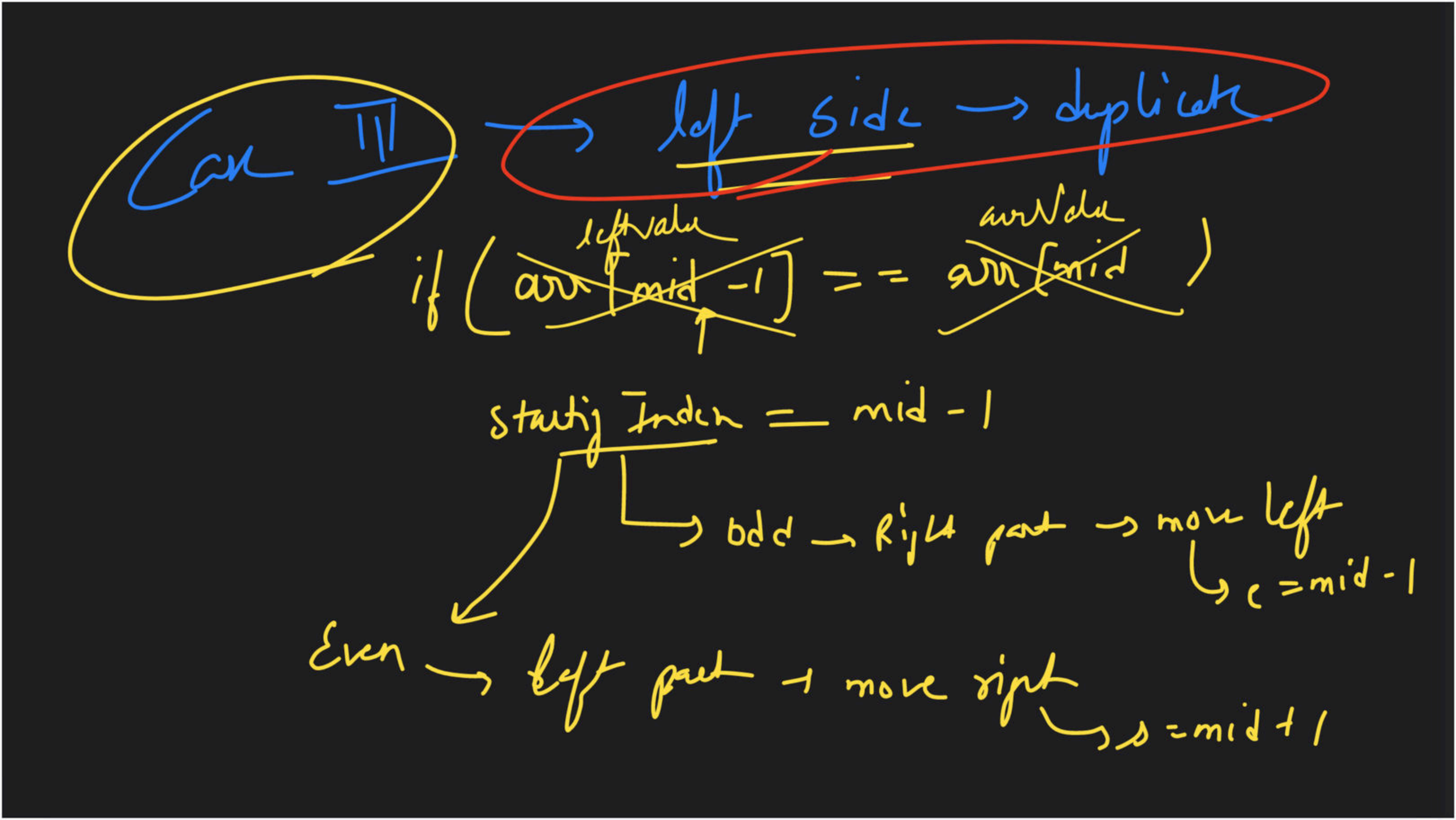

int currValue = arr[mid]
int leftValue = -1
int rightValue = -1
if (mid-1 >=0)
leftValue = arr(mid-1]
if (mid+1 < n)
rightValue = arr[mid+1]


Instructor
A
B
C
D
E
HW
Sorting Algorithm
maze


sqrt
precision
div
One min
B.S
Mega Session
B.S


s
c
10
mid
Case 1
Single Element


non-duplicates
mid → any
mid
left → mid - 1
right → mid + 1
if
arrValue
leftValue
rightValue
return mid


