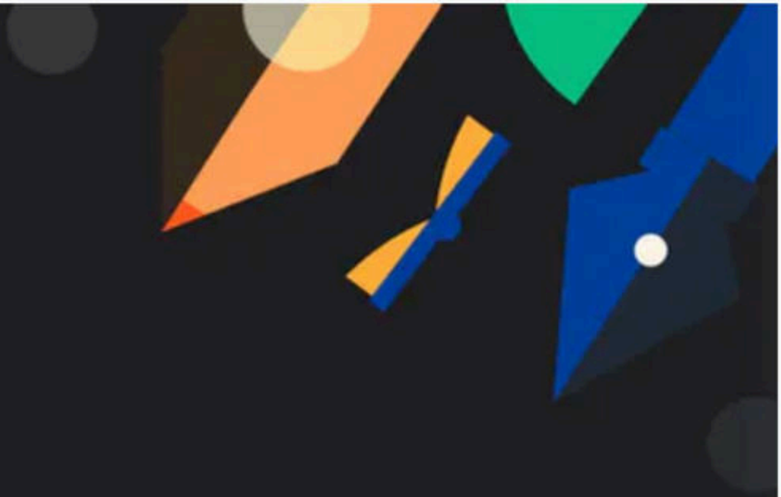


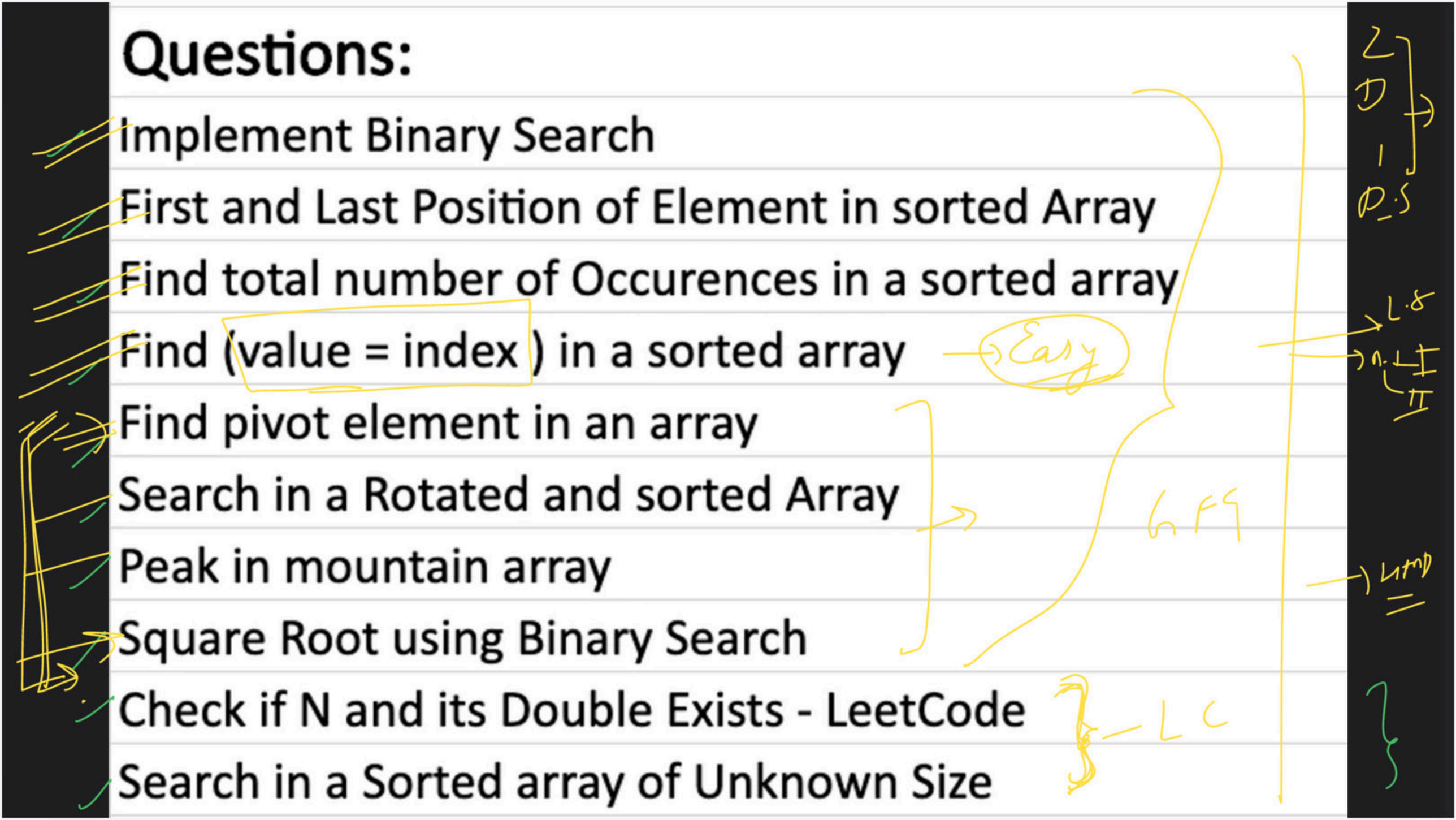
Binary Search

Foundation Course on Data Structures & Algorithm - Part I

Love Babbar • Lesson 10 • May 3, 2022

Questions:

- Implement Binary Search
- First and Last Position of Element in sorted Array
- Find total number of Occurences in a sorted array
- Find (value = index) in a sorted array
- Find pivot element in an array
- Search in a Rotated and sorted Array
- Peak in mountain array
- Square Root using Binary Search
- Check if N and its Double Exists - LeetCode
- Search in a Sorted array of Unknown Size



Binary Search :-

→ monotonic fn

T.C → O(log n)

pre-requisite

condn

Sorted order

n = 2^{10} → 1024 → B.S → 10 comp

1024 → L.S → 1024 comp

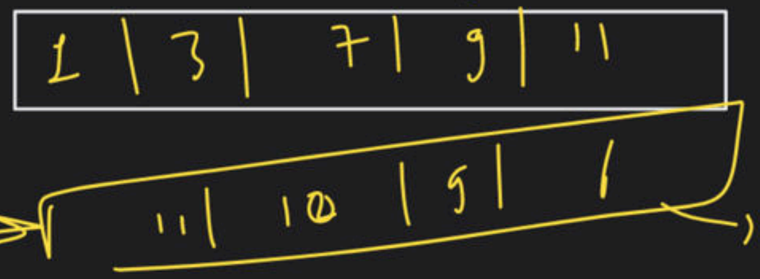
2^{100} → Bada → B.S → 100

2^{100} → value

Linear Search

T.C → O(n)

single traversed



B.S → O(log n) → How?

→ why?

