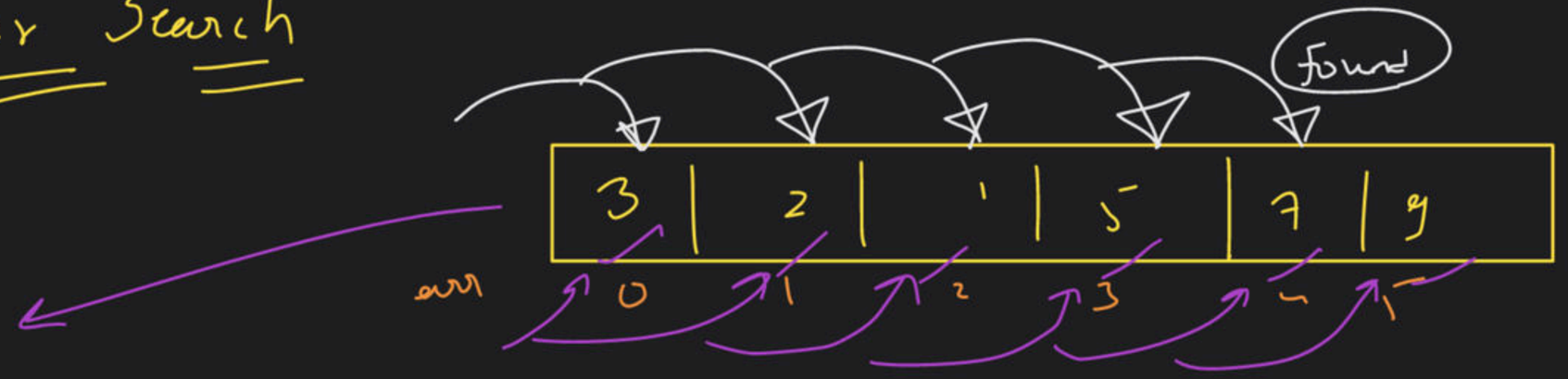
Searching and Sorting Class - I

Special class

Love Babbar • Feb 17, 2023

Linear Search



Code:-

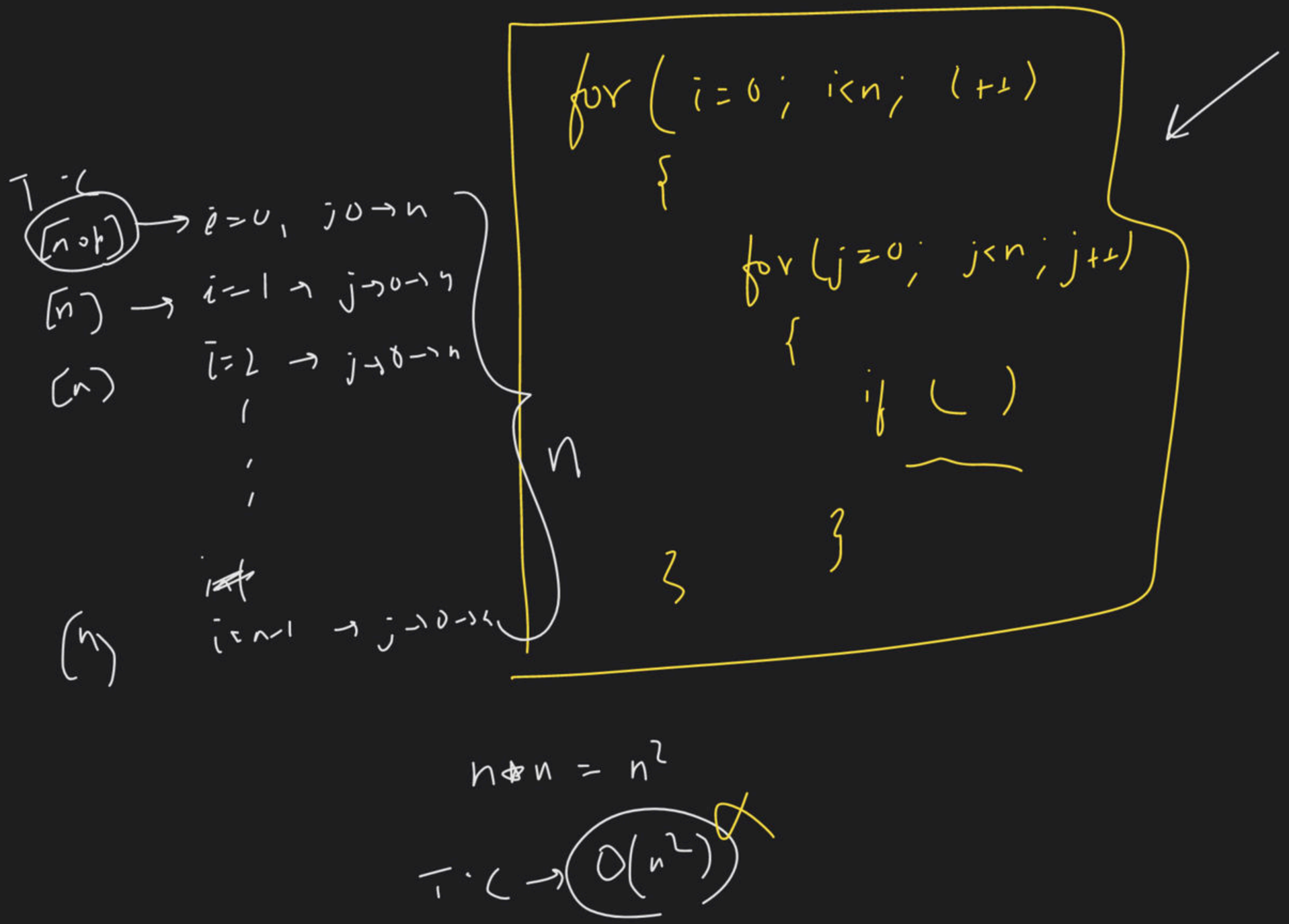
```
for (int i = 0; i<n; i++)
{
    if (arr[i] == target)
        cout << "found";
}
```