Case III
left side → duplicate
if (arr[mid-1] == arr[mid])
leftValue
arrValue
staticj Index = mid - 1
odd → Right part → move left
c = mid - 1
Even → left part → move right
s = mid + 1

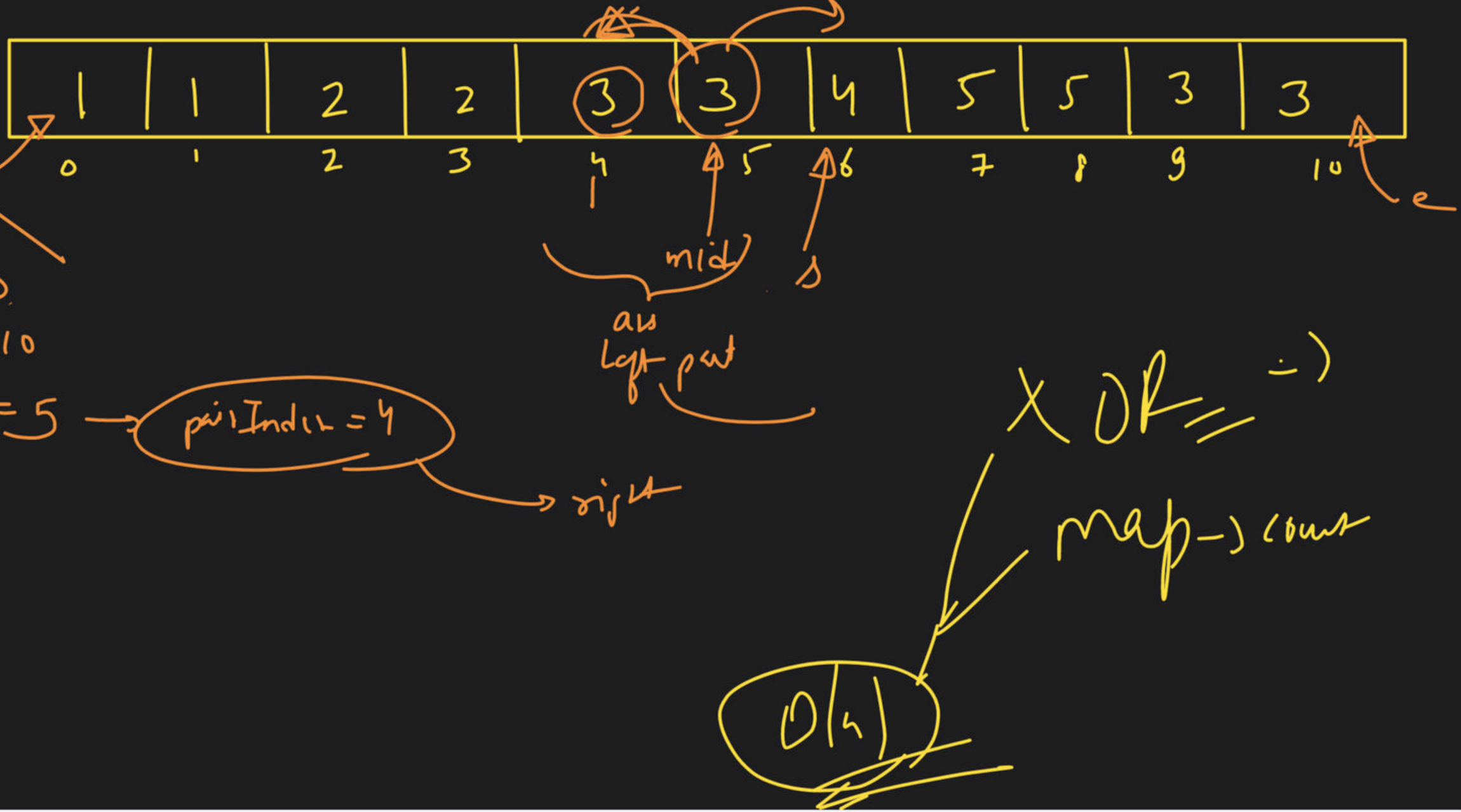
Case IV -> right -> duplicate

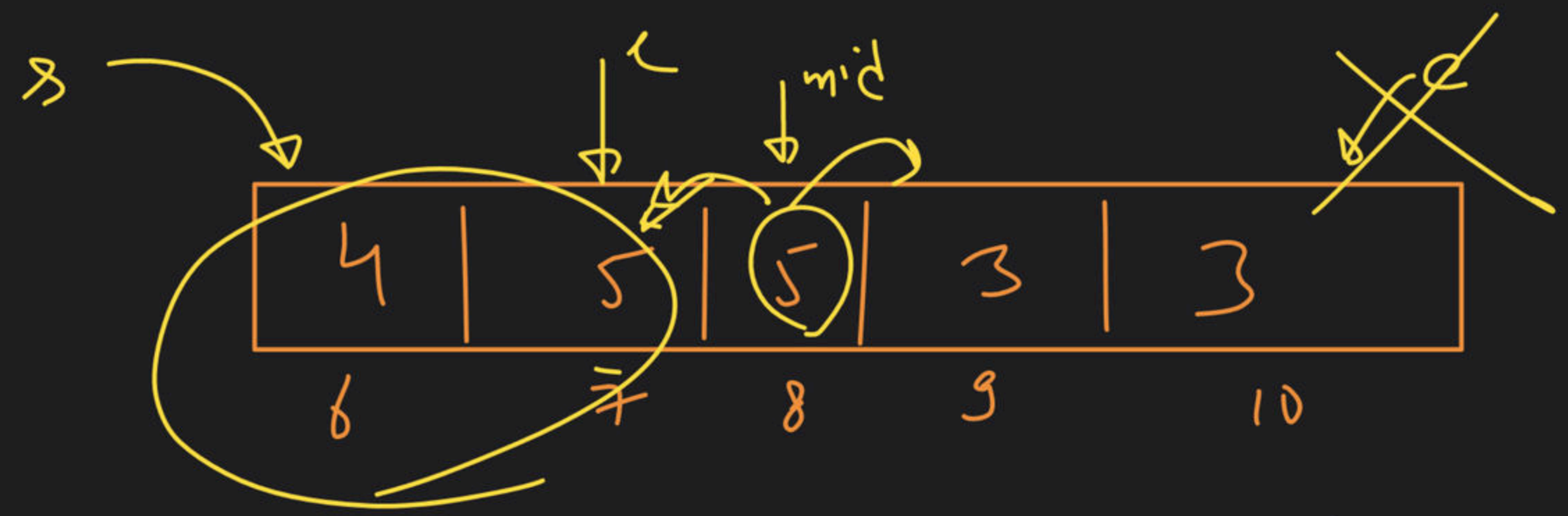
if (arr[mid] == arr[mid+1])