arr[mid] = 9

search

5 == 15

condn

inc/dec

mid

mid

mid

i/p →

inc order

1>29

target = 15

↓

present or absent

app #1 → Linear Search → O(n) →

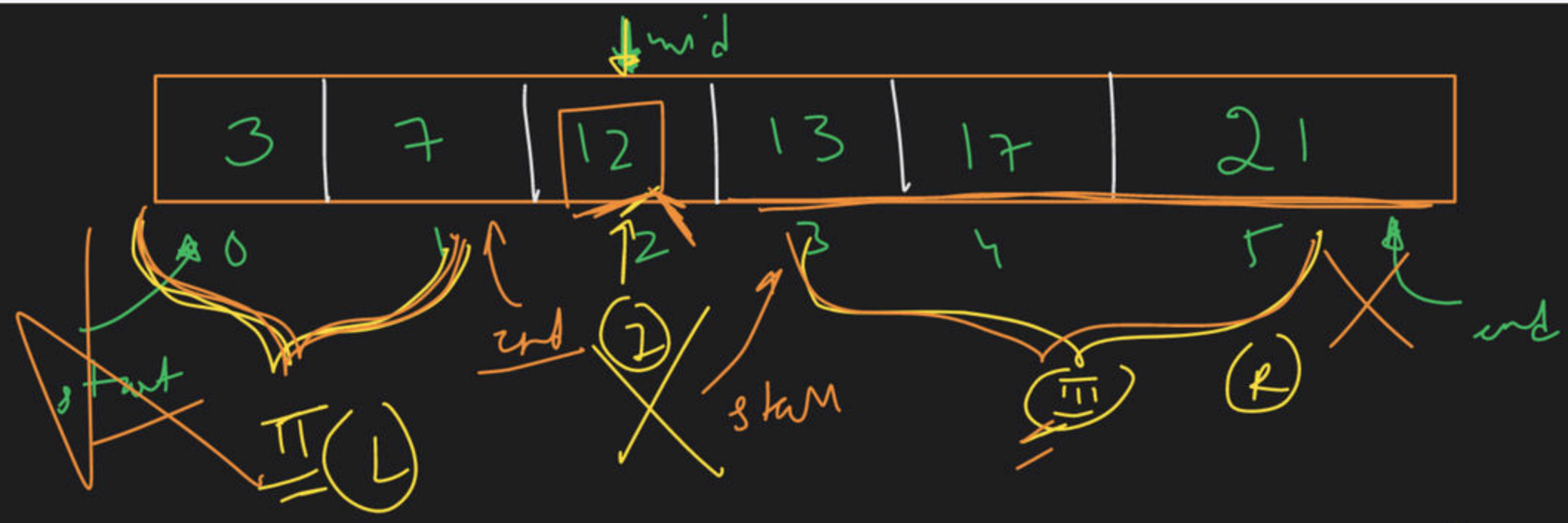
app #2 → Binary Search

$$mid = \left(\frac{s+e}{2}\right)_z \quad \left(\frac{0+7}{2}\right)_z$$

```
if( arr[mid] == target)
    return true;

if( target < arr[mid])
    e = mid - 1;

if ( target > arr[mid])
    s = mid + 1;
```



arr(mid) == target

12 == 17 → F

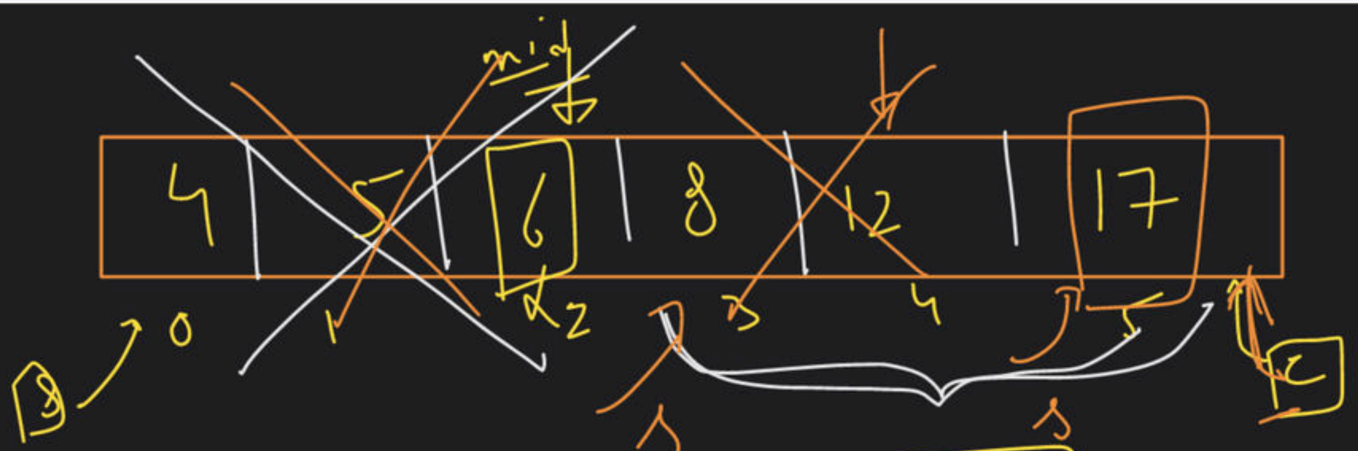
if (target < arr(mid)) (Left)

end = mid - 1

if (target > arr(mid)) → Right

start = mid + 1

→ Code



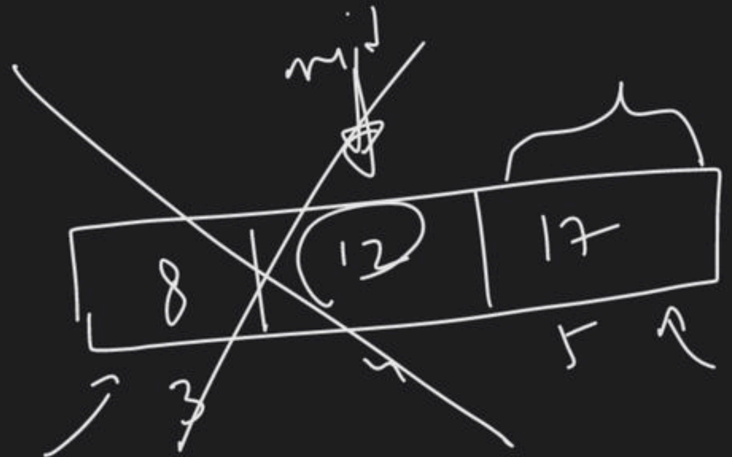
target = 17

s = 0
e = 5 → mid = $\frac{0+5}{2}$ = 2 → 6 == 17 → False

Left → 17 < 6 → False

Right → 17 > 6 → True

s <= e

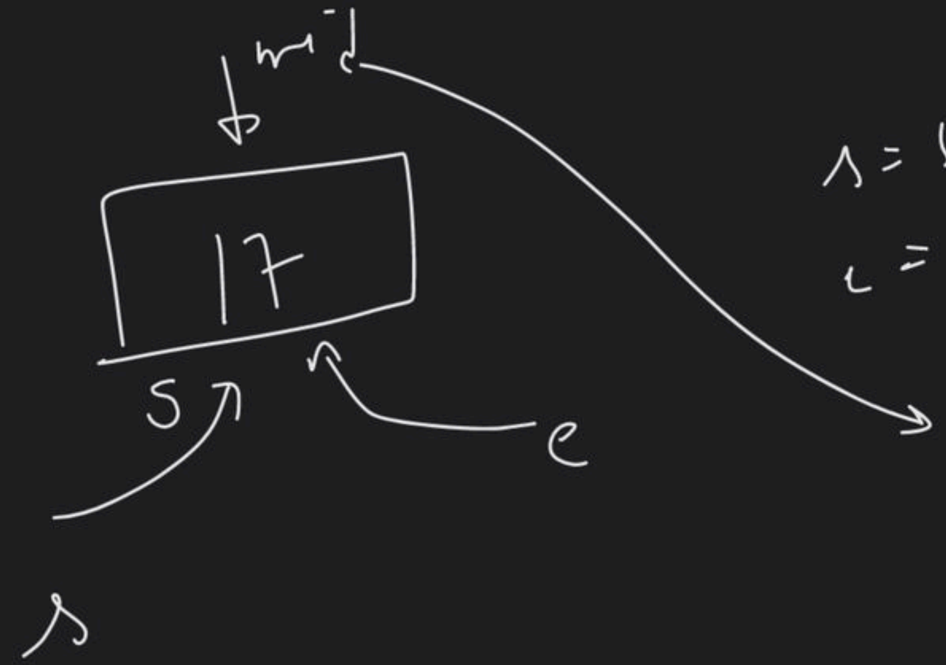


s = 3
e = 5 → mid = $\frac{3+5}{2}$ = (4)

