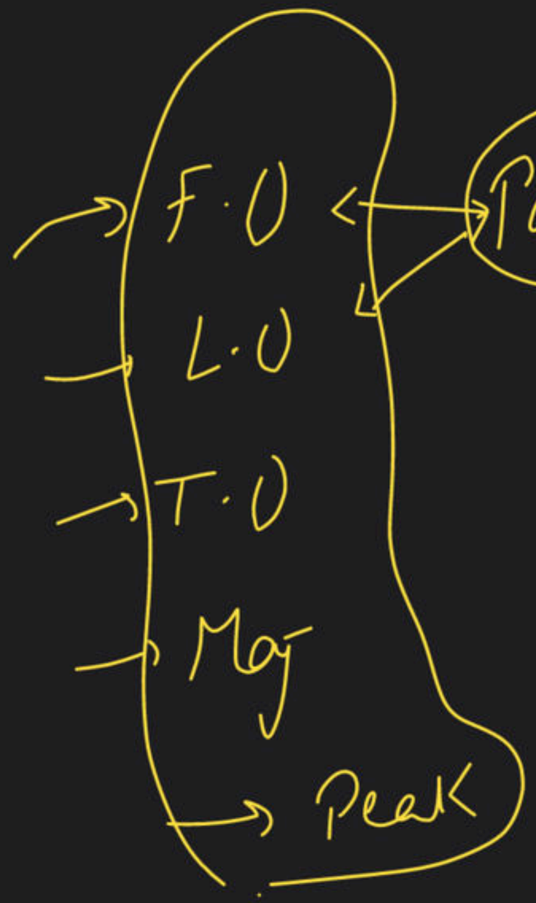
Searching && Sorting - Level 2

Special class

Love Babbar • Sept 18, 2023

→ Binary Search ⟹ (cond^{n}) monotonic func^{n} → asc / desc

T.C → O(log n)
↳ n → size of array

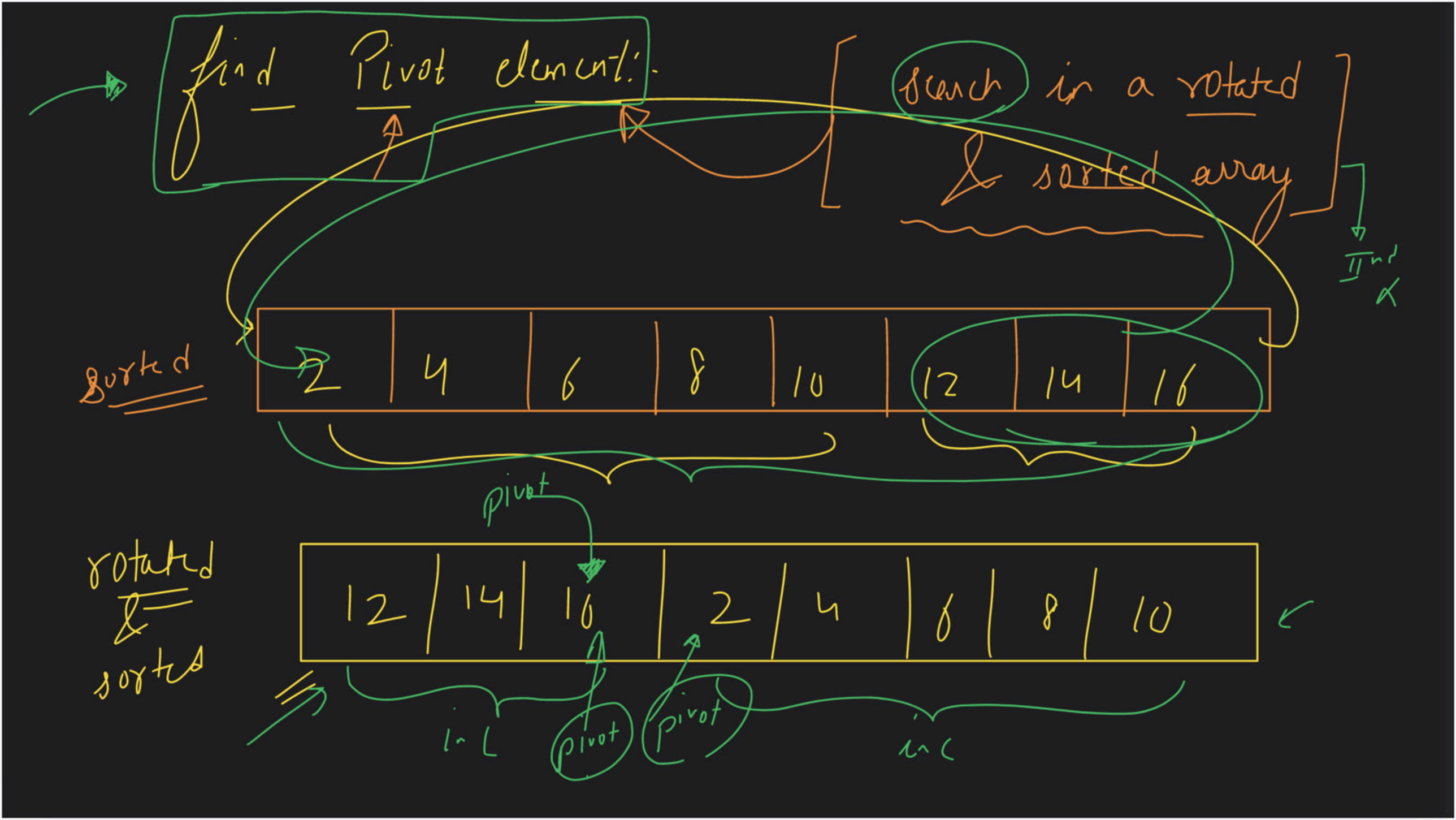


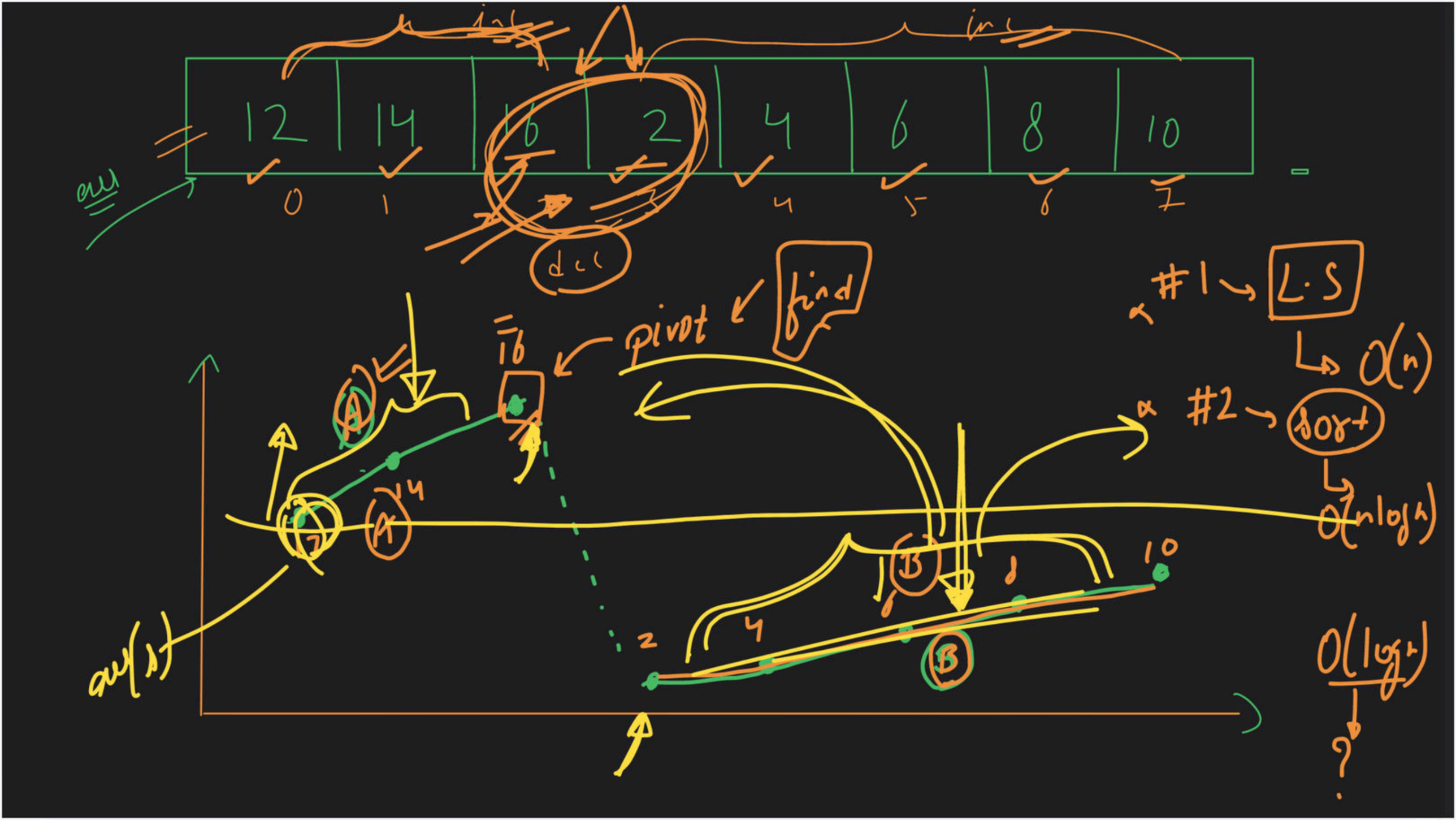
Pattern → any store
→ 1/c → 1/s → Know

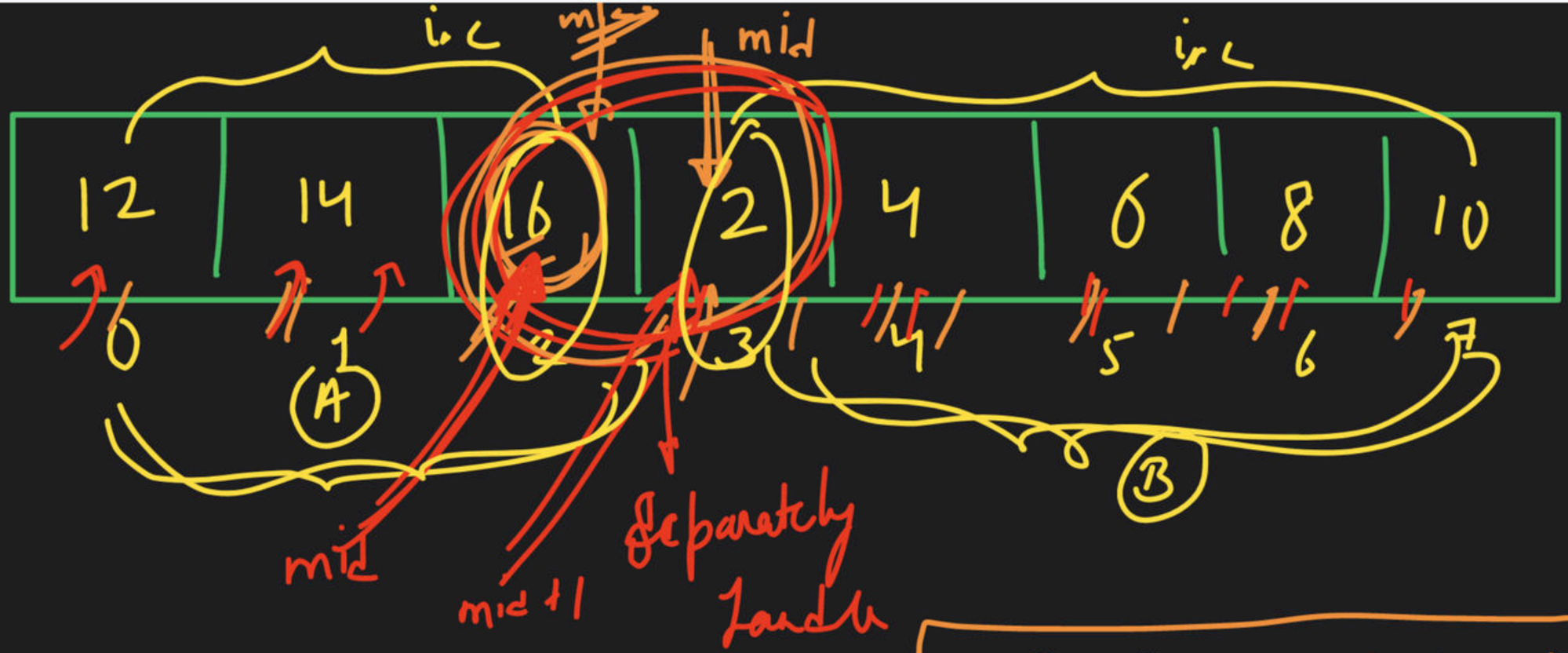