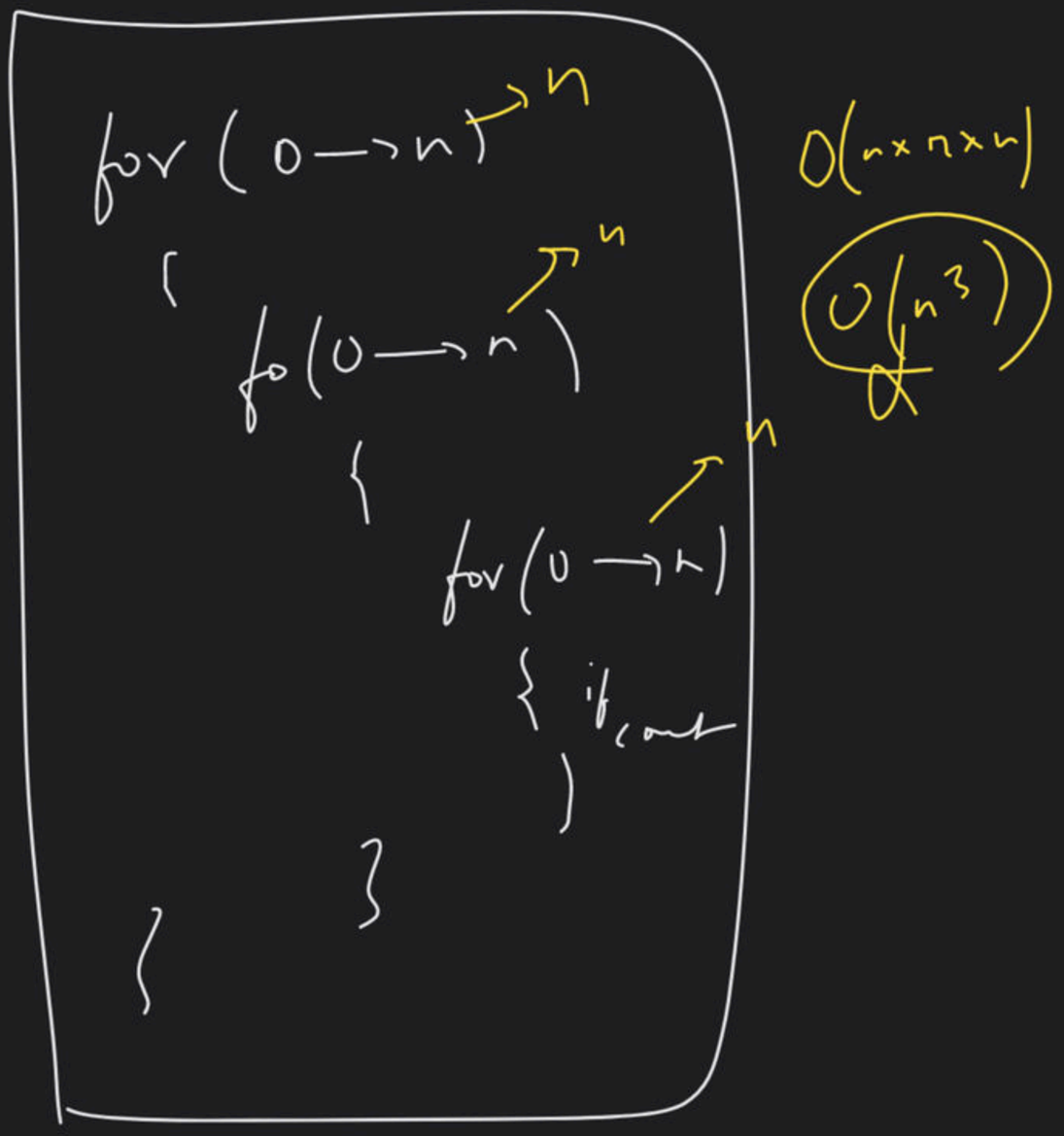
T.C → O(n)

Linear Complexity

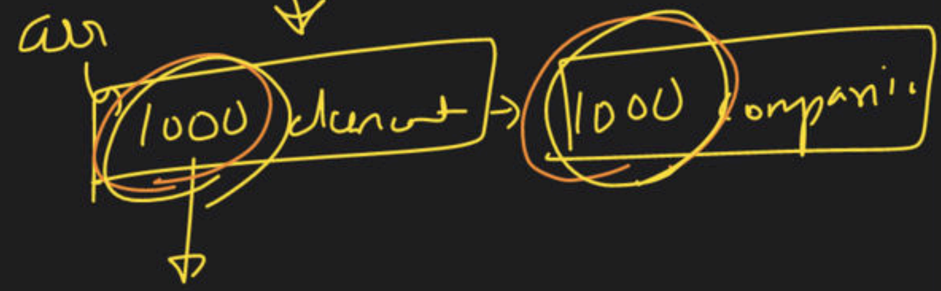
target = 7

target = 10


for (i=0 ; i<n ; i++)
{
for (j=0 ; j<n ; j++)
{
if ( )
}
}
T.C
(n+1)
(n) → i=1 → j→0→n
(n) i=2 → j→0→n
(n) i=n-1 → j→0→n
n
n*n = n²
T.C → O(n²)


for (0 → n) → n
{
for (0 → n) → n
{
for (0 → n) → n
{ if (cond
)
}
}
O(n × n × n)
O(n³)

→ Searching algo:- Linear Search → T.C → O(n)