→ 12 == 17 → false

Left, 17 < 12 → False

Right → 17 > 12 → T

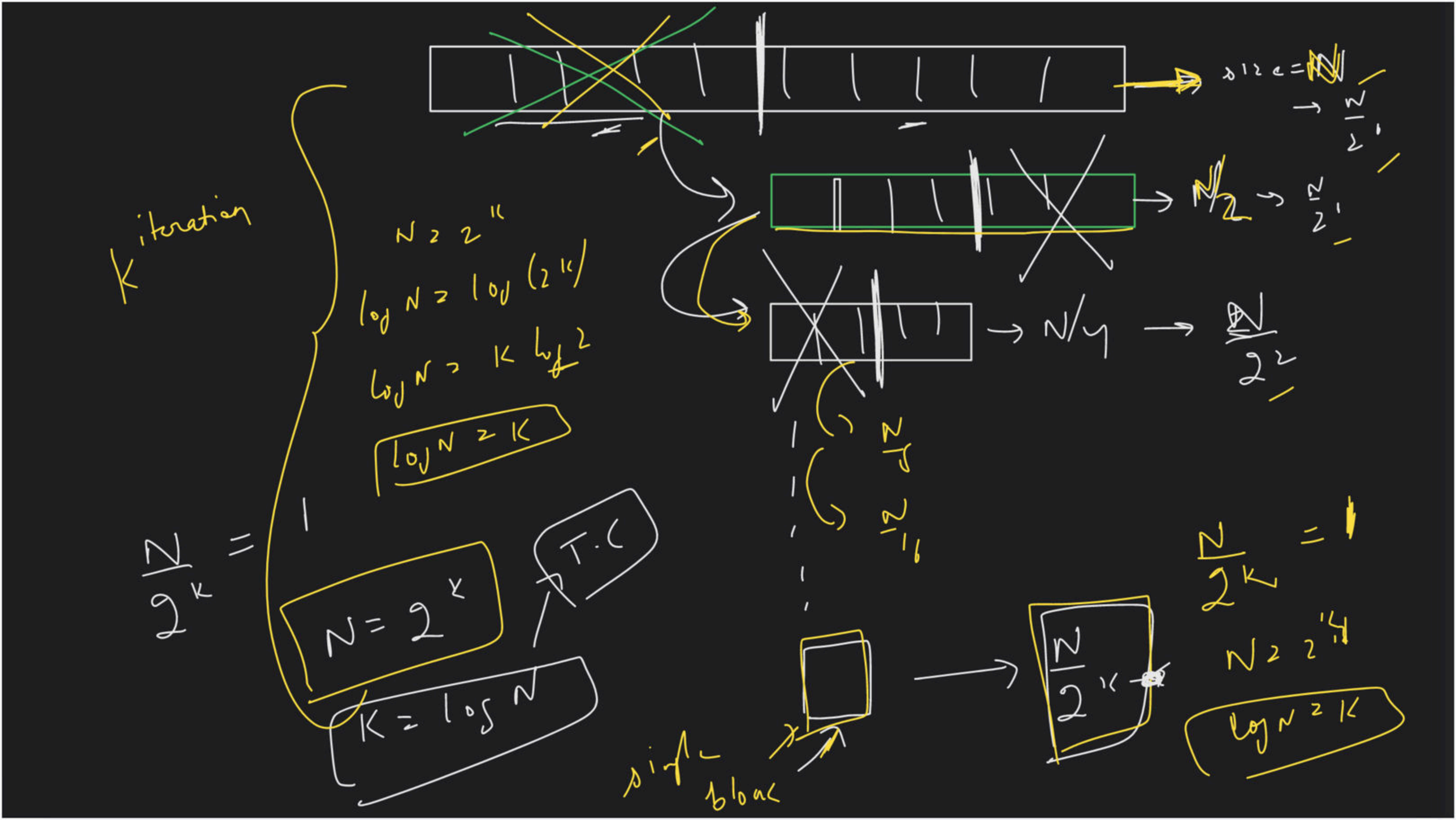


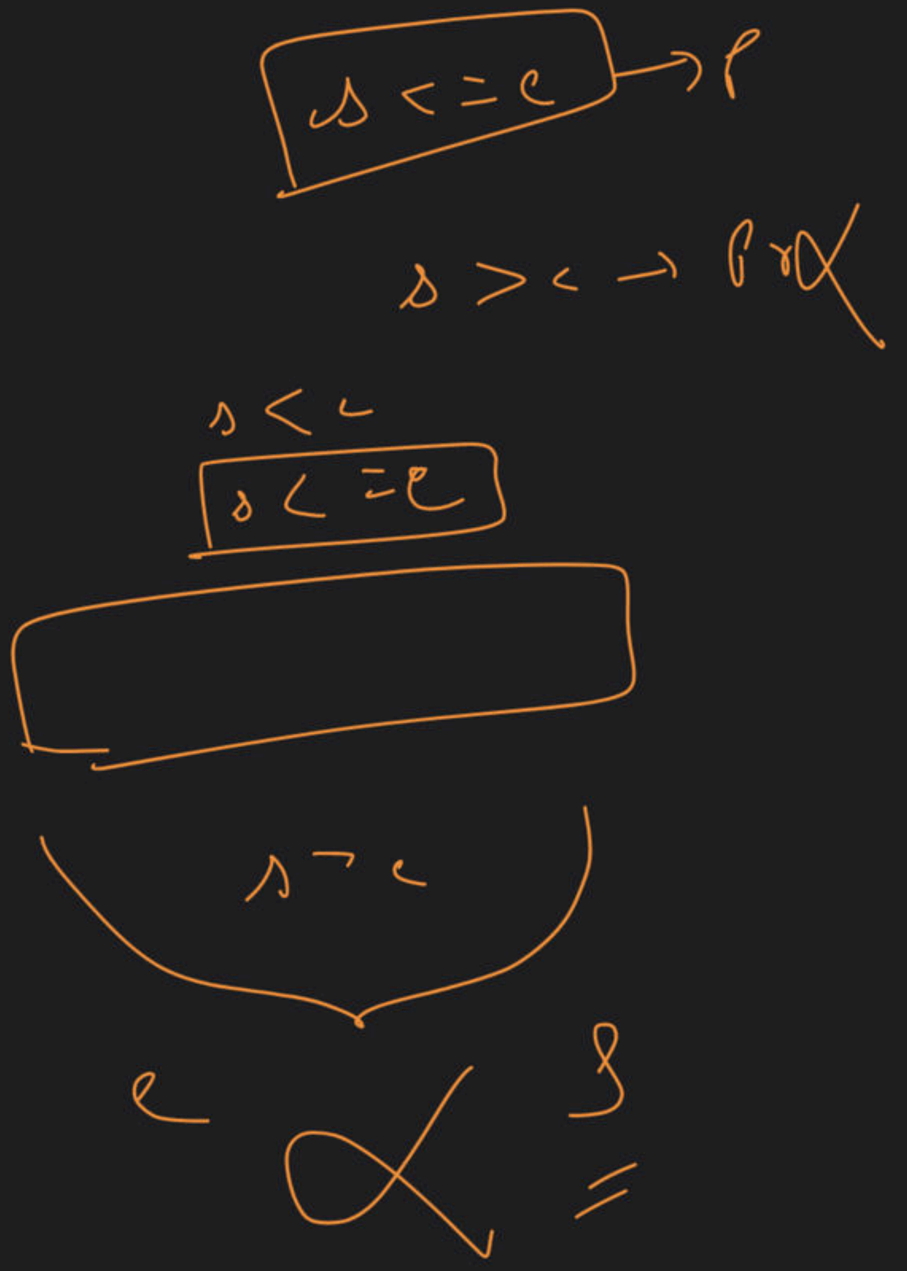
$s = 5$ $e = 5$ → $mid = (\frac{5+5}{2})_2 = 5$