1024 size

10 operation

1024 opcr


find Pivot element:-
search in a rotated & sorted array
II nd
sorted
2
4
6
8
10
12
14
16
pivot
rotated & sorted
12
14
16
2
4
6
8
10
in L
pivot
pivot
in C


12
14
16
2
4
6
8
10
0
1
4
5
6
7
dcl
pivot
find
#1
L.S
O(n)
#2
sort
O(nlogn)
O(logn)
?

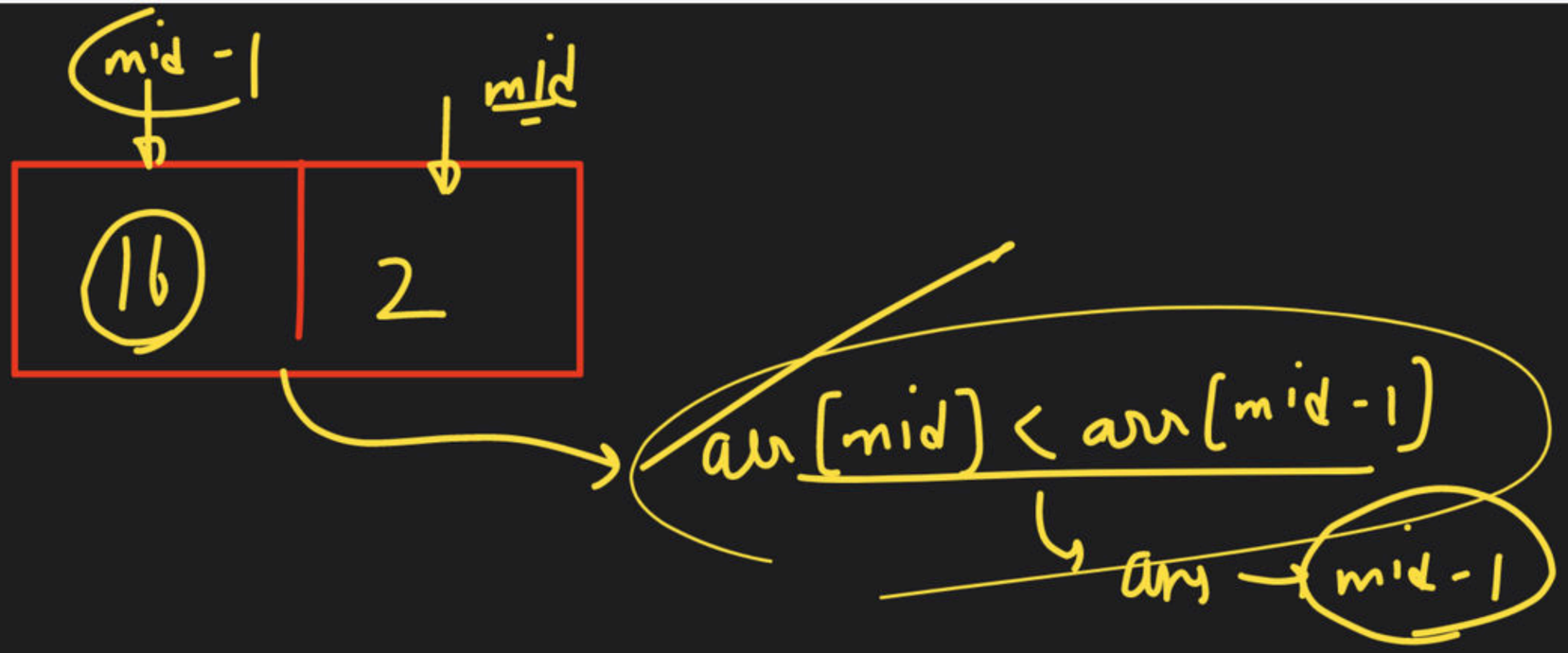


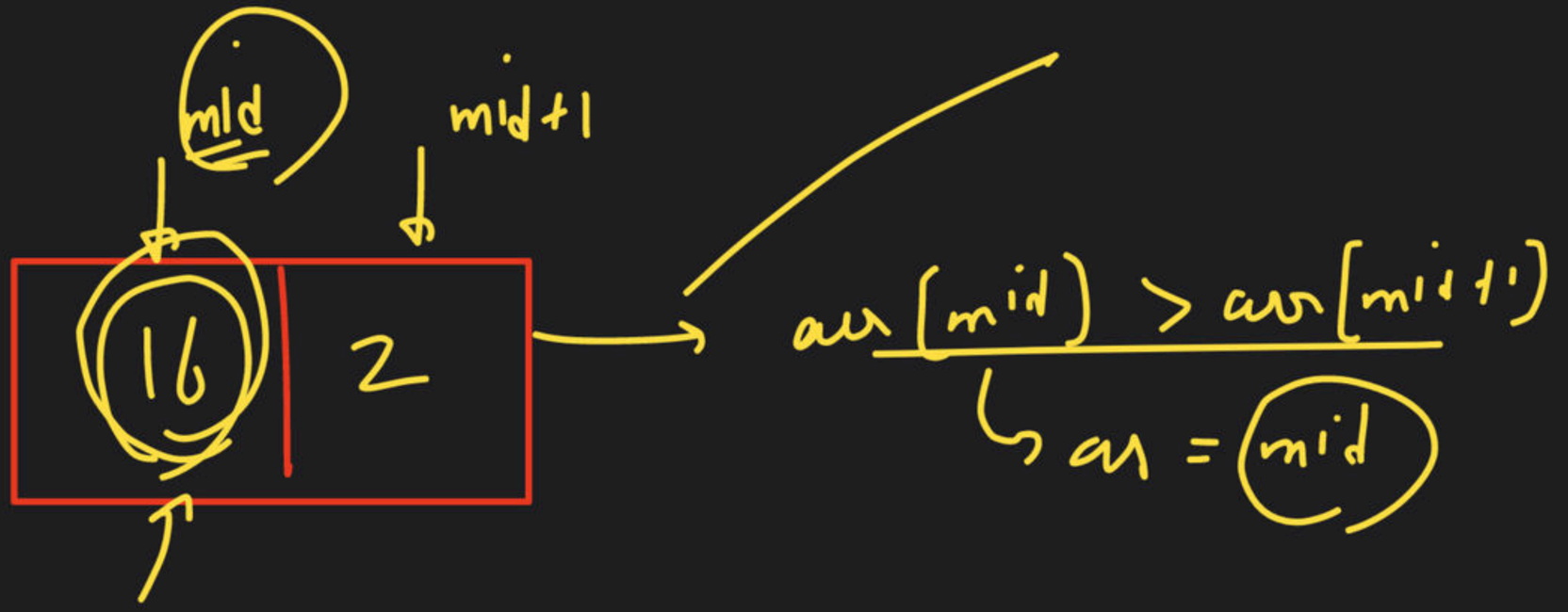
arr[mid] < arr[mid-1]