↳ startIndex -> mid

↳ Odd -> R.P -> move left

↳ Even -> Left part -> move right

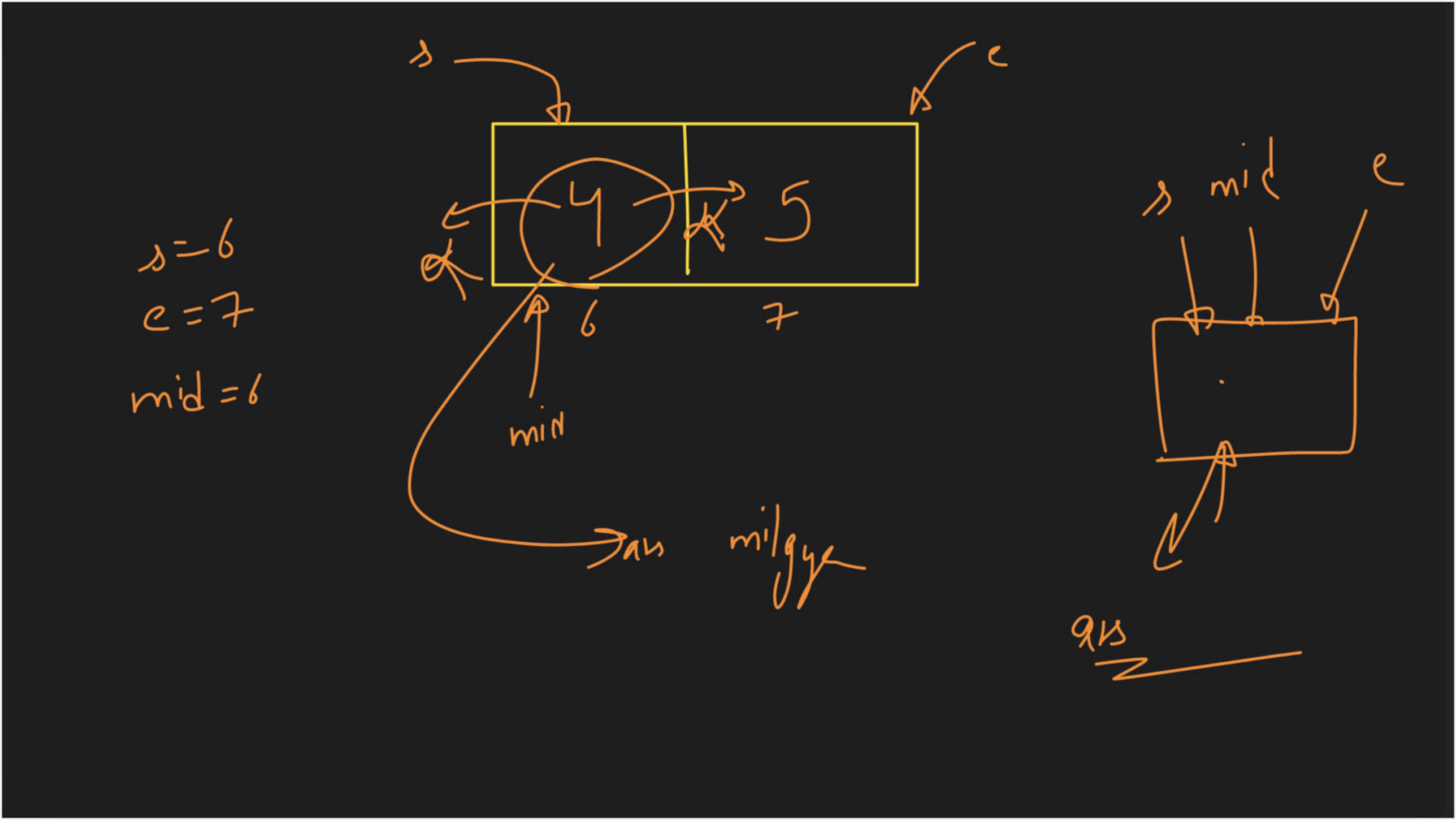

1 1 2 2 3 3 4 5 5 3 3
0 1 2 3 4 5 6 7 8 9 10
s
mid
e
s=0
e=10
mid=5
pairIndex=4
right
XOR
map
0/1