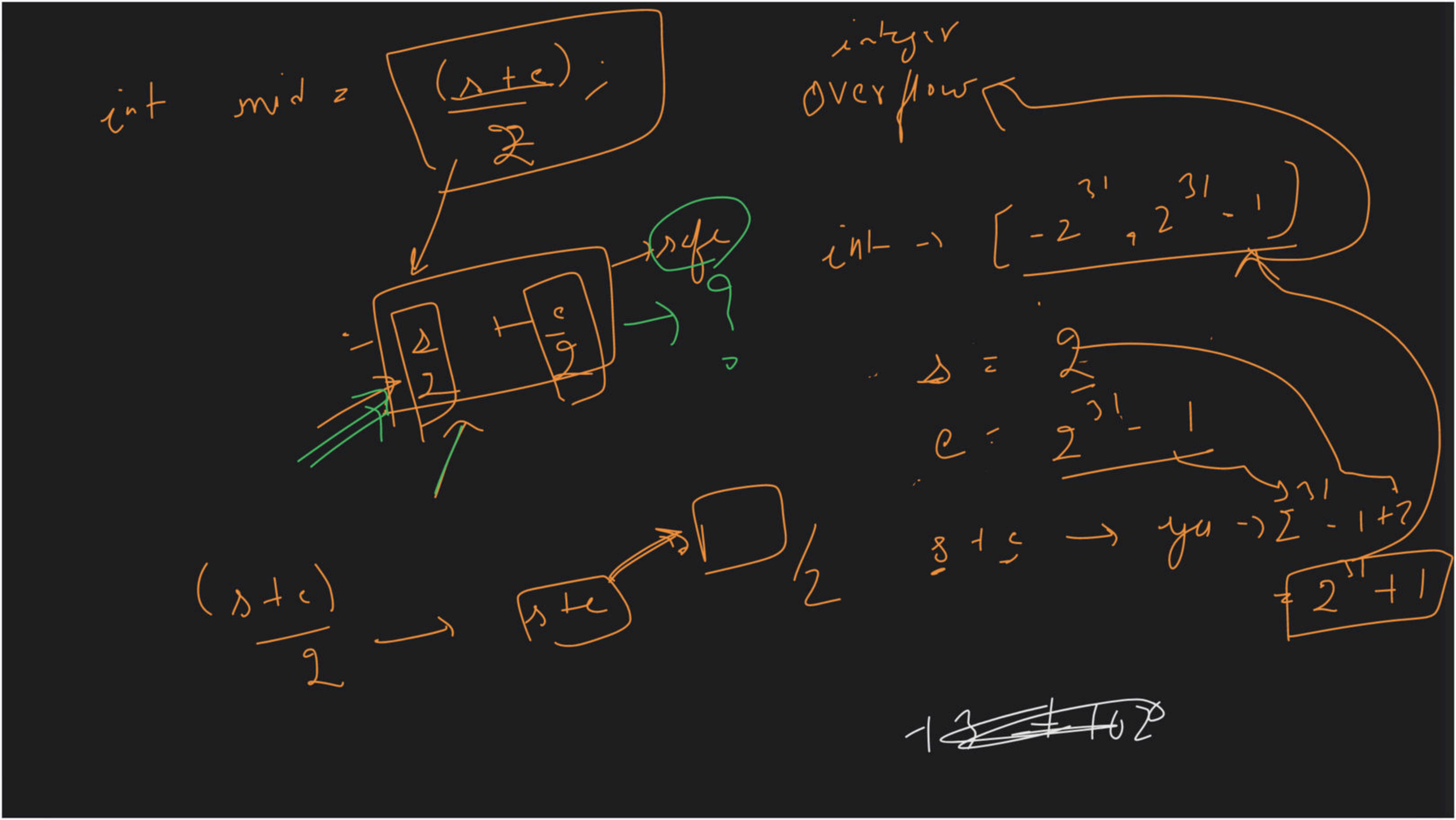
arr(mid) == target

17 == 17 → TRUE

return TRUE


K iteration
N = 2^K
log N = log (2^K)
log N = K log 2
log N = K
N/2^K = 1
N = 2^K
T.C
K = log N
size = N
N/2
N/4
N/2^K
single block
N/2^K = 1
N = 2^K
log N = K



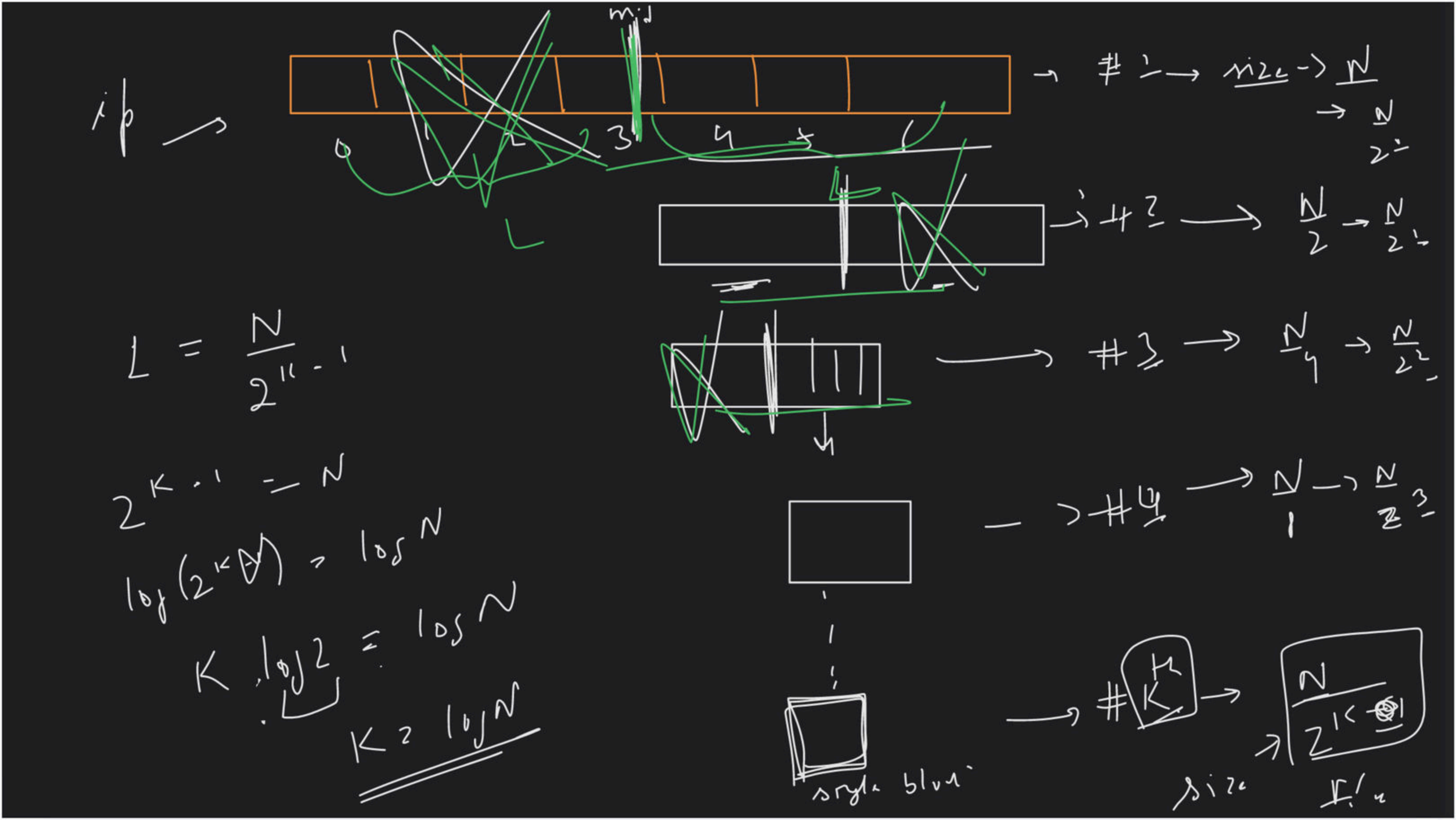

int mid =
(s+e)/2;
integer
overflow
int -> [-2^31 , 2^31 - 1]
s = 2
e = 2^31 - 1
s+e → yu -> 2^31 - 1 + 2
= 2^31 + 1
(s+e)/2
s+e
/2

$$mid = s + \frac{(e - s)}{2}$$

$$= s + \frac{(e - s)}{2}$$

$$= s + \frac{e}{2} - \frac{s}{2}$$

$$= \frac{s}{2} + \frac{e}{2} = \frac{s + e}{2}$$



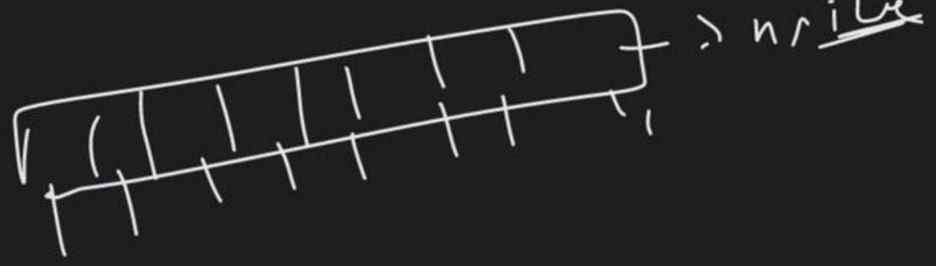
$$L = \frac{N}{2^{K-1}}$$

$$2^{K-1} = N$$

$$\log(2^{K-1}) = \log N$$

$$K \cdot \log 2 = \log N$$

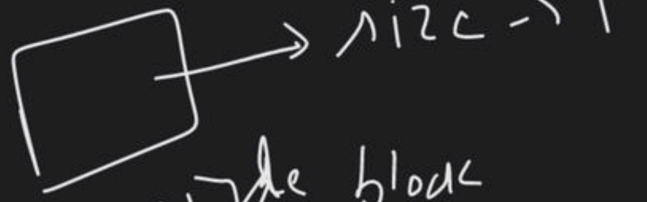
$$K = \log N$$



Concept

Binary Search

adv ->



single block

length of array

4 bytes

$mid = \frac{s}{2} + \frac{e}{2}$ → good or not?