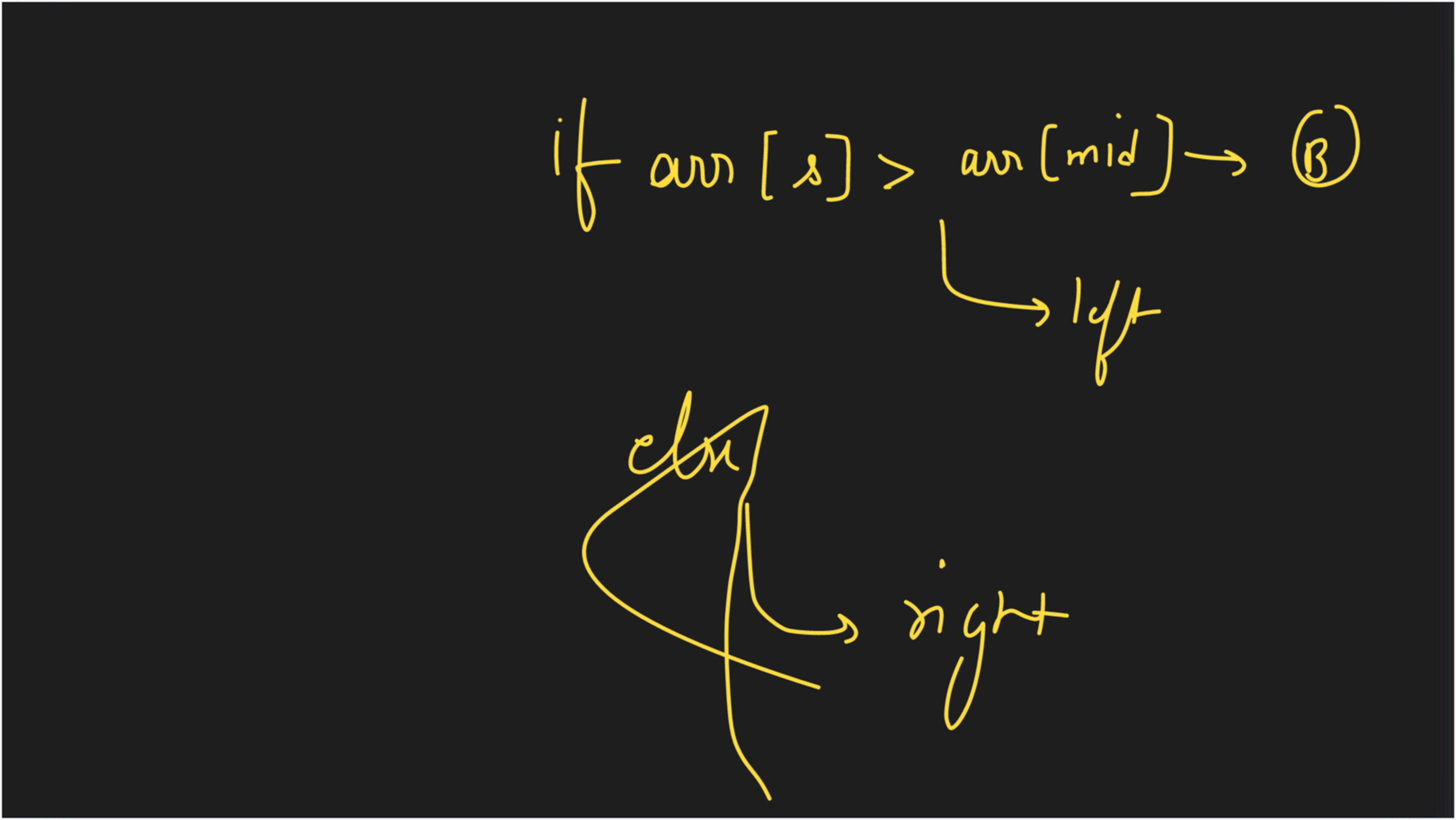
(I) → any = (mid-1)

arr[mid] > arr[mid+1]

(II) → any = mid


mid - 1
mid
16
2
arr[mid] < arr[mid-1]
ans
mid - 1


mid
mid+1
16
2
arr[mid] > arr[mid+1]
ans = mid


if arr[s] > arr[mid] → (B)
left
else
right

```
int findPivotIndex(arr[], n)
{
    int s=0, e=n-1;  int mid = (s+e)/2;
    while(s<=e)
    {
        if(s==e) return s;
        if (arr[mid] < arr[mid-1]) return mid-1;
        else if (arr[mid] > arr[mid+1]) return mid;
        else if (arr[s] > arr[mid]) e=mid-1;
        else
            s = mid+1;
        mid = (s+e)/2;
    }
    return -1;
}
```