100000 element :) 100000 comparison

What

Binary Search → fast → How

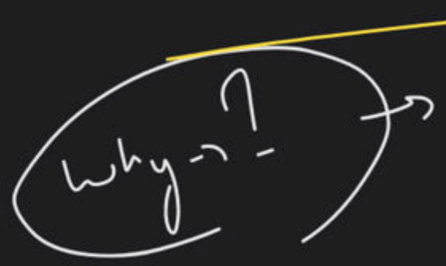
why arr → 1000 element

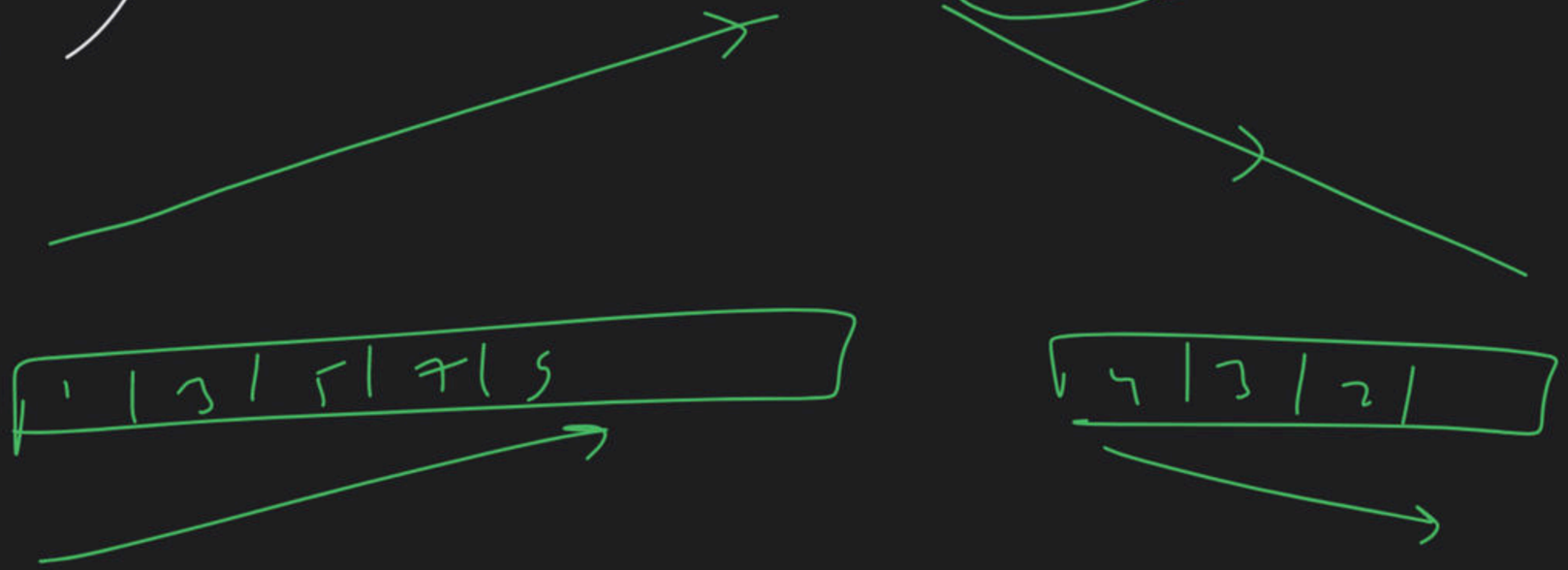
10 comparison