$s + \left(\frac{e - s}{2}\right)$ → good

vector → stl video

11th 12th

IP

2^{1000}

L.S 2^{1000}

B.S 1000

B.S → what → no of c

→ why

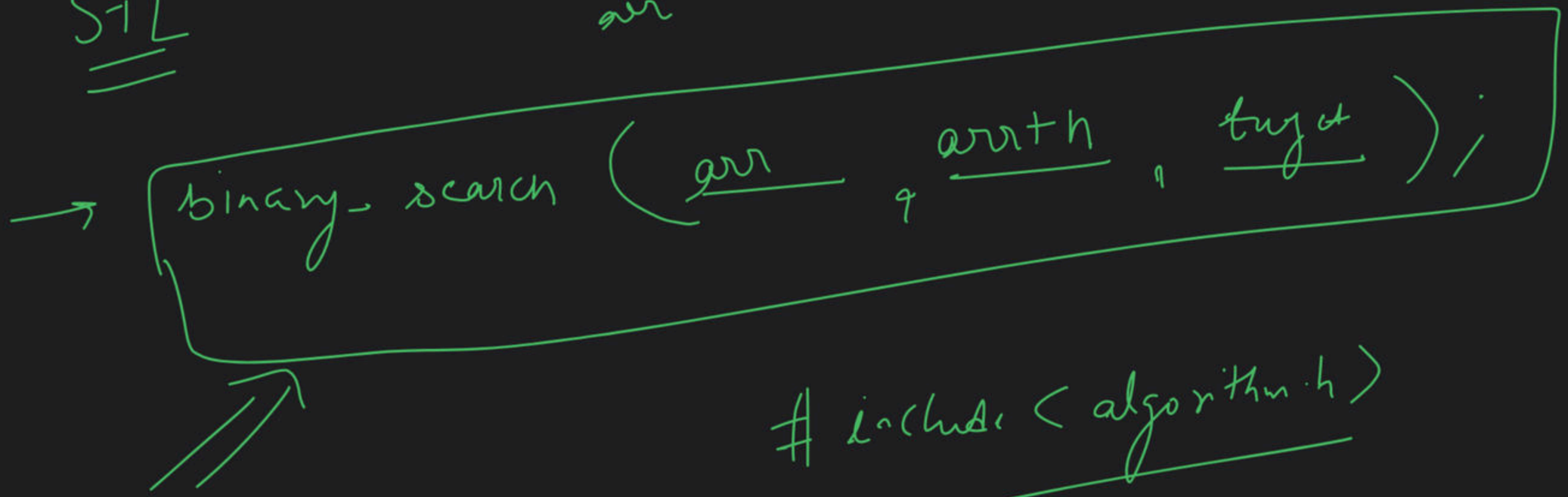
→ adv → O(log n)

→ Impl →

→ T.C → O log n

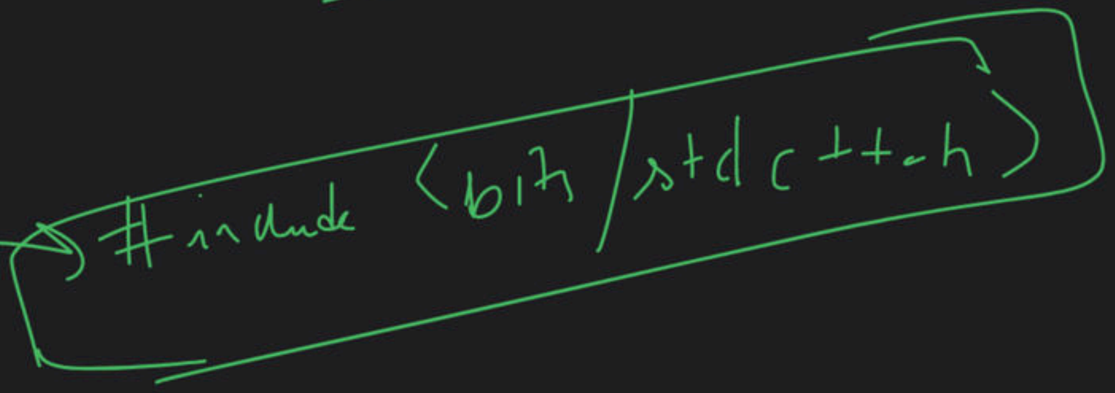
→ S.C → O(1)