corner case

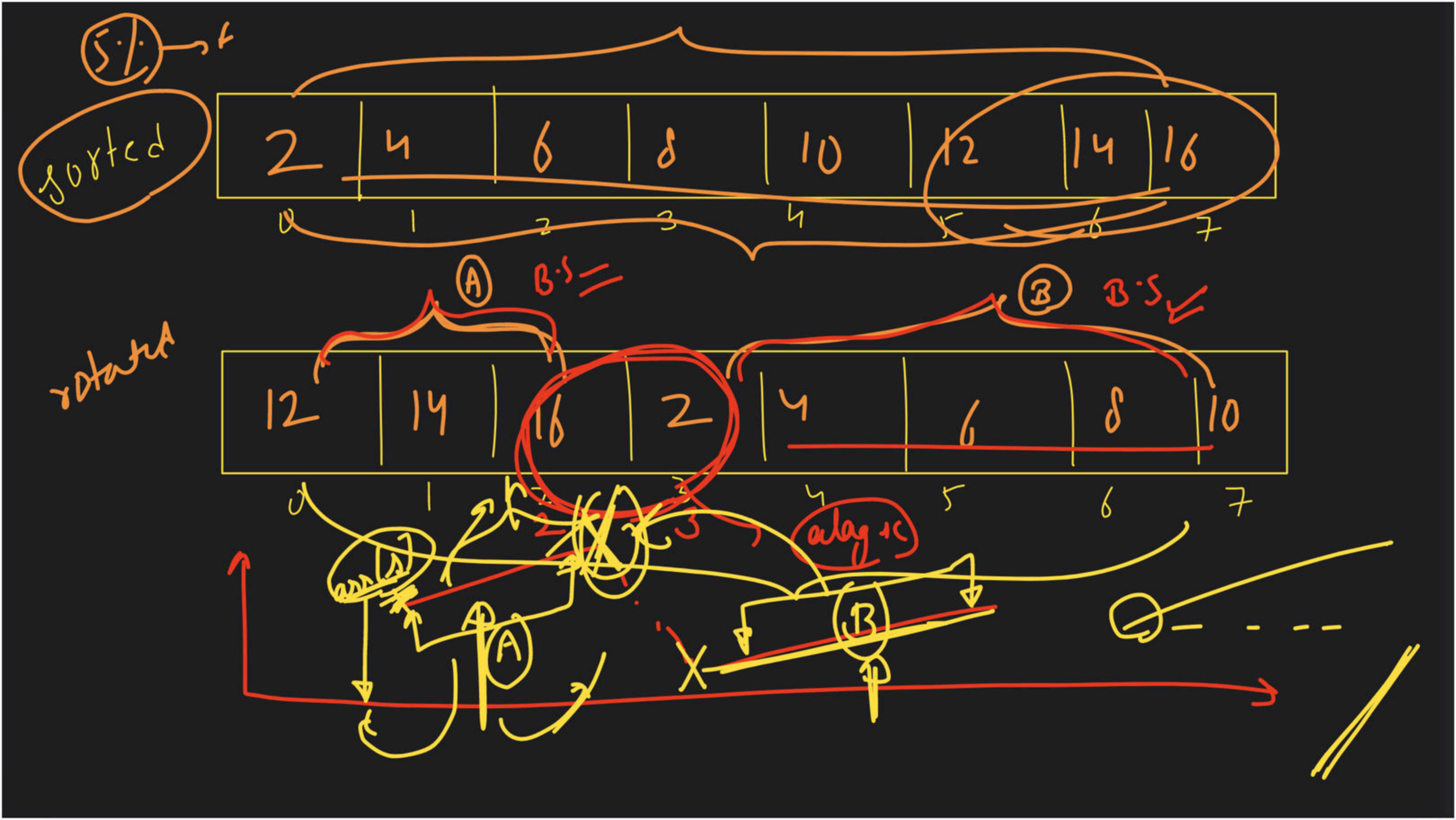
single element

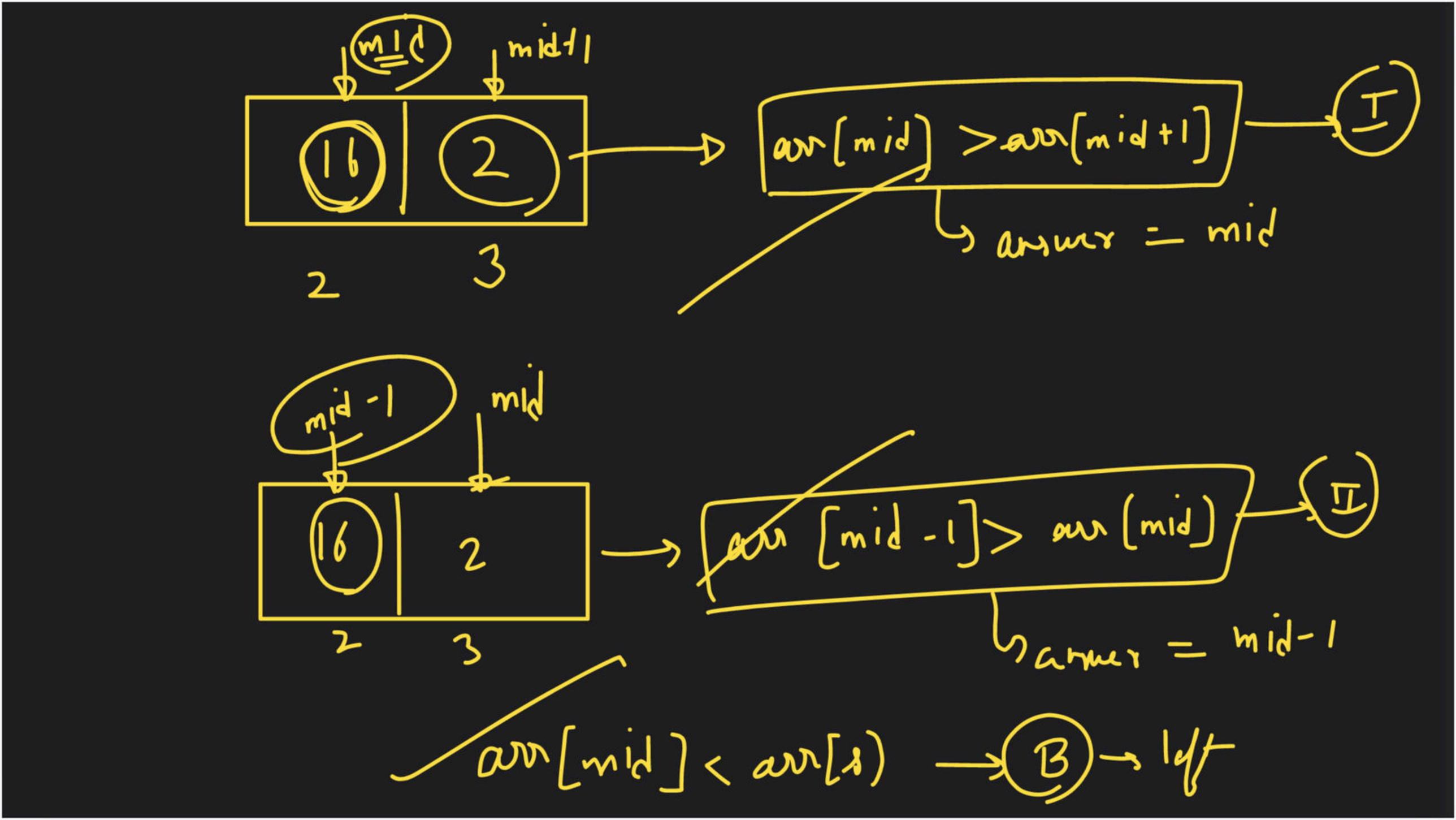
s==e

i/p → sorted / rotated array

o/p → Target → Abs

→ Pre

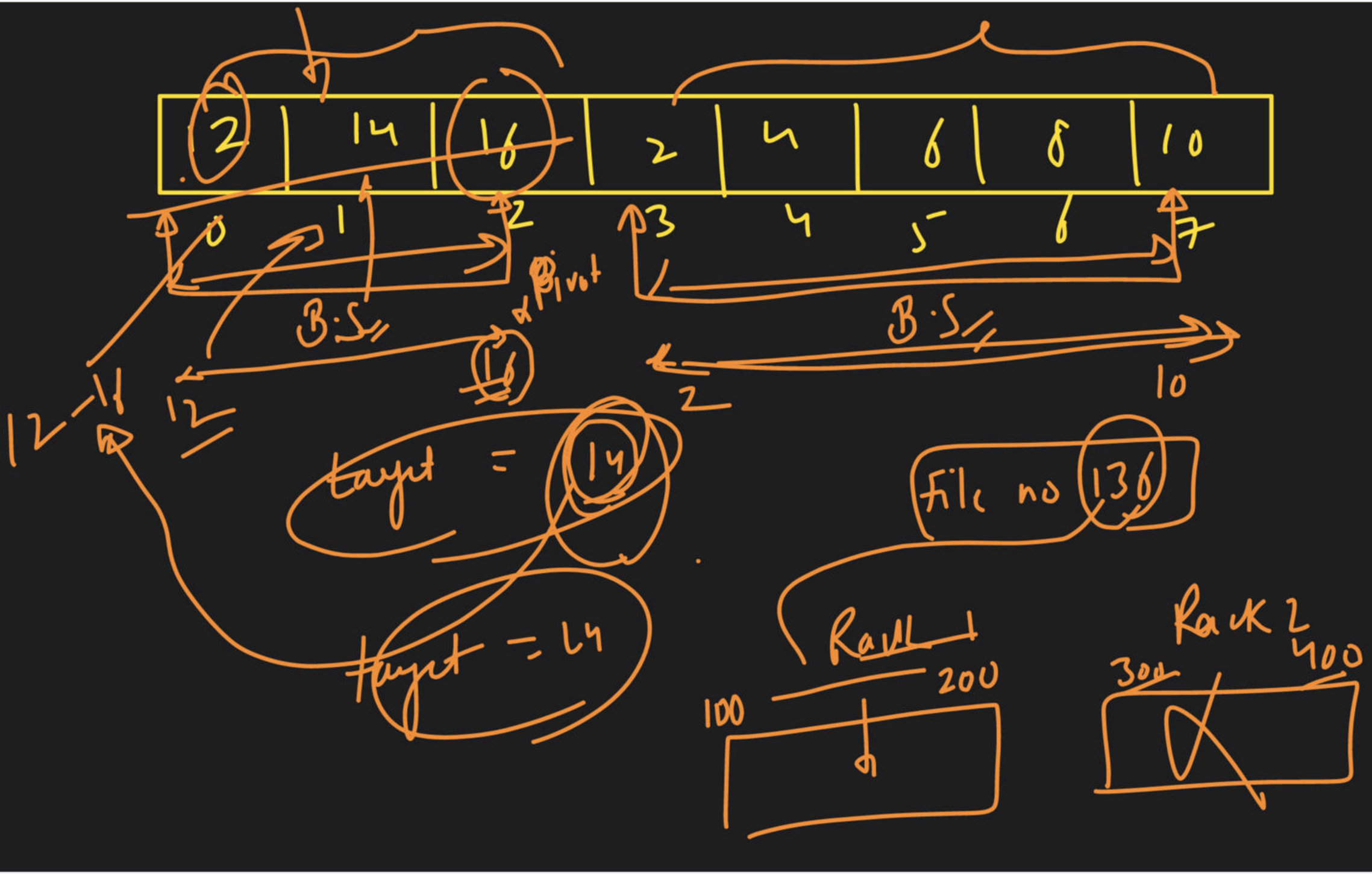

sorted
2
4
6
8
10
12
14
16
0
1
2
3
4
5
6
7
A
B
B·S
B·S
rotated
12
14
16
2
4
6
8
10
0
1
2
3
4
5
6
7
A
B
X

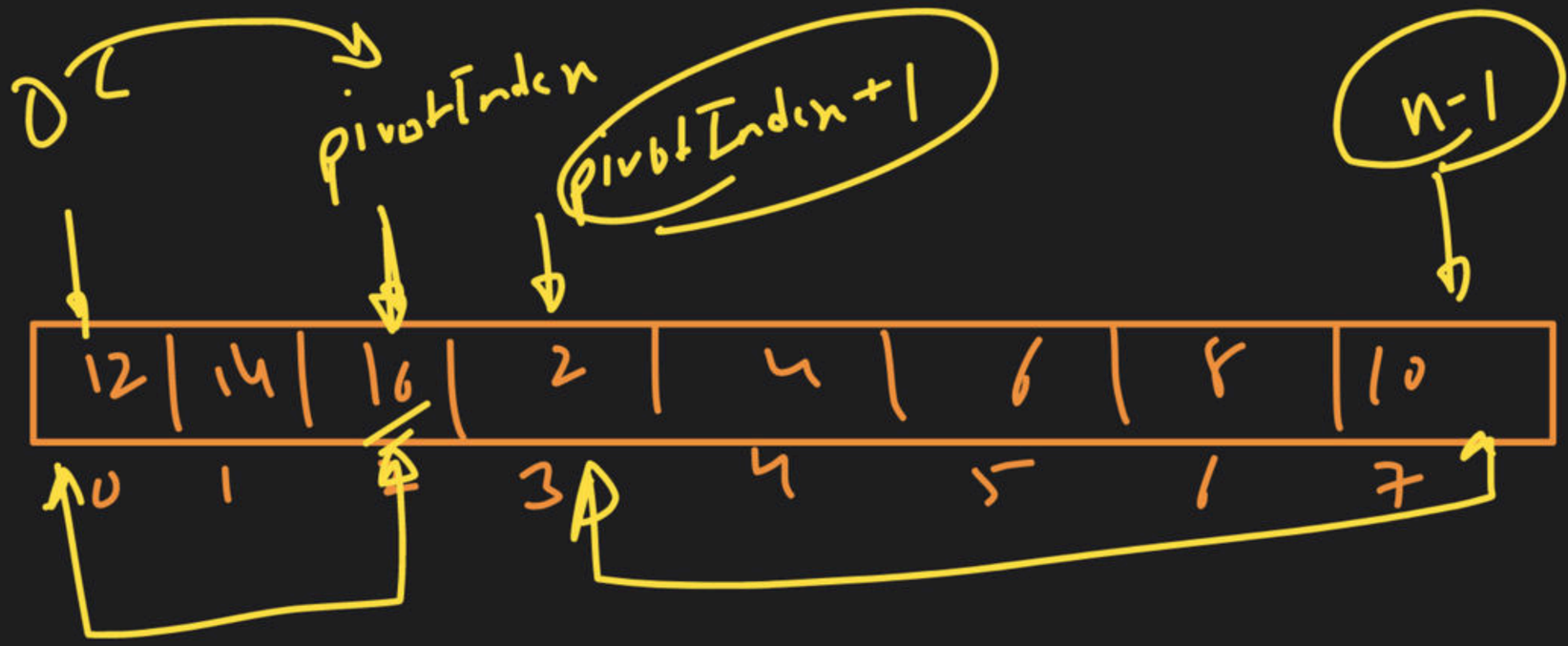

mid
mid+1
16
2
2
3
arr[mid] > arr[mid+1]
T
answer = mid
mid-1
mid
16
2
2
3
arr [mid-1] > arr [mid]
II
answer = mid-1
arr[mid] < arr[1]
B
left

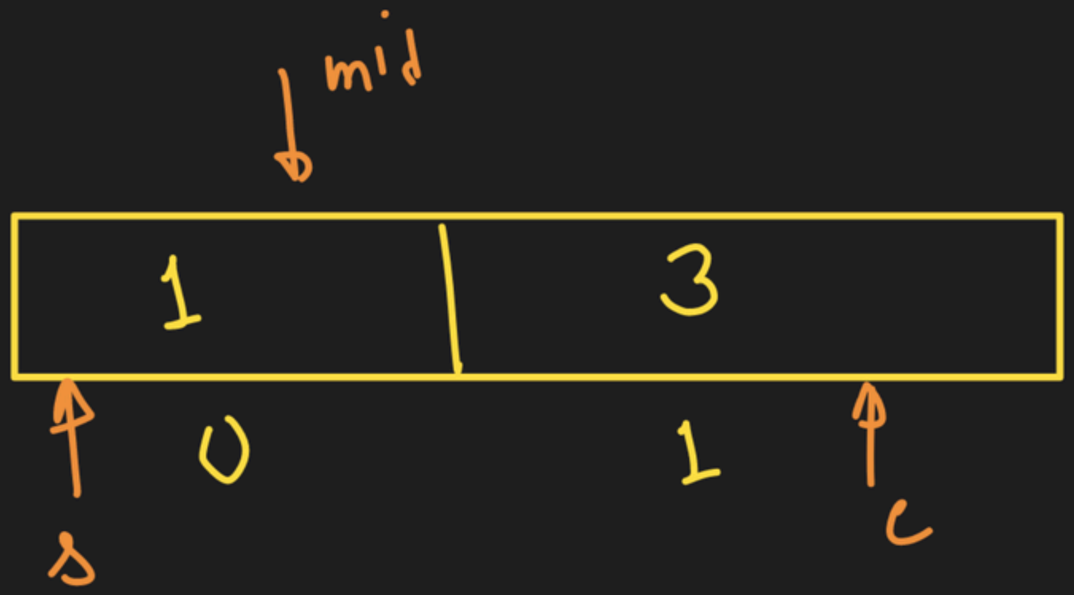
arr[mid] > arr[s] → (A) → right

if(s==e) return s

while → single elem →


2
14
16
2
4
6
8
10
0
1
2
3
4
5
6
7
B.S
Pivot
B.S
12
12
2
10
target = 14
target = 14
File no 136
Rack 1
100
200
Rack 2
300
400