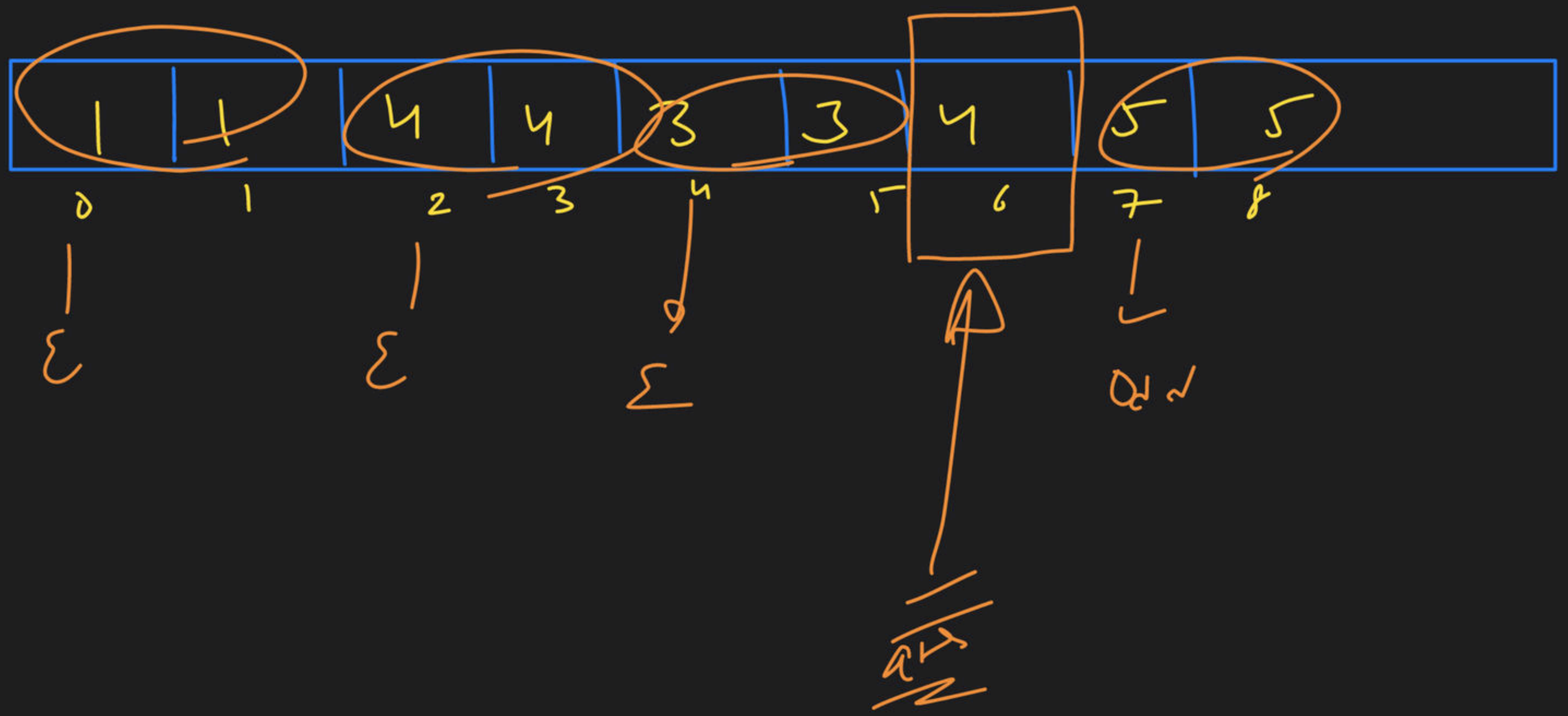


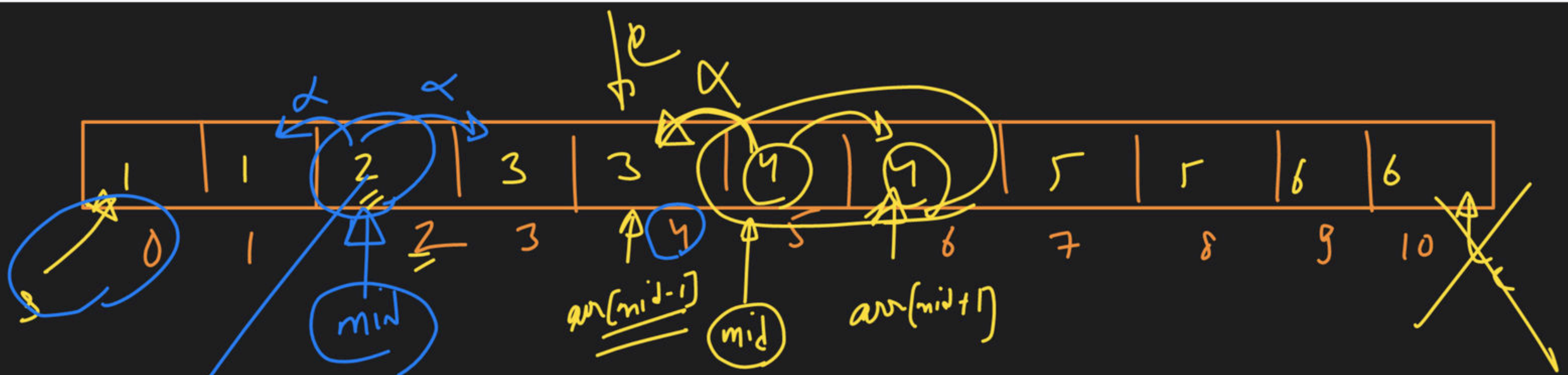
s = 6

e = 10

mid = 8 → pairIndex = 7 → Odd → Right Part → left

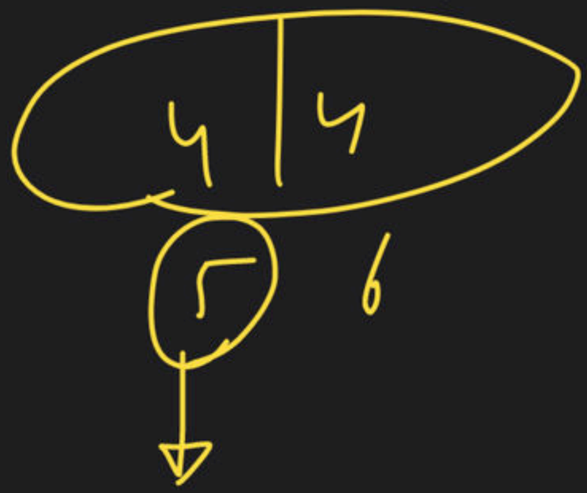