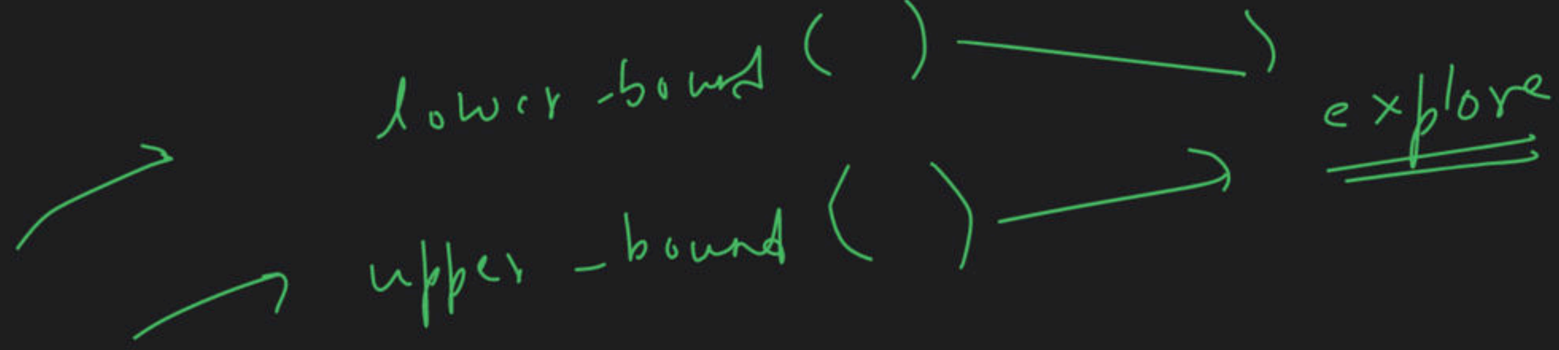
STL

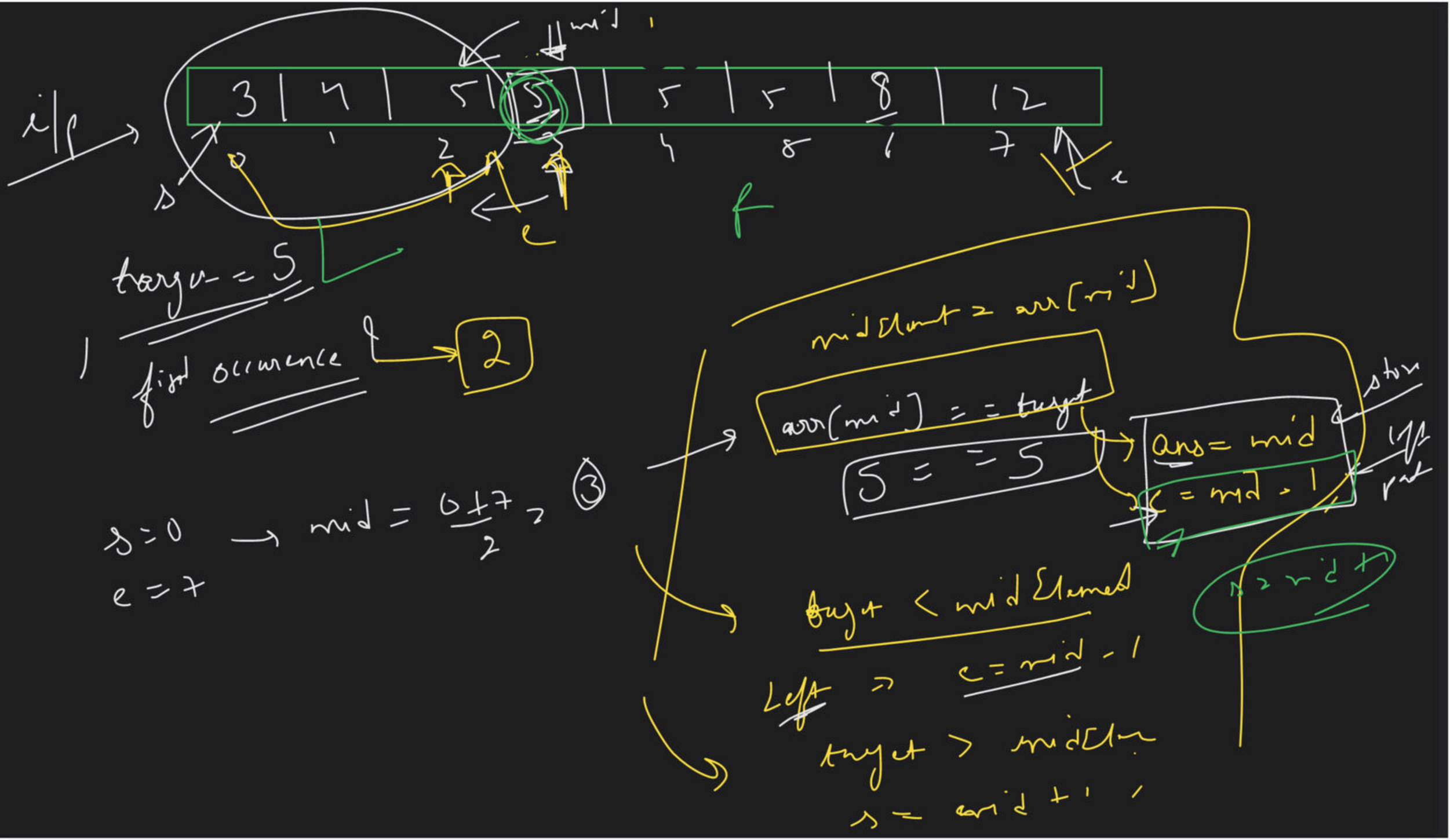


#include <algorithm.h>

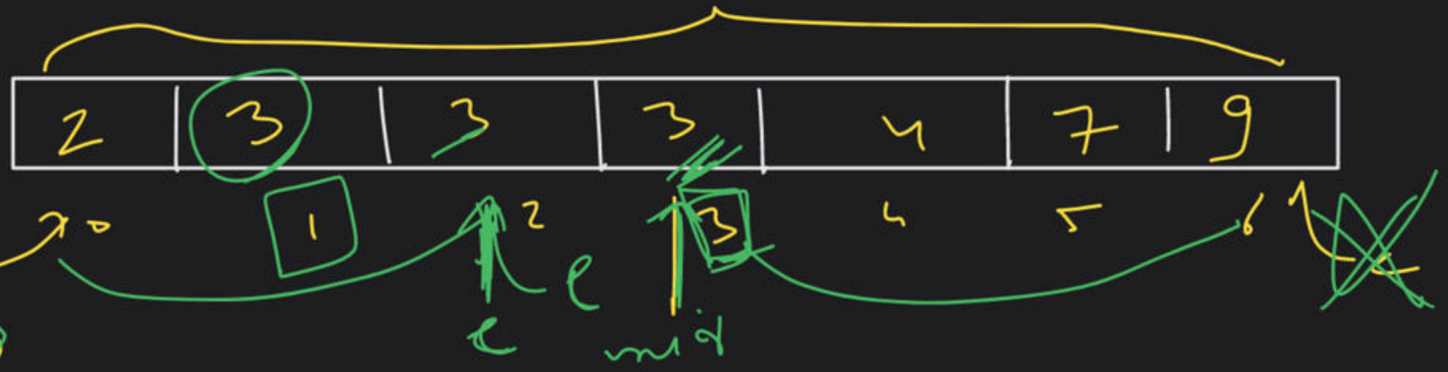
Kinda




lower_bound( )
upper_bound( )
explore


i/p
3 4 5 5 5 5 8 12
0 1 2 3 4 5 6 7
s
e
mid
f
target = 5
find occurence
2
s=0 → mid = 0+7/2 = 3
e=7
midElement = arr[mid]
arr[mid] == target
5 == 5
ans = mid
e = mid - 1
store
right
target < midElement
Left → e = mid - 1
target > midEle
s = mid + 1

ans = ~~-1~~ ~~3~~ 1



target = 3

s = 0
e = 6 → mid = $\frac{(0+6)}{2}_2$ ③

left → e = mid - 1

right → s = mid + 1

3 == 3

↳ ans = 3

↳ e = mid - 1 = 3 - 1 = 2

mid = $\frac{0+2}{2}_2$ = 1

s = 0
e = 2

3 == 3

↳ ans = mid

↳ left → e = mid - 1

e = 1 - 1 = 0

s = 0

e = 0 → mid → 0

2
0

2 == 3 → F

3 < 2 → F

3 > 2 → T

↳ right

s = mid + 1

s = 0 + 1 = 1

s = 1

e = 0

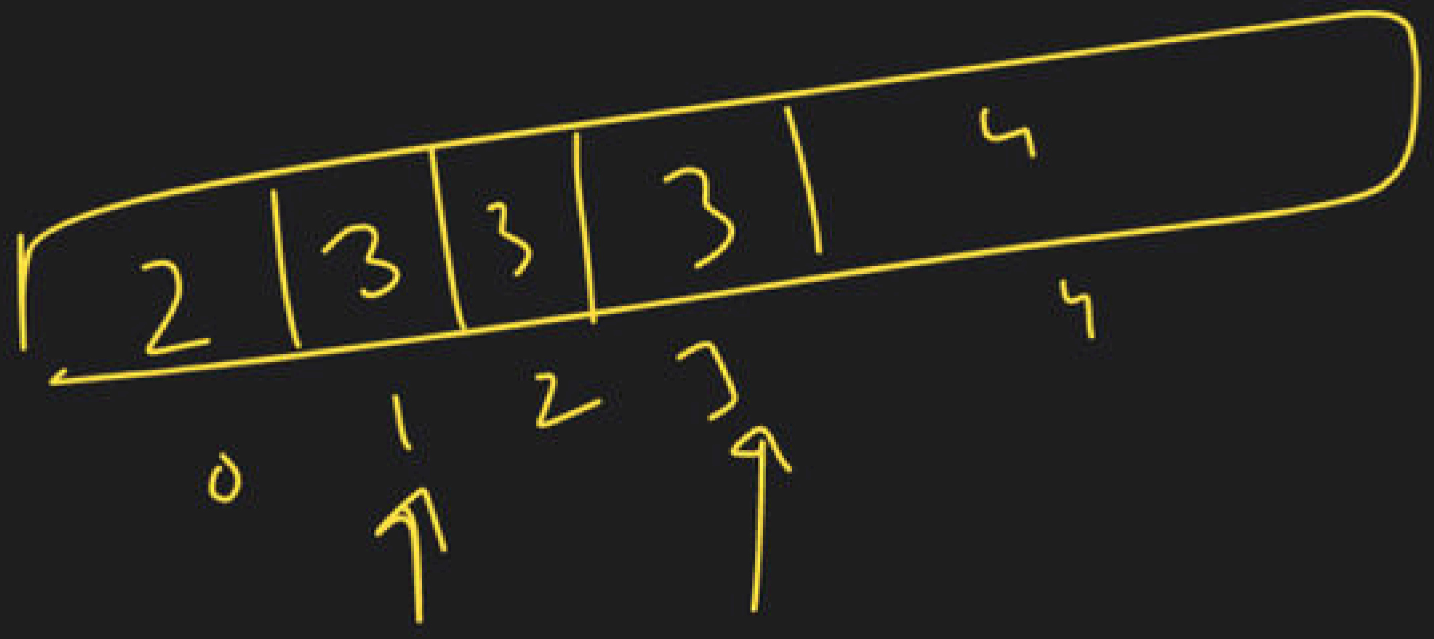
s > e ∴ sab khatam hai

→ find first / last occ of element in sorted arr

→ find total no of occurrence of element in sorted array

#1 Linear search → O(n)