0
pivotIndex
pivotIndex+1
n-1
12
14
16
2
4
6
8
10
0
1
2
3
4
5
6
7



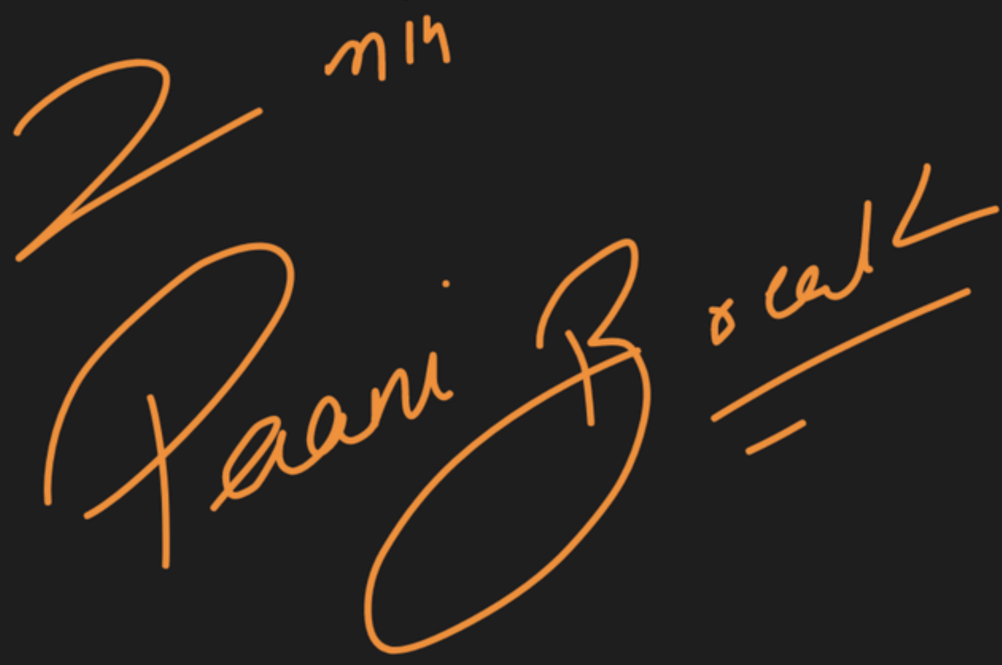
pivot

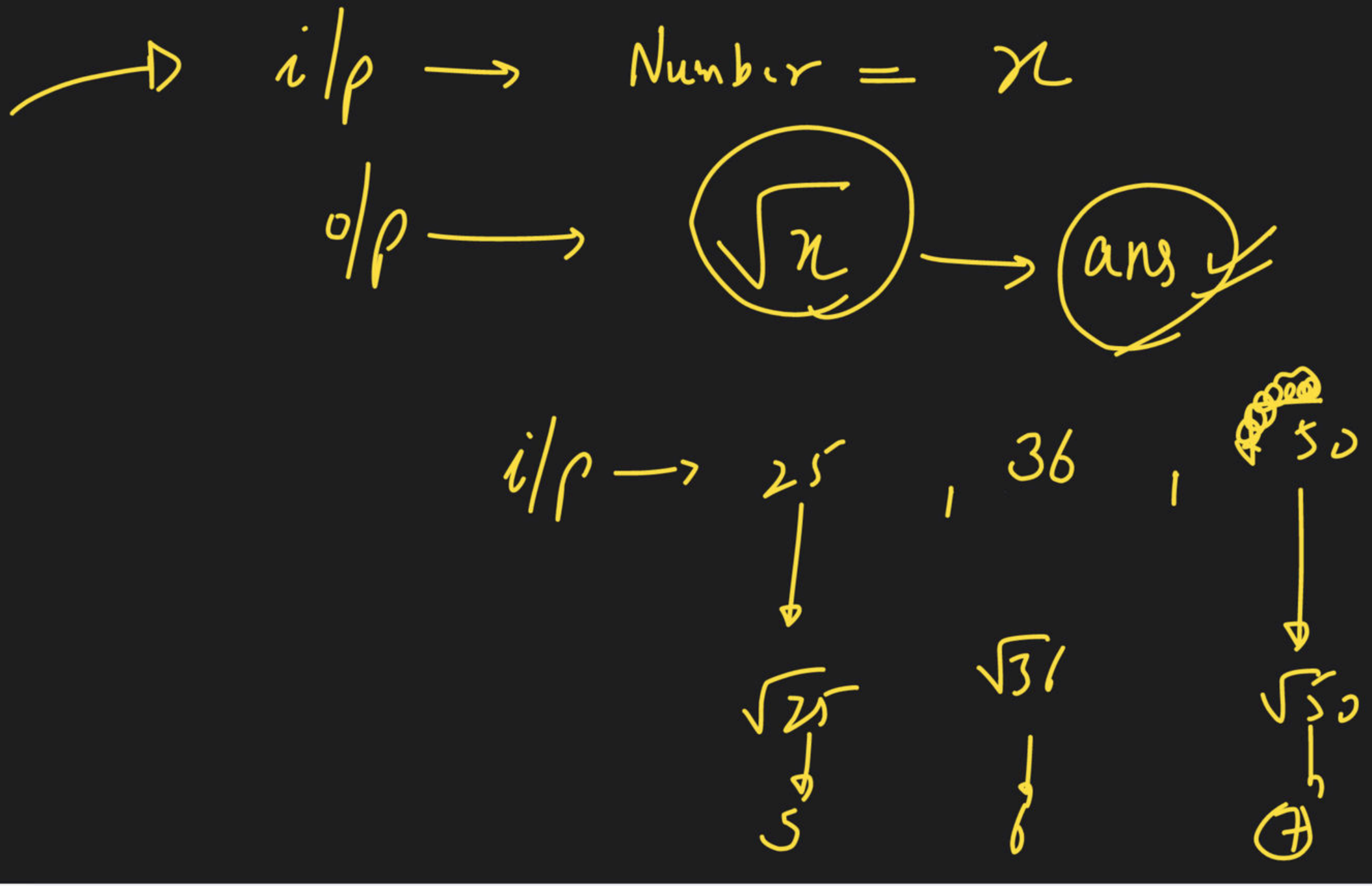
arr[mid] > arr[mid+1] X

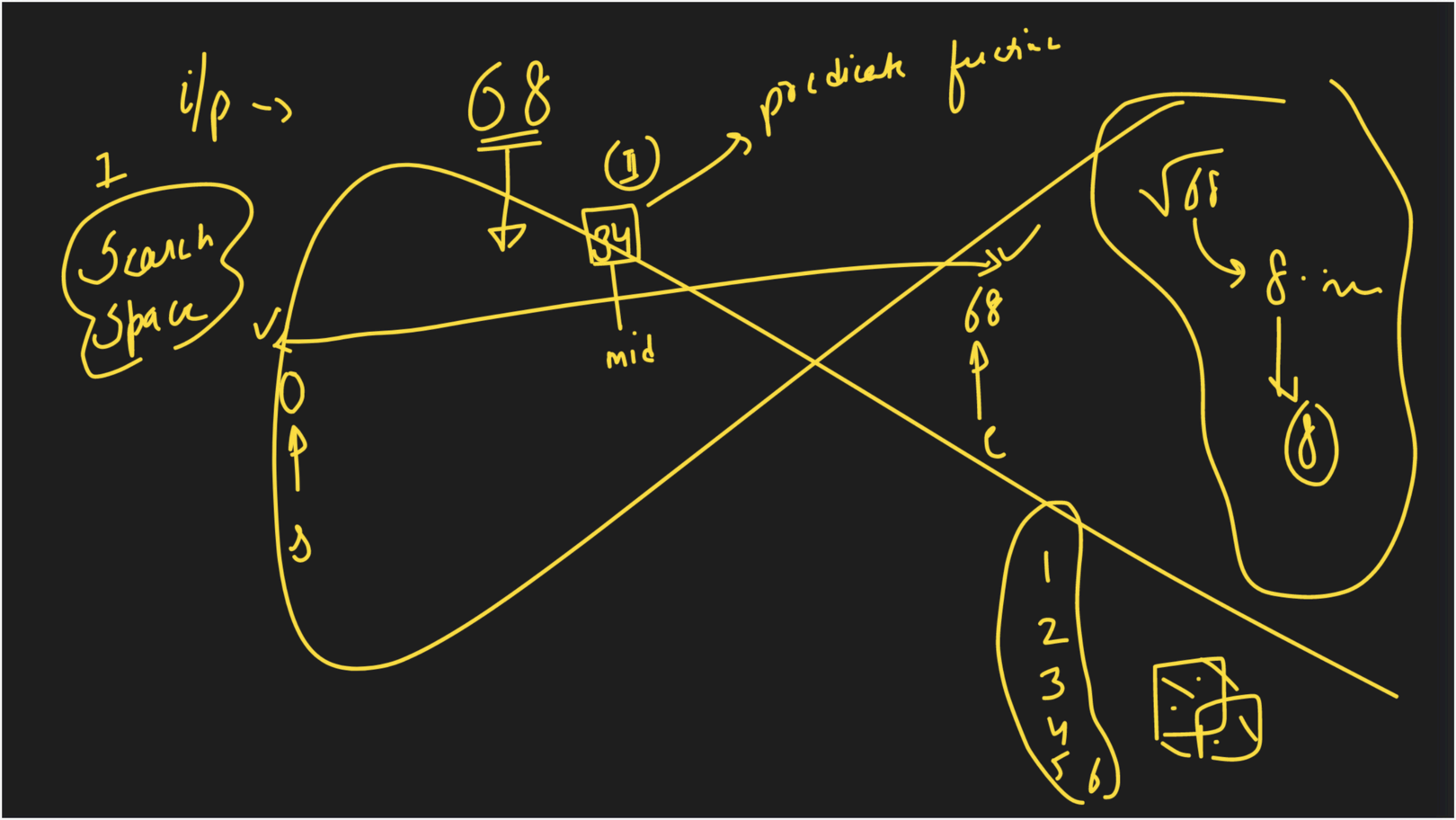
arr[mid-1] > arr[mid]

0 1

[-1]


2 min
Paani Break


i/p → Number = n
o/p →
√n
ans
i/p → 25 , 36 , 50
√25
5
√36
6
√50
7


i/p ->
68
predicate function
1
Search Space
(I)
34
mid
0
s
68
e
√68
8
1
2
3
4
5
6