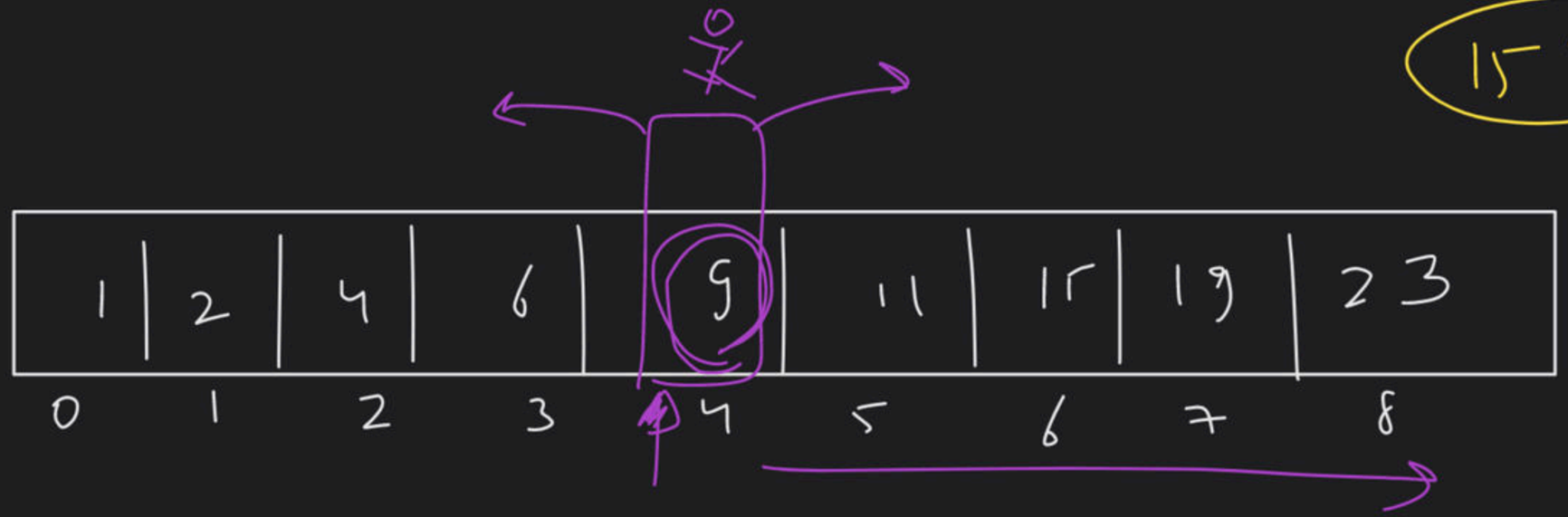
Binary Search:-

condition

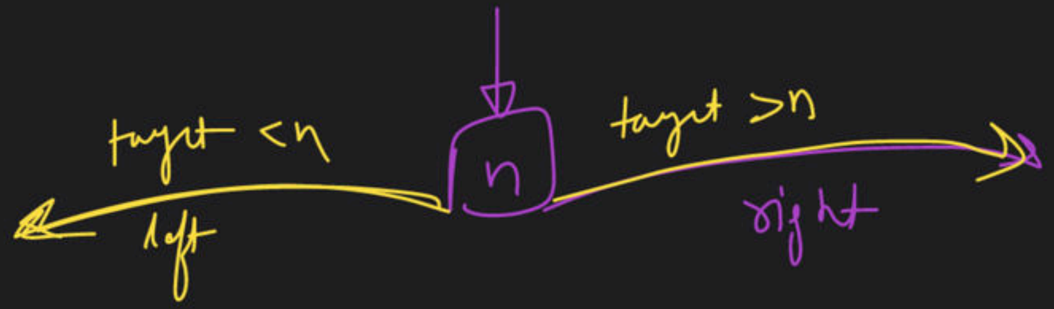
element should be in monotonic order





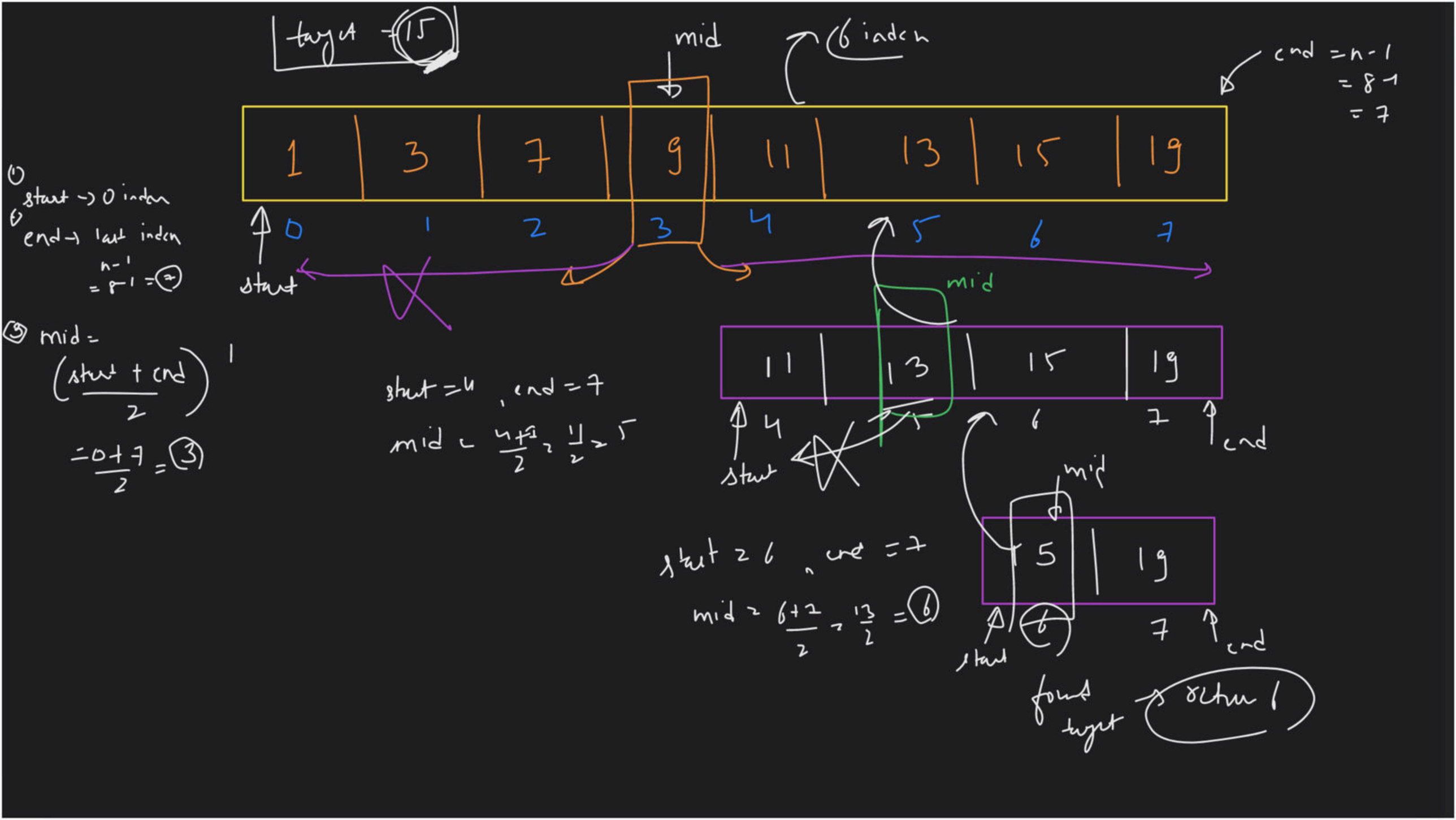