#2 last - first + 1 → O(log n) → Binary search



→ BS

→ STL → BS
→ LB →
→ UB →

→ First

→ Last

→ Total no of occur

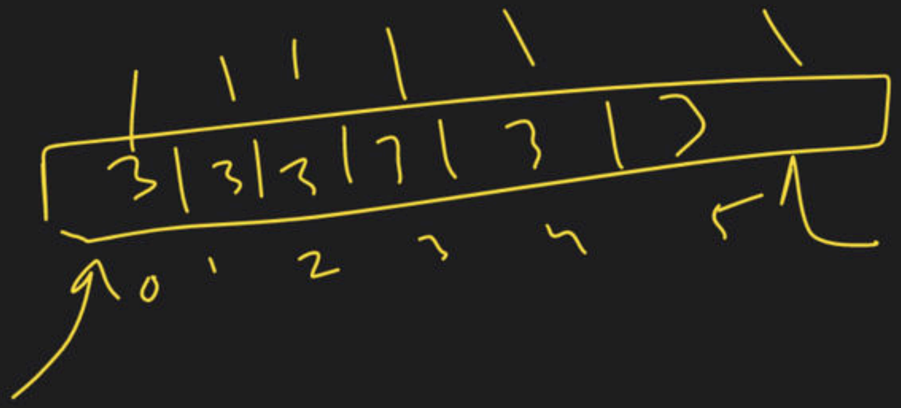
Last = 3

first = 1

Last − first

= 3 − 1 = 2 + 1

= 3



last − first + 1

5 − 0 + 1

= 6

→ find fixed point in a sorted array

(value = index)