(1) i/p → n

o/p → sqrt(n) → $\sqrt{n}$ → ans

n = 68

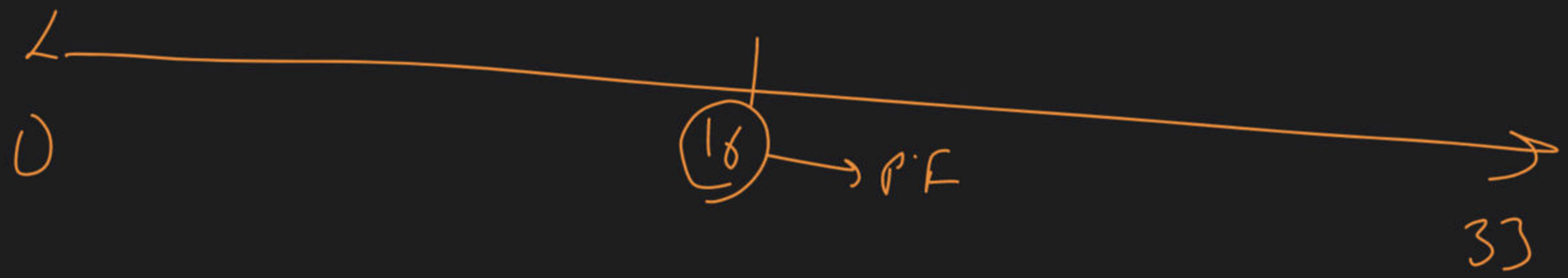
o/p → $\sqrt{68}$ → 8

(A) Search space
(II) predicate fun

s = 0
e = 68
} mid = 34

$34 \times 34 = 18$

$34 \times 34 > 68$ → left → e = mid - 1
e = 34 - 1
e = 33



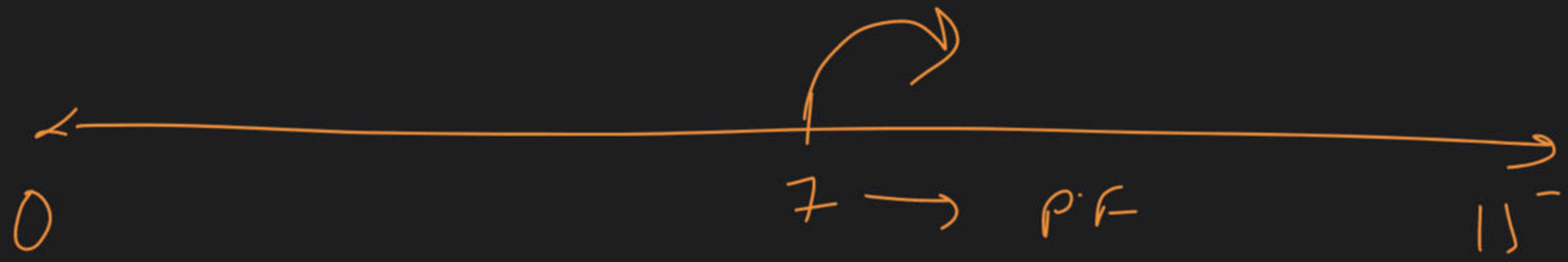
mid > 16

$16 \times 16 \rightarrow 256 = 68$

$256 > 68 \rightarrow$ left = e = mid - 1

e = 16 - 1

= 15



mid = $\frac{0+11}{2}$

7×7 = 49 <= 68

7×7 = 49 < 68 → min

s = mid+1

= 8

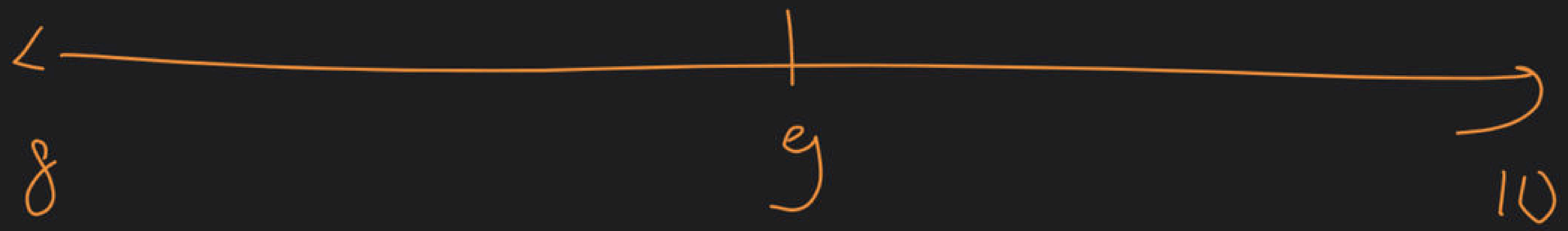
ans = 7