s
e
4
5
6
7
mid
s=6
e=7
mid=6
ans mil gya
s
mid
e
ans


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



mid → 5 index

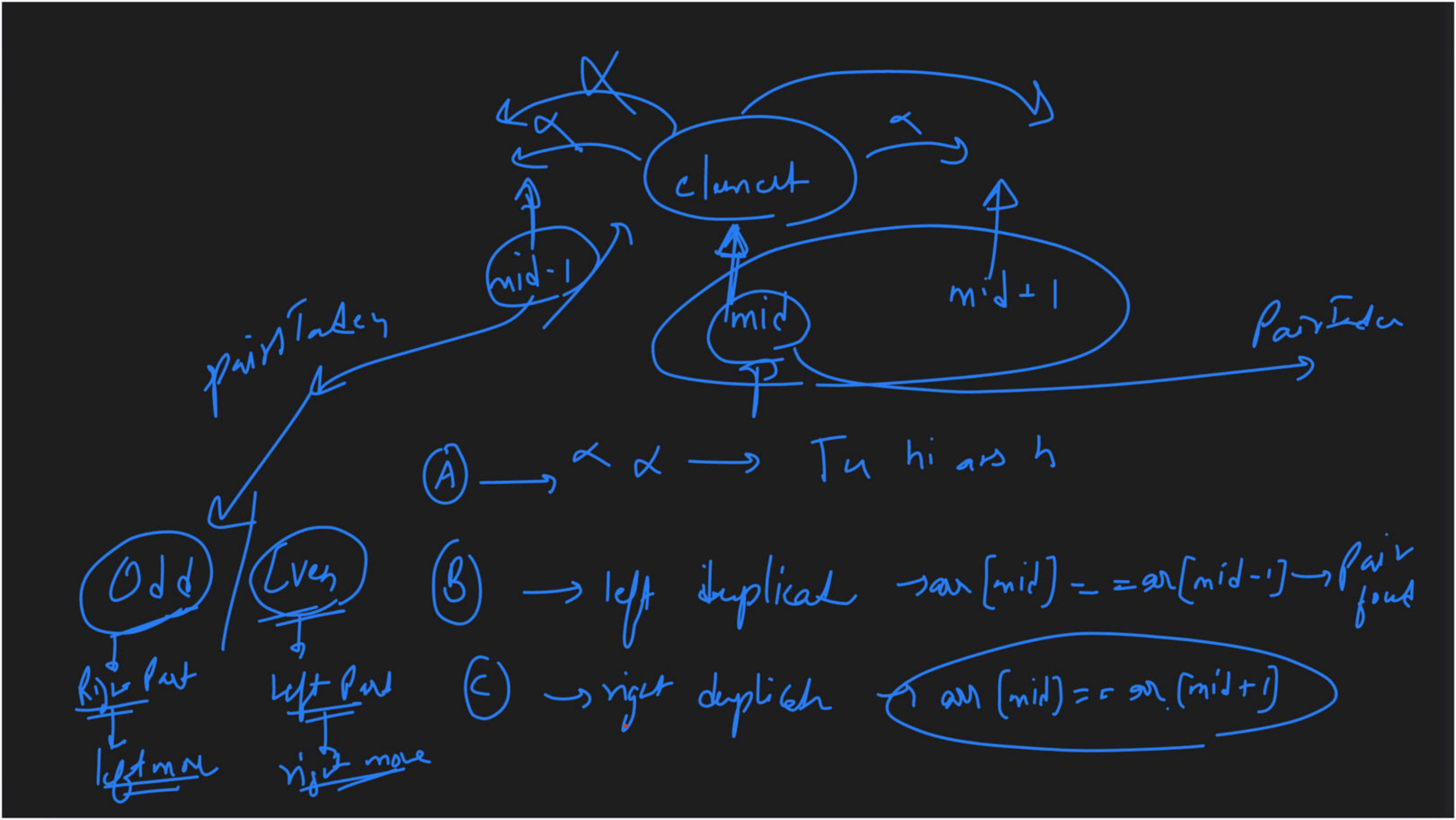
→ element = 4