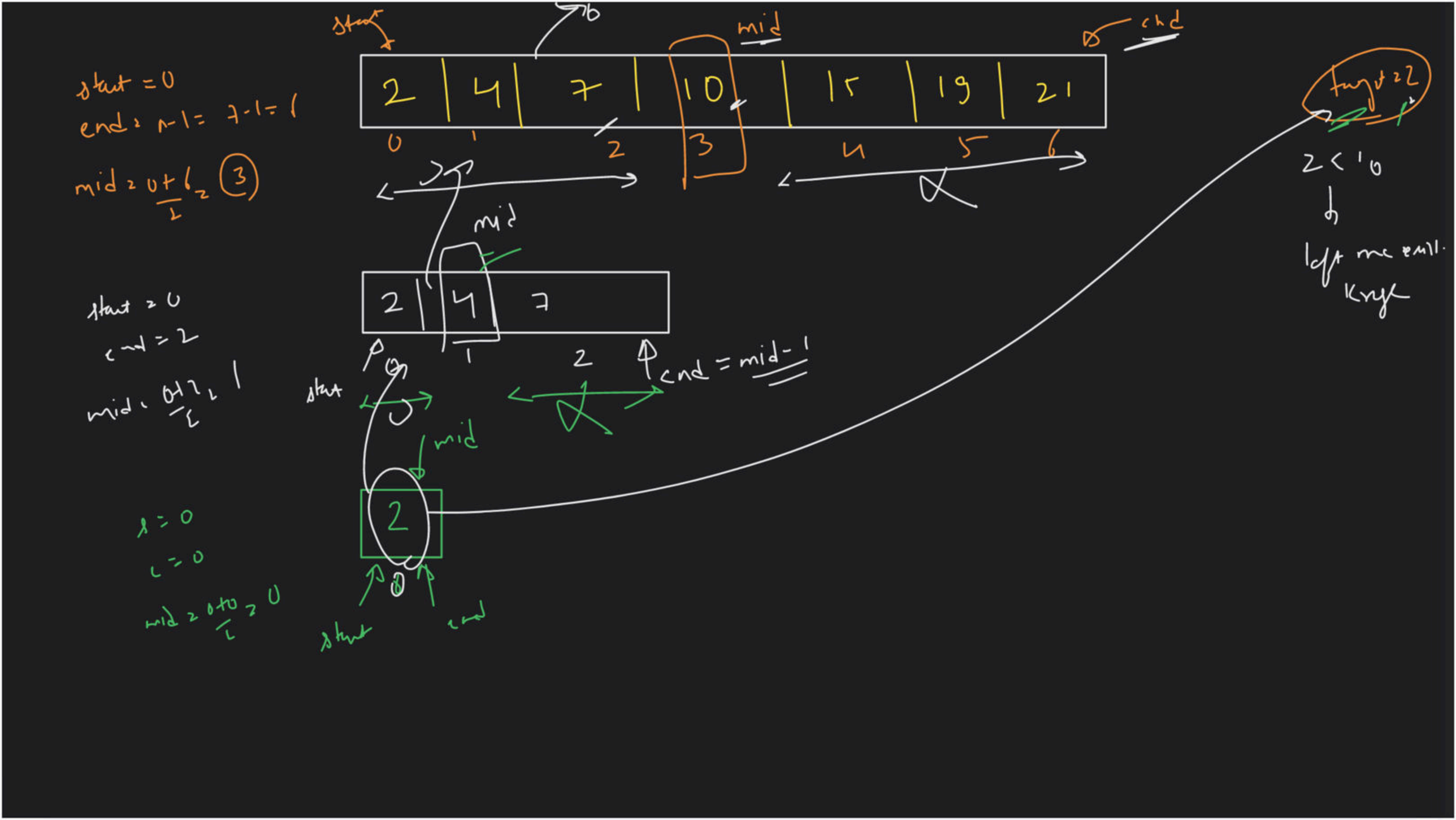
target -> 15

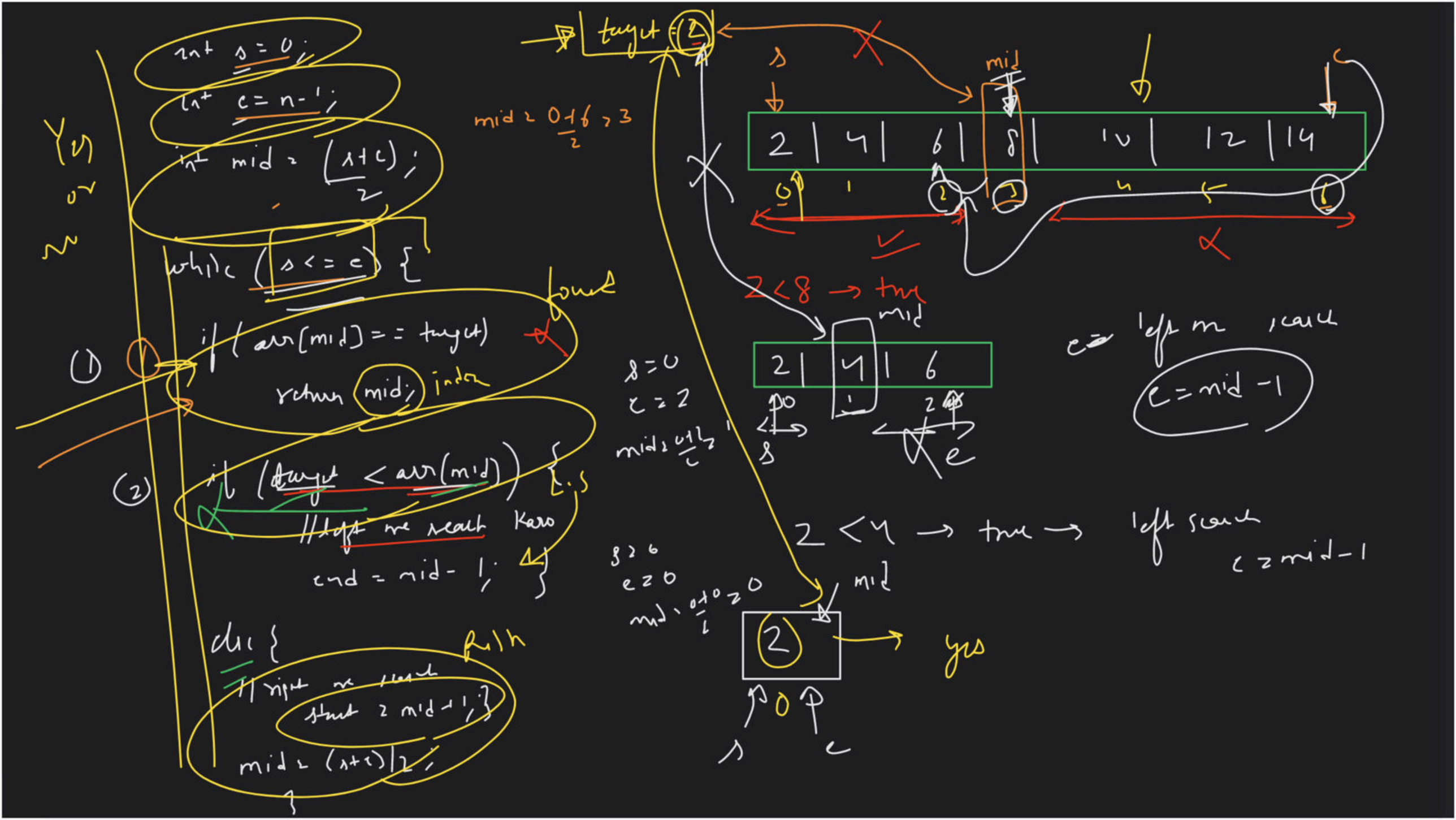


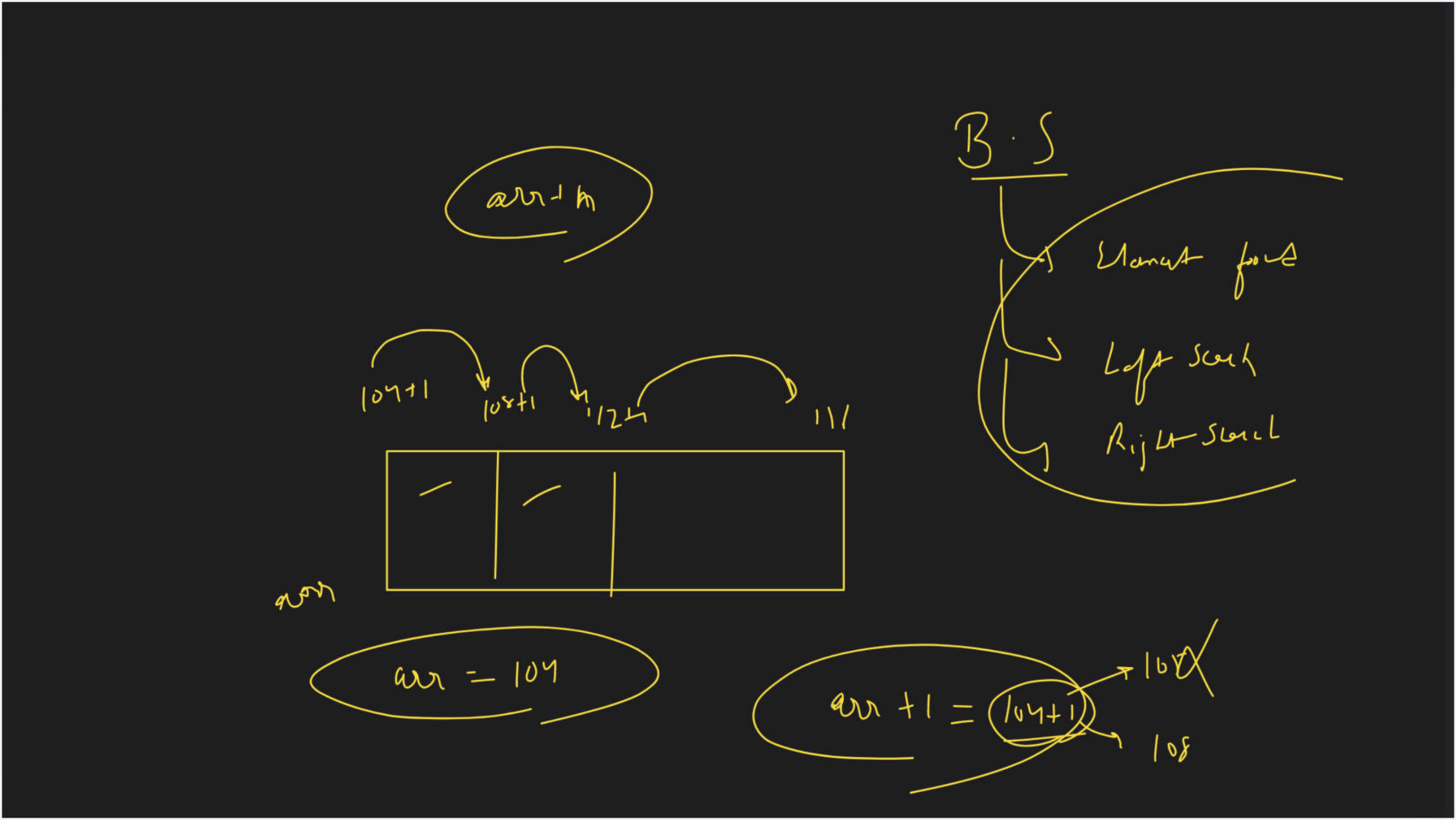
target > n

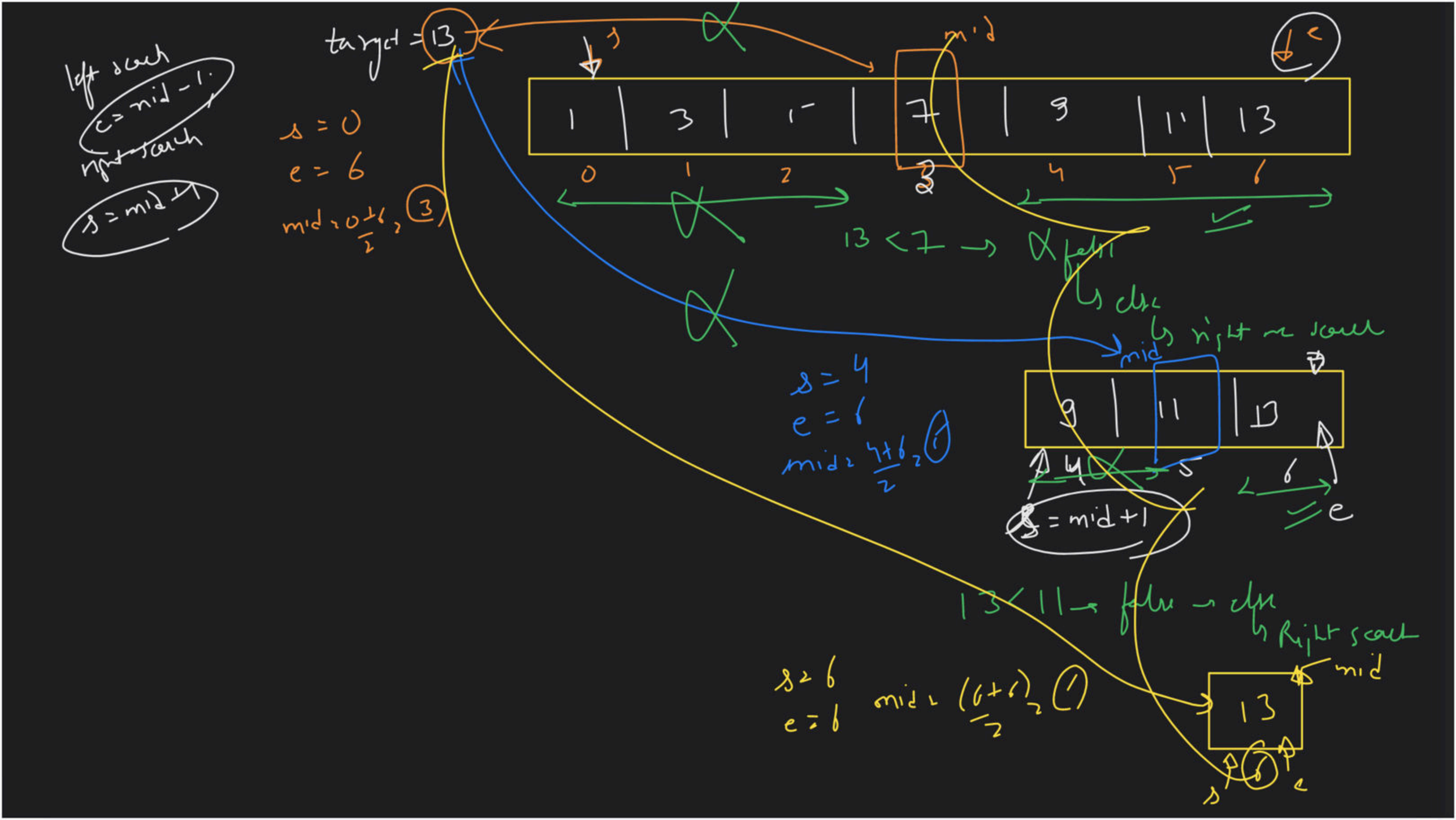
target < n