i/p → { -10, -5, 0, 3, 7 }

index: 0, 1, 2, 3, 4

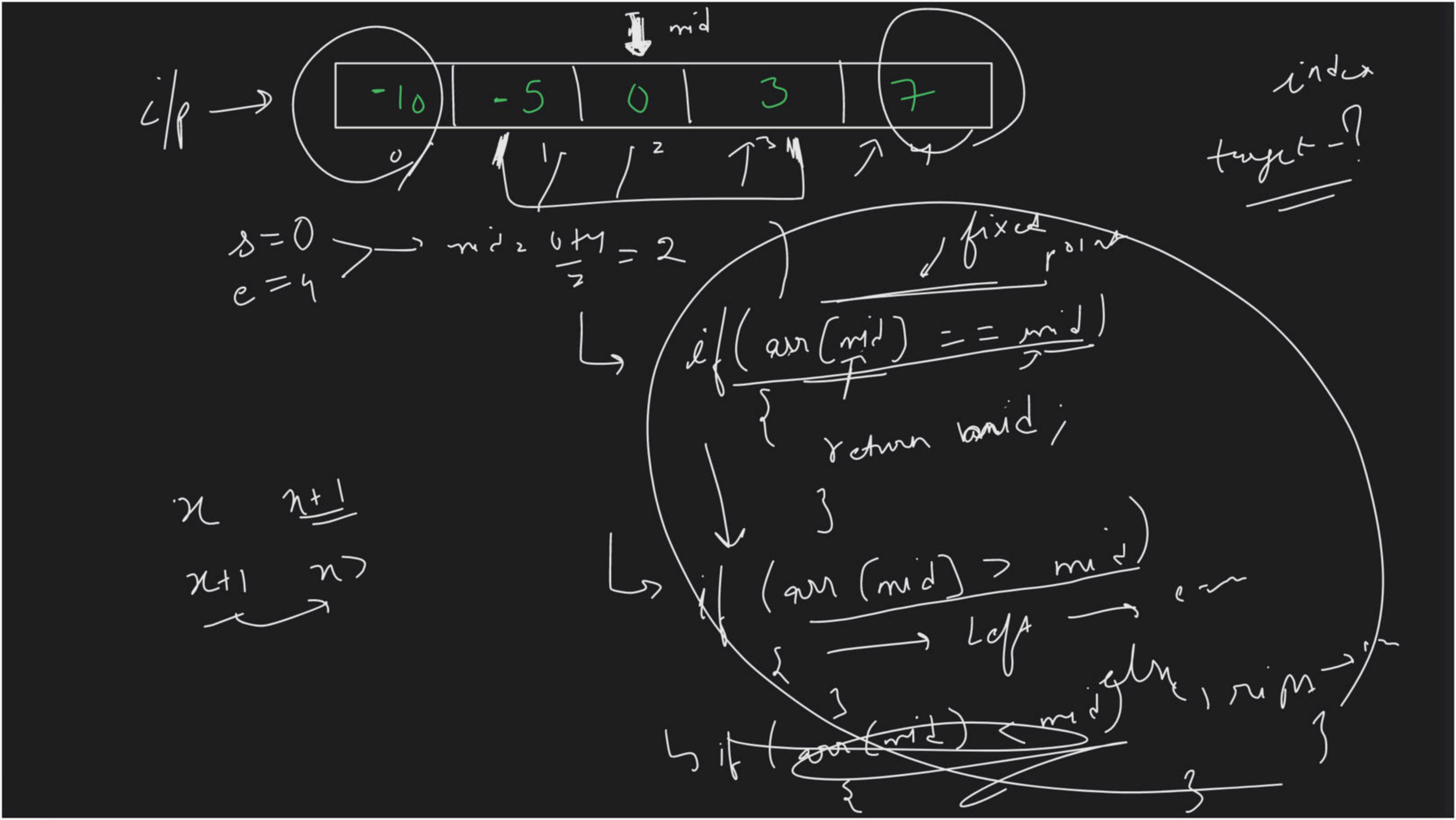
o/p → 3

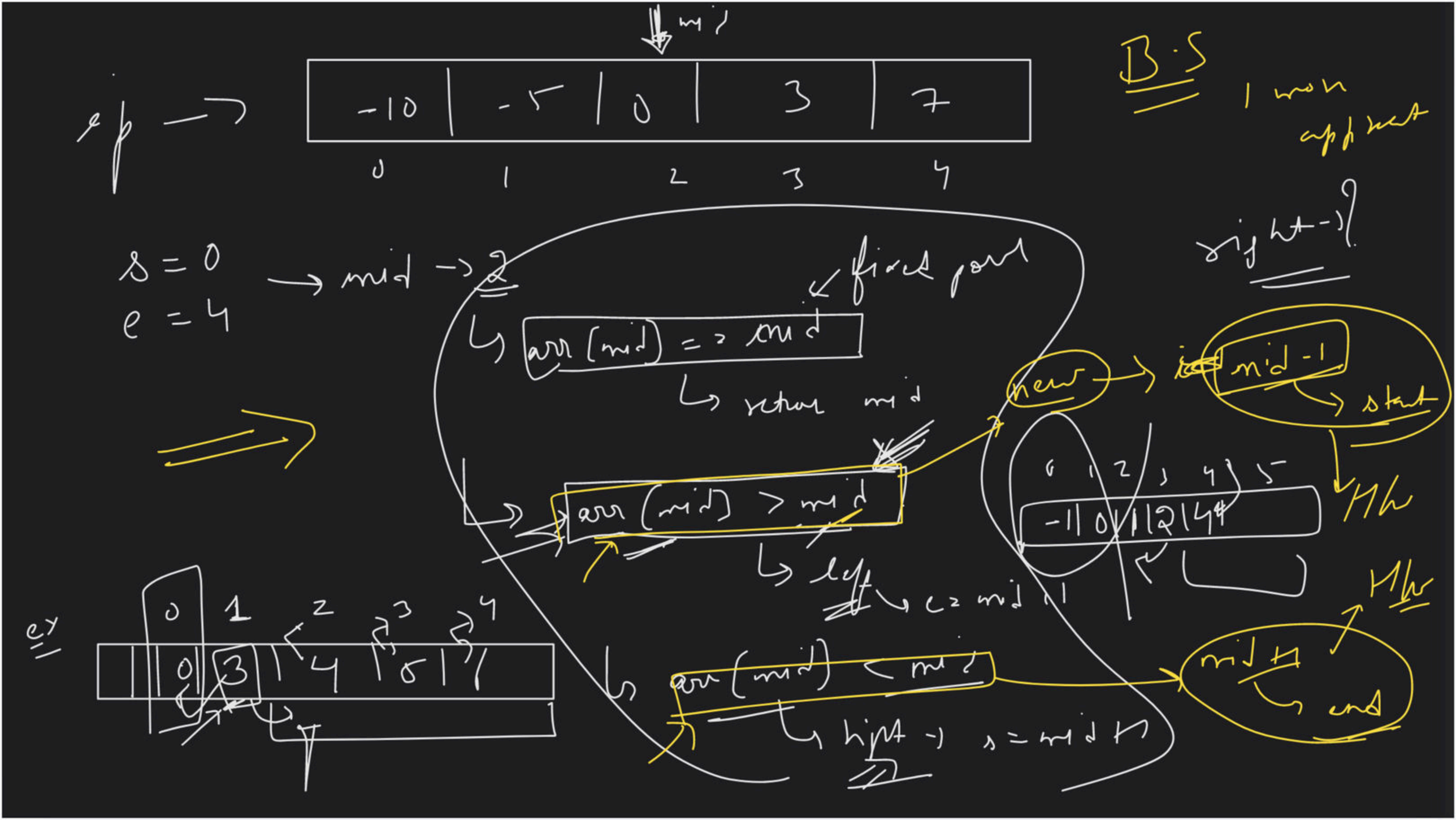
#1 Brute force

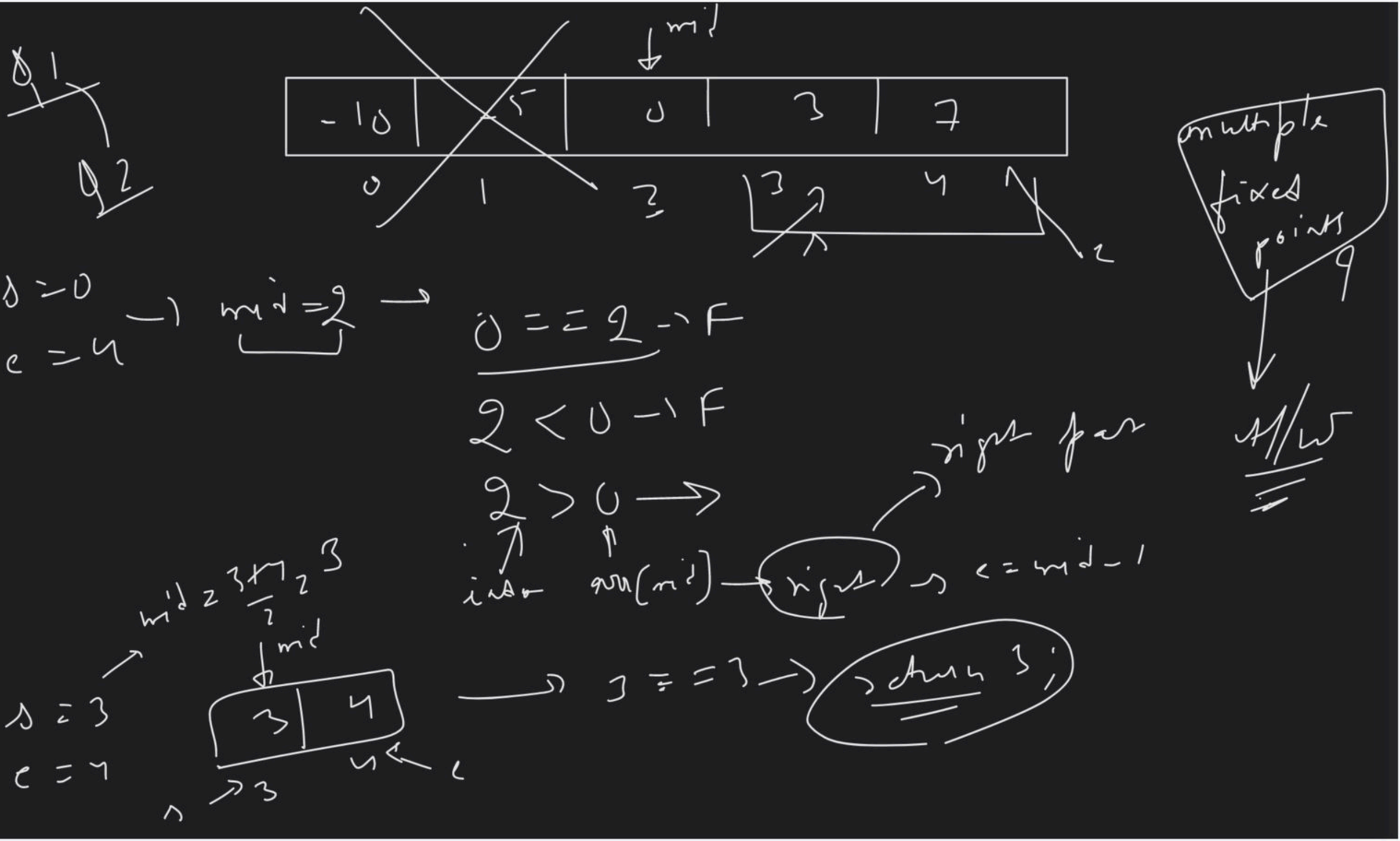
↳ O(n)

→ Binary Search

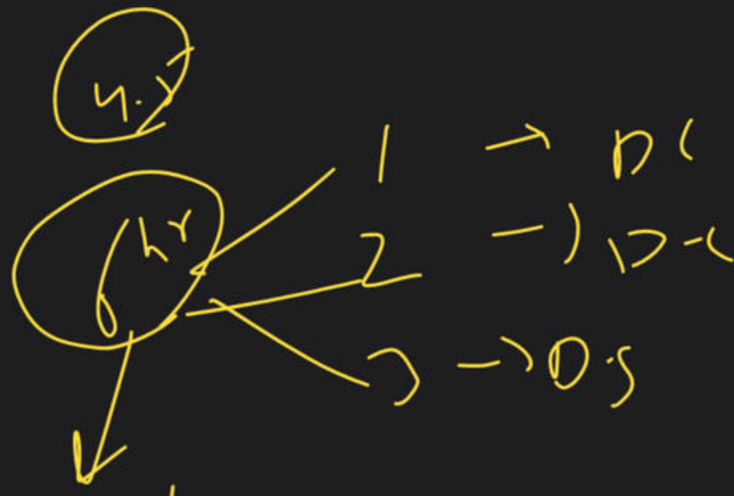
↳ O(log n)


mid
i/p →
-10
-5
0
3
7
0
1
2
3
4
index
target = ?
s = 0
e = 4
→ mid = (0+4)/2 = 2
fixed point
if (arr[mid] == mid)
{
return mid;
}
if (arr[mid] > mid)
→ Left → e = mid - 1
if (arr[mid] < mid)
s = mid + 1
n
n+1
n+1
n


mid
-10
-5
0
3
7
0
1
2
3
4
B.S
1 more application
s = 0
e = 4
mid -> 2
fixed point
arr[mid] == mid
return mid
right ->
arr[mid] > mid
left
e = mid - 1
new
mid - 1
start
mid
arr[mid] < mid
right
s = mid + 1
mid + 1
end
ex
0
3
4
5
7


Q1
Q2
mid
-10
5
0
3
7
0
1
2
3
4
multiple fixed points
9
11/15
s=0
e=4
mid=2
0==2 -> F
2<0 -> F
2>0 ->
right part
arr(mid)
right
<=mid-1
mid=3+4/2=3
mid
s=3
e=4
3
4
3==3 -> return 3;

TLE



o hr → D.S