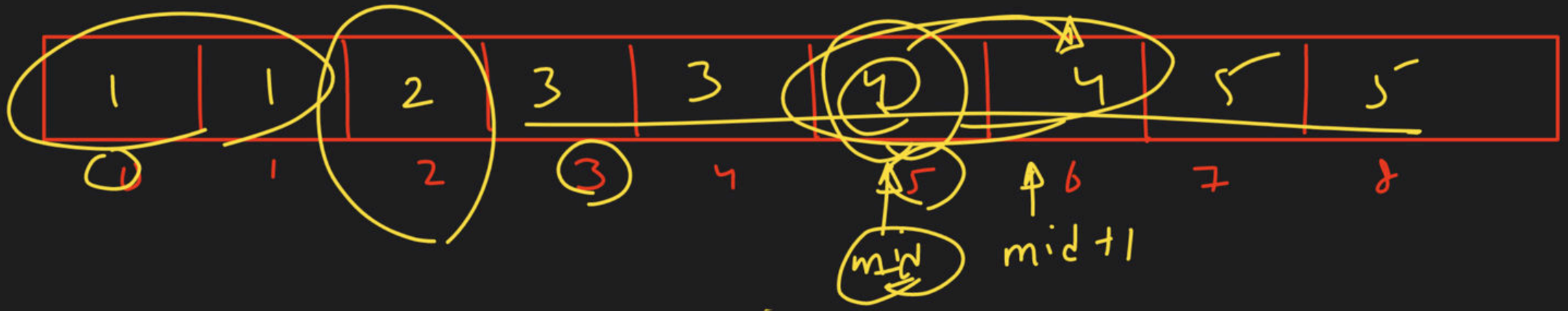


Odd → Right Part

left

All


clumcut
mid-1
mid
mid+1
pairIndex
Odd
Even
Right Part
Left Part
leftmost
right most
A → α α → Turn hi ans h