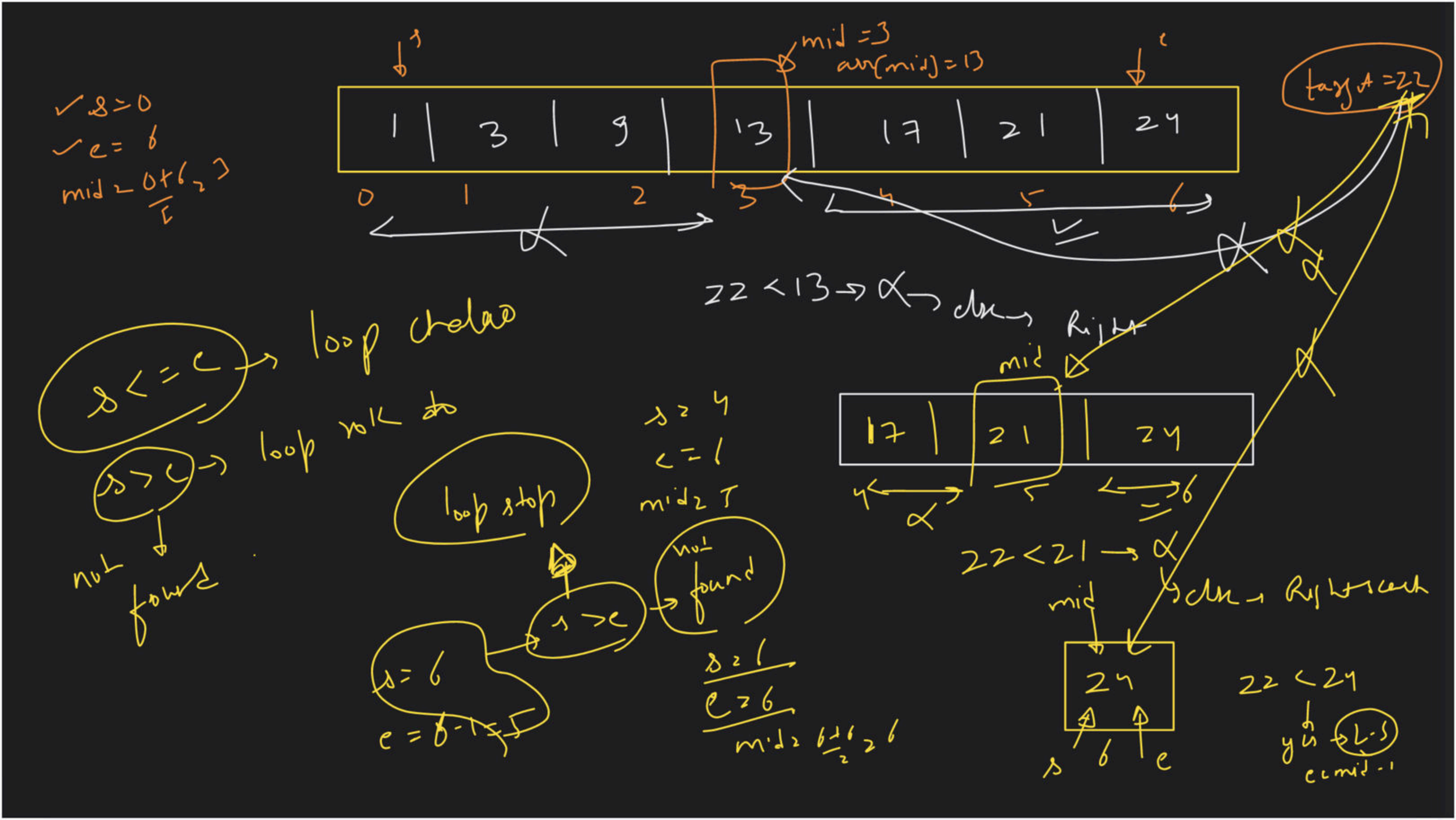

target = 15
mid
6 index
end = n-1
= 8-1
= 7
1 | 3 | 7 | 9 | 11 | 13 | 15 | 19
0 1 2 3 4 5 6 7
start
① start → 0 index
② end → last index
n-1
= 8-1 = ⑦
③ mid =
(start + end) / 2
= 0+7/2 = ③
start = 4 , end = 7
mid = 4+7/2 = 11/2 = 5
mid
11 | 13 | 15 | 19
4 5 6 7
start
end
start = 6 , end = 7
mid = 6+7/2 = 13/2 = ⑥
mid
5 | 19
6 7
start
end
found target → return 1