mid =

$11 \times 11 = 121$ ✗

$11 \times 11 > 88$ → left $r = 11 - 1$

$= 10$

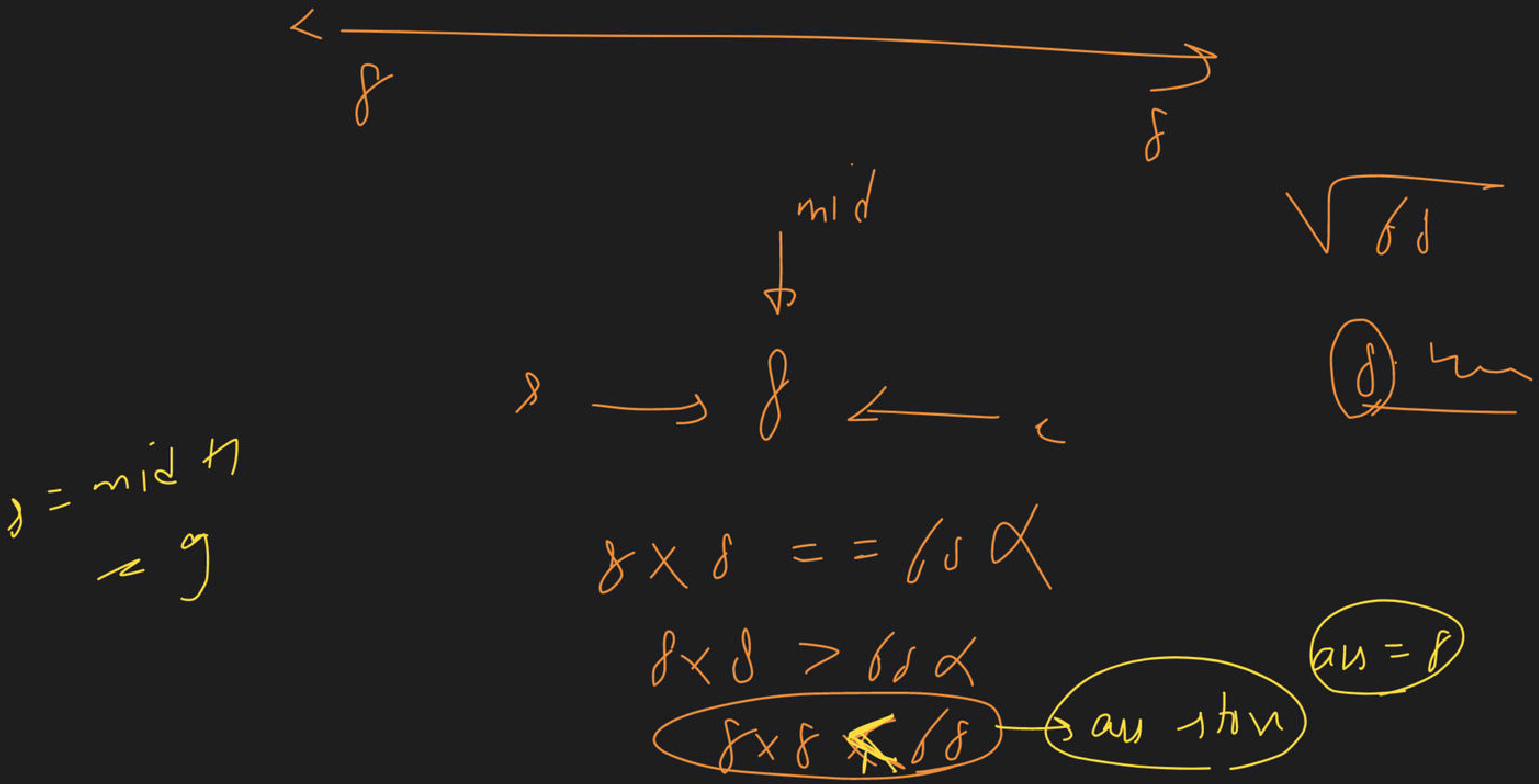


$9 \times 9 = 81$ ✗

$9 \times 9 > 81 \longrightarrow \;\longrightarrow\; e = mid - 1$

$= 9 - 1$

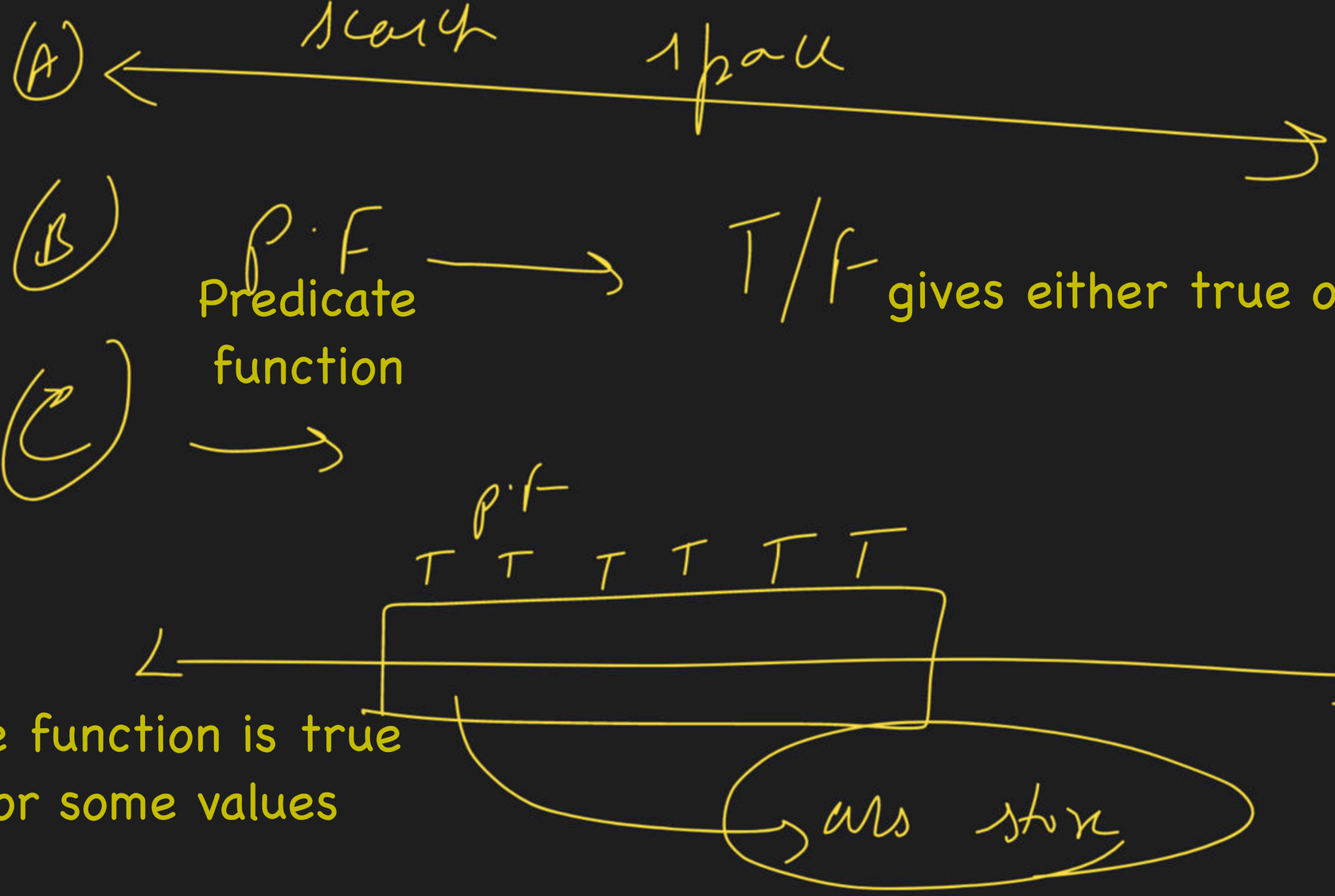
$= \boxed{8}$

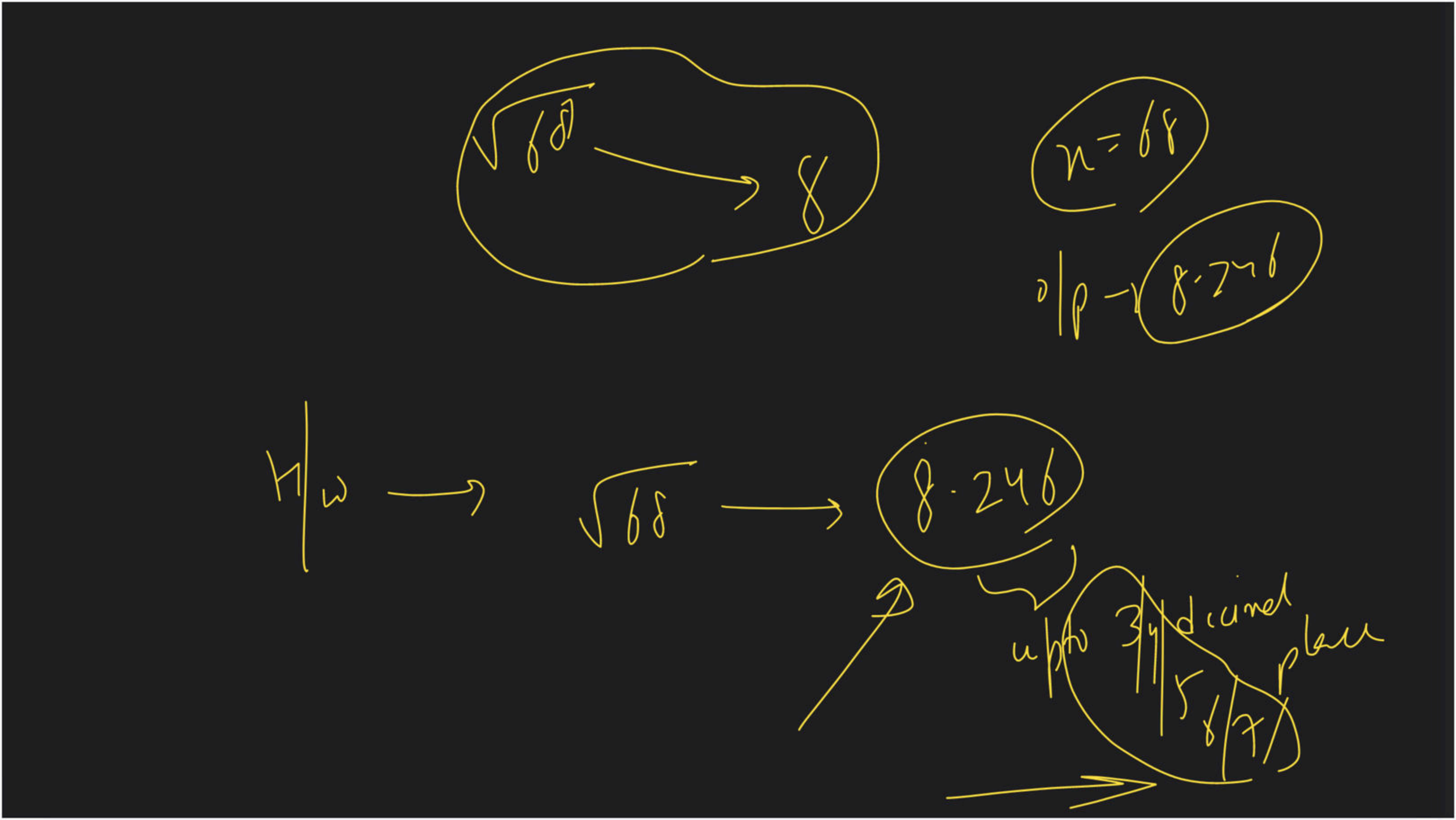


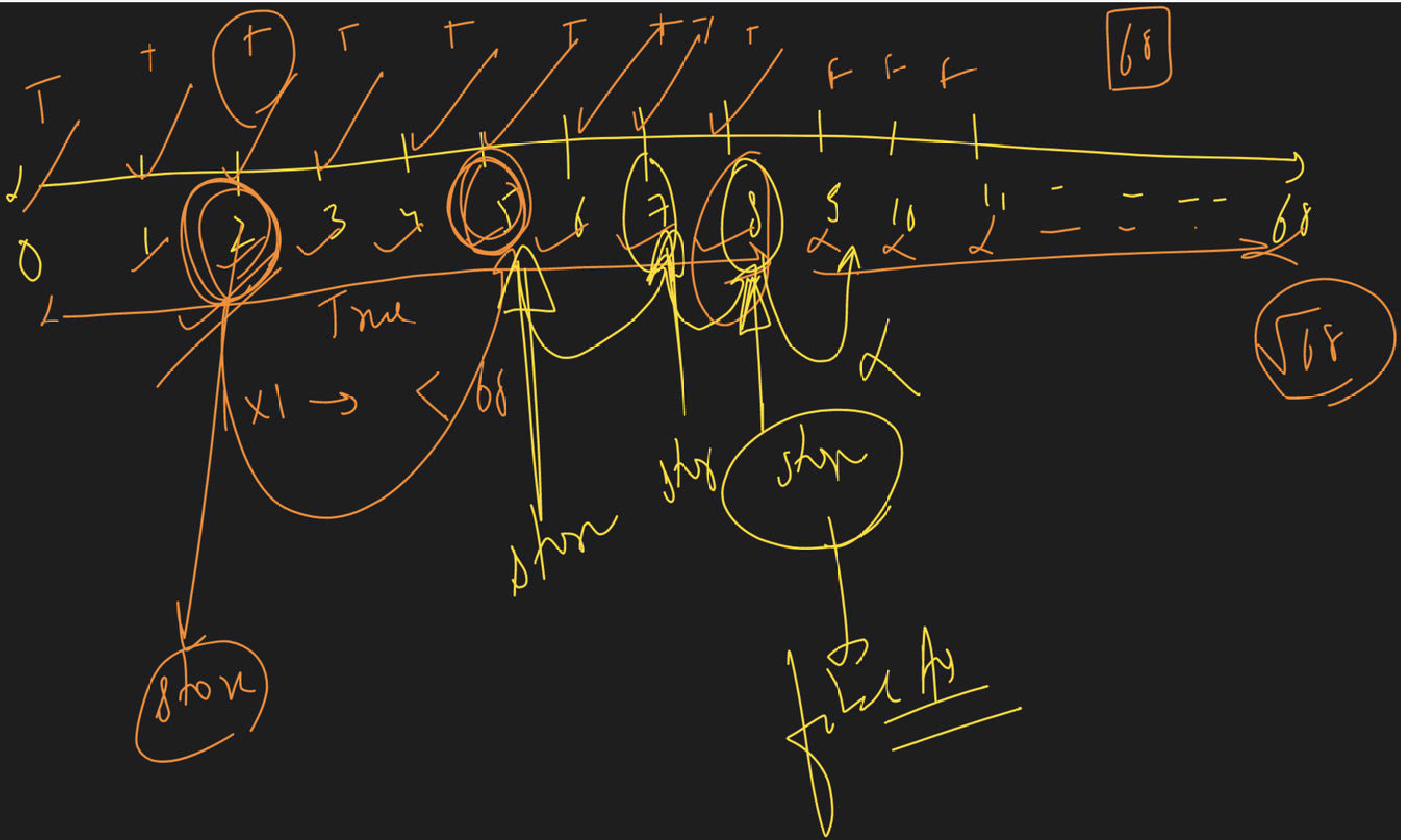
mid
mid +1
ans

Rutgers

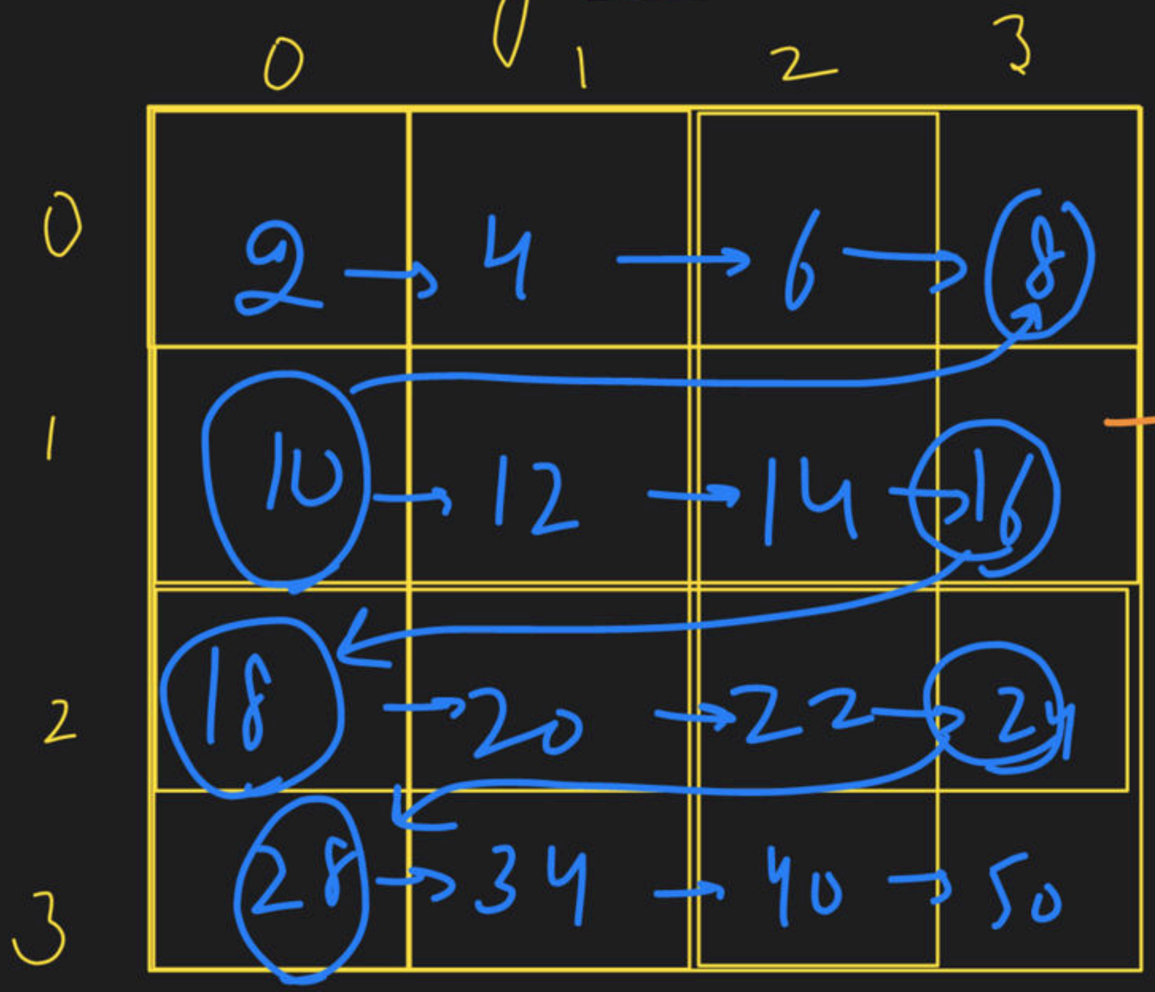
predicate f(): it tells weather the element upon which mid lies, is a valid answer or not




√68
8
n = 68