B → left duplicat → arr[mid] == arr[mid-1] → pair found
C → right duplicate → arr[mid] == arr[mid+1]



Case IV ——> right duplicate

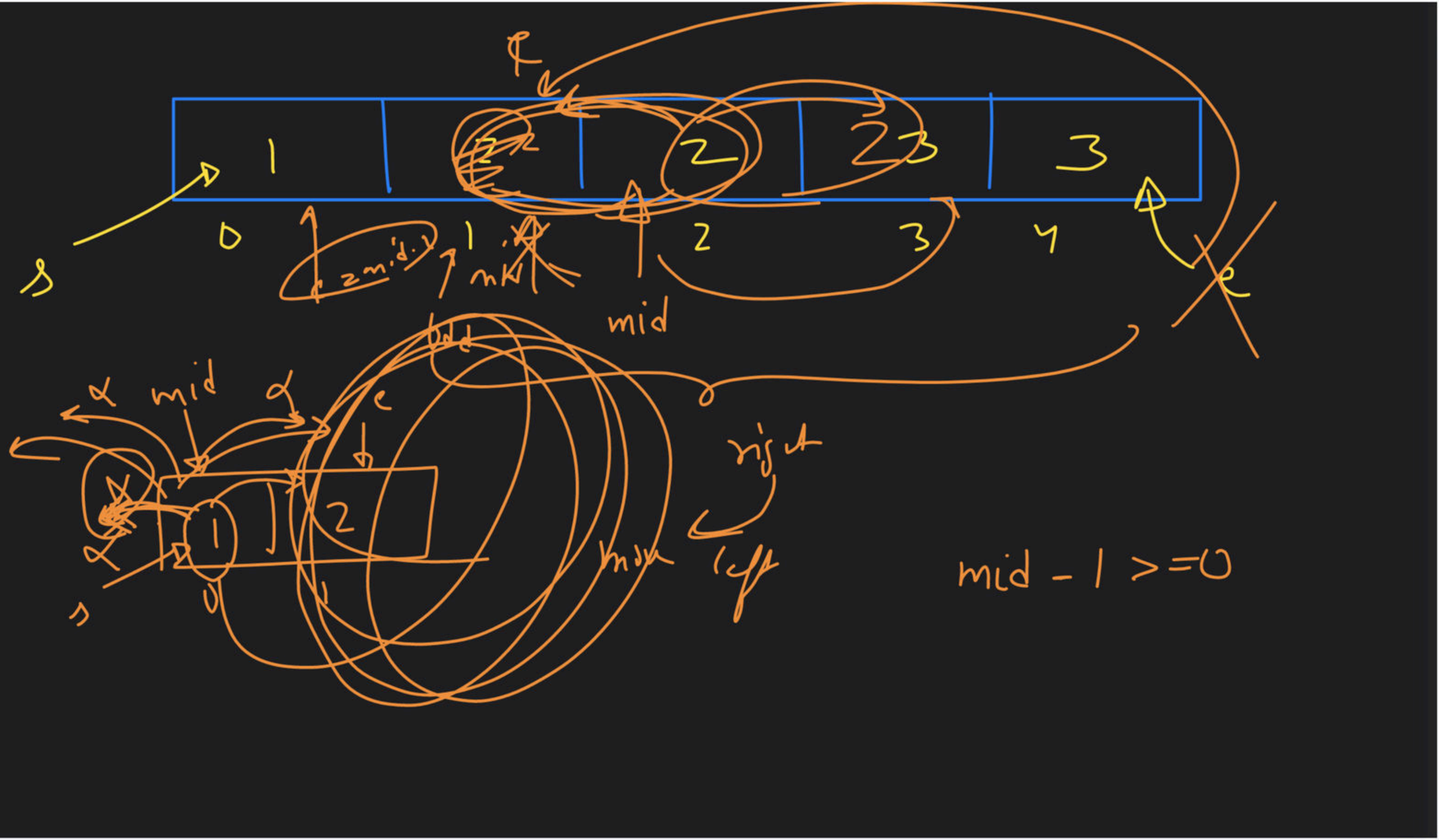
if (arr[mid] == arr[mid+1])

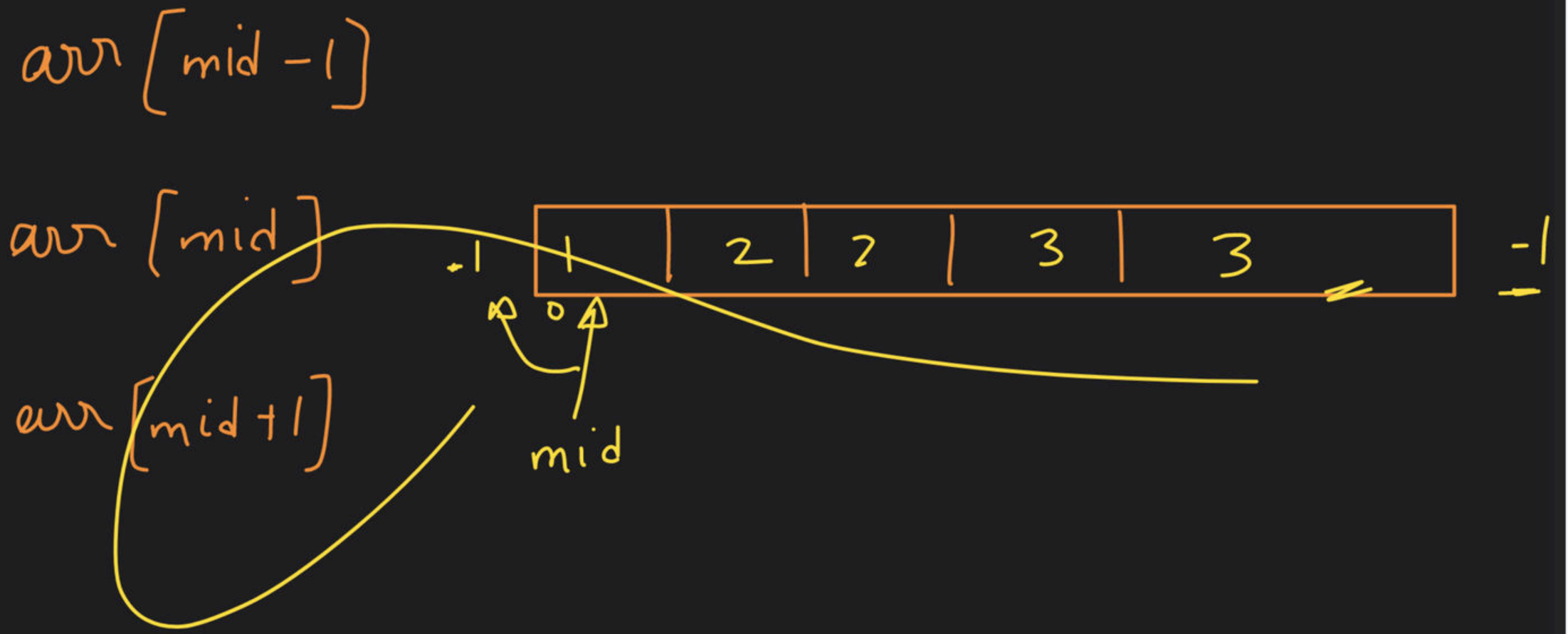
↳ pairStartIndex = mid

↳ Odd → R.P → move left → e = mid-1

↳ Even → L.P → move right → s = mid+1

90.1.


1
2 2
2
2 3
3
0
1
2
3
4
mid
mid
mid
mid - 1 >= 0


arr[mid-1]
arr[mid]
arr[mid+1]
-1
1
2
2
3
3
-1
0
mid