o/p → 8.246
I/w → √68 → 8.246
upto 3 decimal places

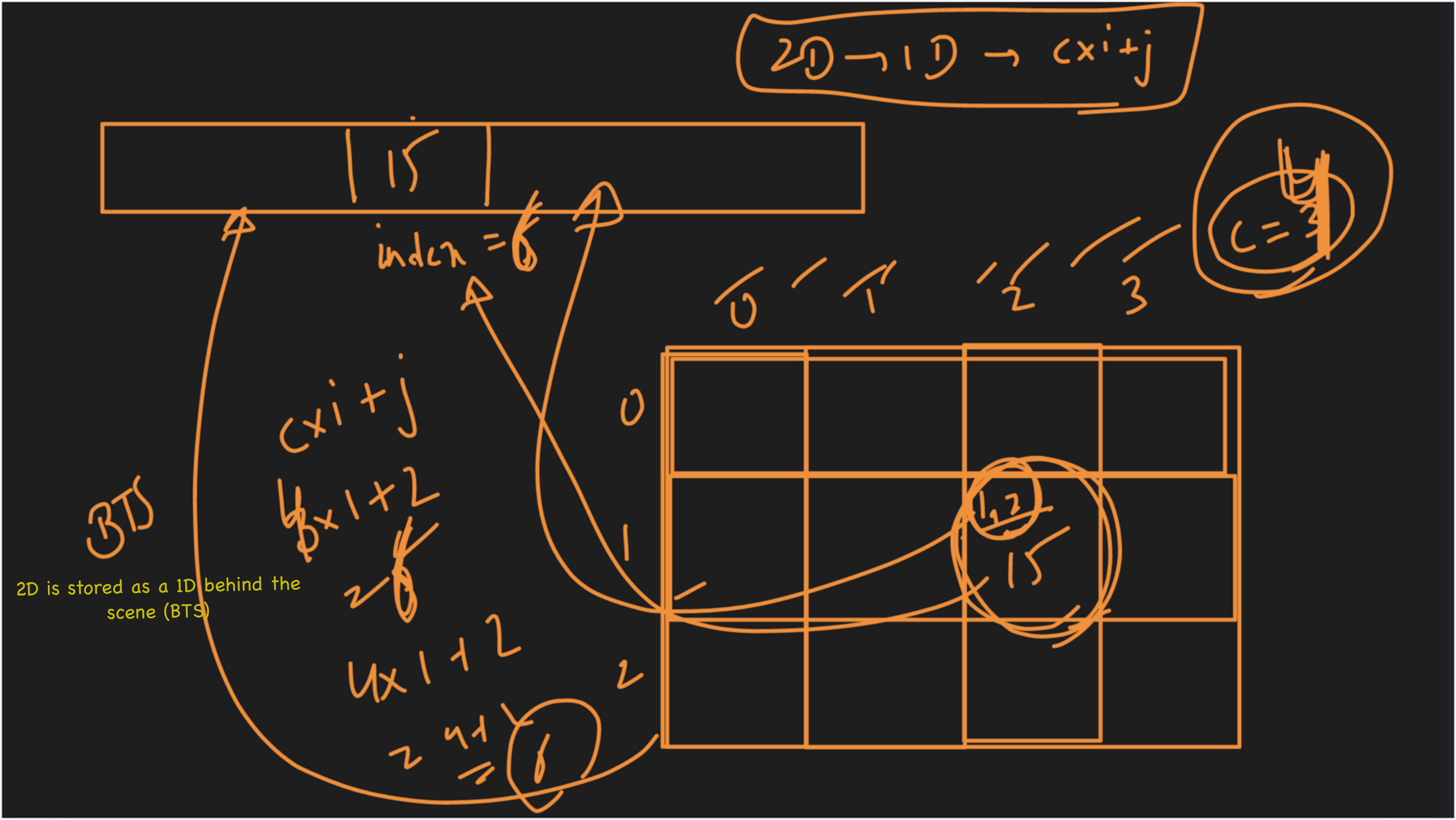

68
0 1 2 3 4 5 6 7 8 9 10 11 68
True
x1 → < 68
stop
stop
stop
stop
final Ans
√68

Binary Search on 2D array

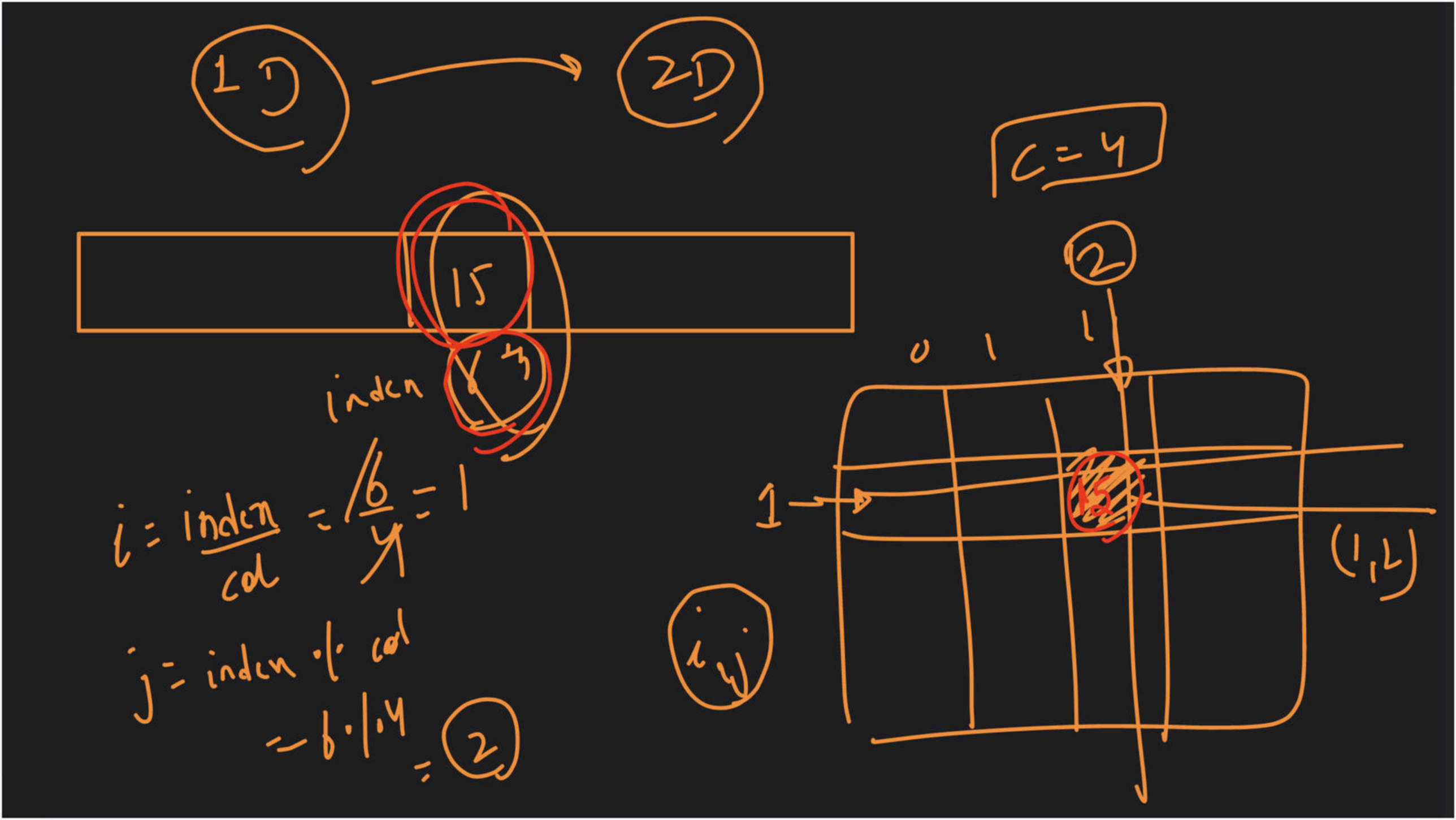


2D → 1D ⇒ c * i + j

1D → 2D ⇒ i : mid / c ; j : mid % c



2D is stored as a 1D behind the scene (BTS)


1D
2D
15
3
index
i = index / col = 6 / 4 = 1
j = index % col
= 6 % 4
= 2
i, j
C = 4
2
0
1
1
1
(1,2)

2D ⟶ 1D ⟶ (x i + j

1D ⟶ 2D

i = index / col

J = index % col

→ O(m x n)

→ O(log(m x n))