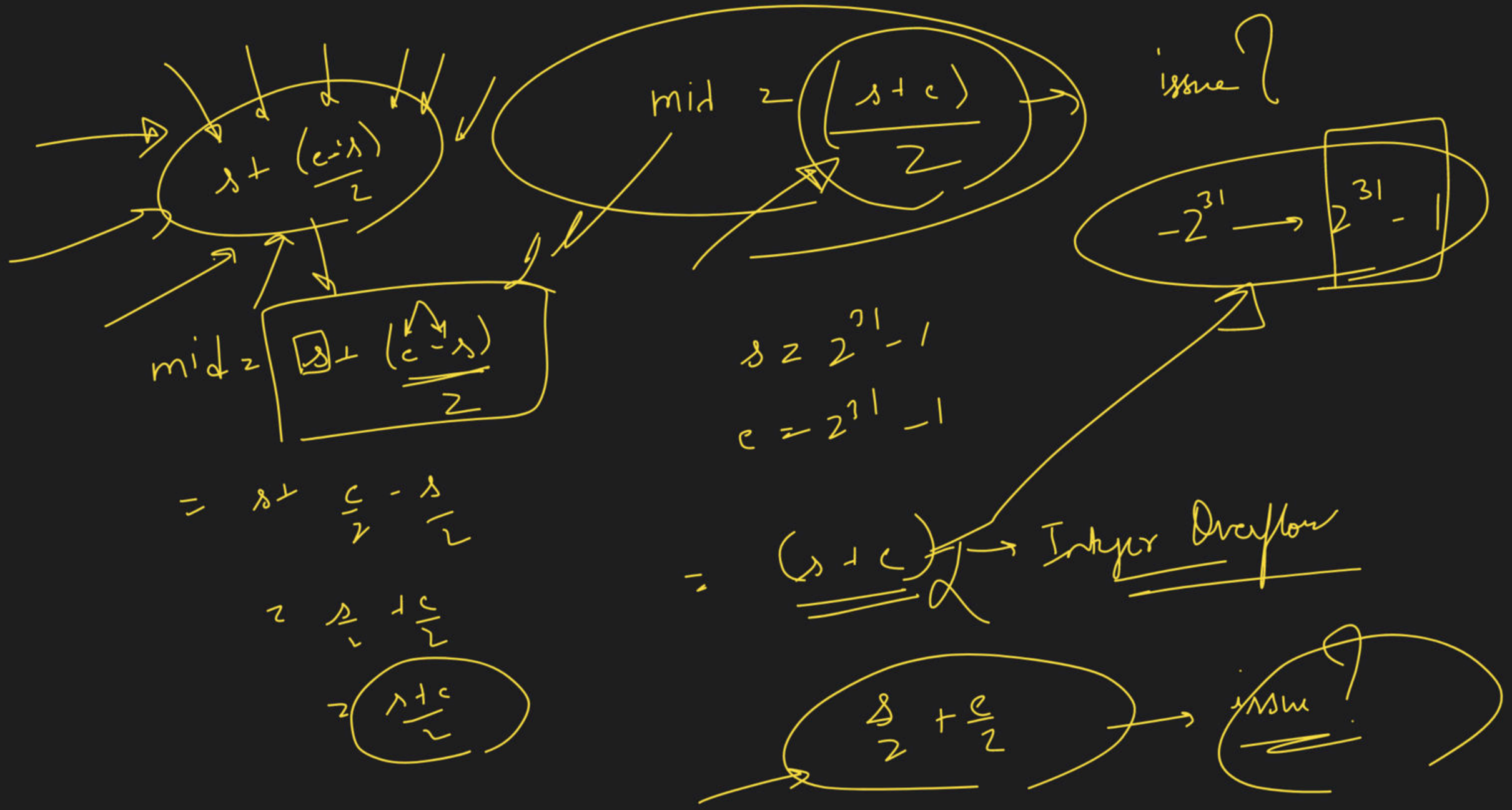

start
mid
end
2
4
7
10
15
19
21
0
1
2
3
4
5
6
start = 0
end = n-1 = 7-1 = 6
mid = 0+6/2 = 3
2 < 10
left me exist. krega
start = 0
end = 2
mid = 0+2/2 = 1
end = mid-1
start = 0
end = 0
mid = 0+0/2 = 0

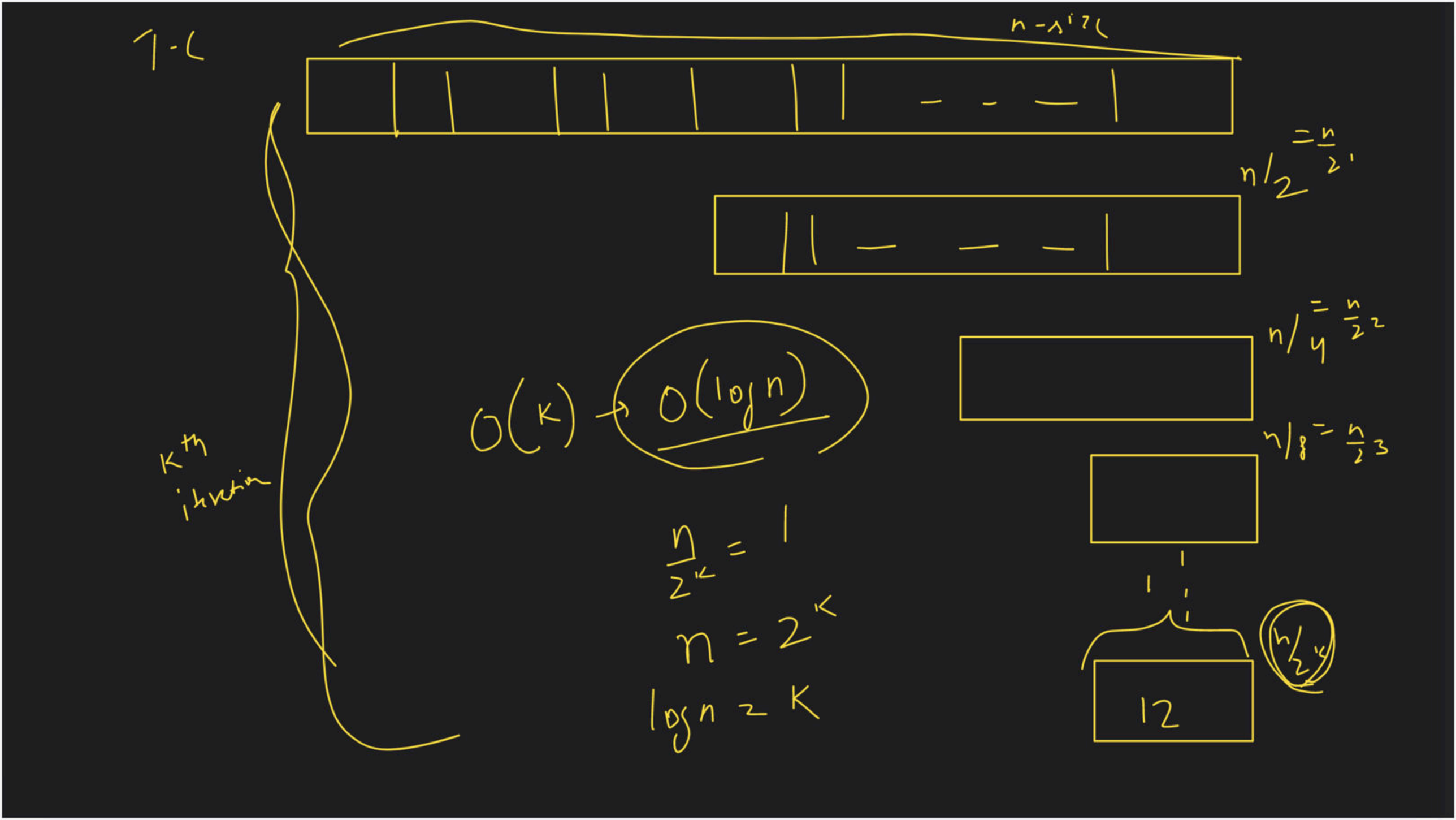

int s = 0;
int e = n-1;
int mid = (s+e)/2;
while (s <= e) {
if (arr[mid] == target)
return mid; index
found
if (target < arr(mid))
//left me search karo
end = mid - 1;
else {
//right me search
start = mid+1;}
mid = (s+e)/2;
}
Yes
No
target
mid = 0+6/2 = 3
2 | 4 | 6 | 8 | 10 | 12 | 14
s
mid
e
2<8 → true
mid
2 | 4 | 6
s
e
e = left me search
e = mid - 1
s = 0
e = 2
mid = 0+2/2 = 1
2 < 4 → true → left search
e = mid-1
s > e
s = 0
e = 0
mid = 0+0/2 = 0
mid
2
yes
s
e

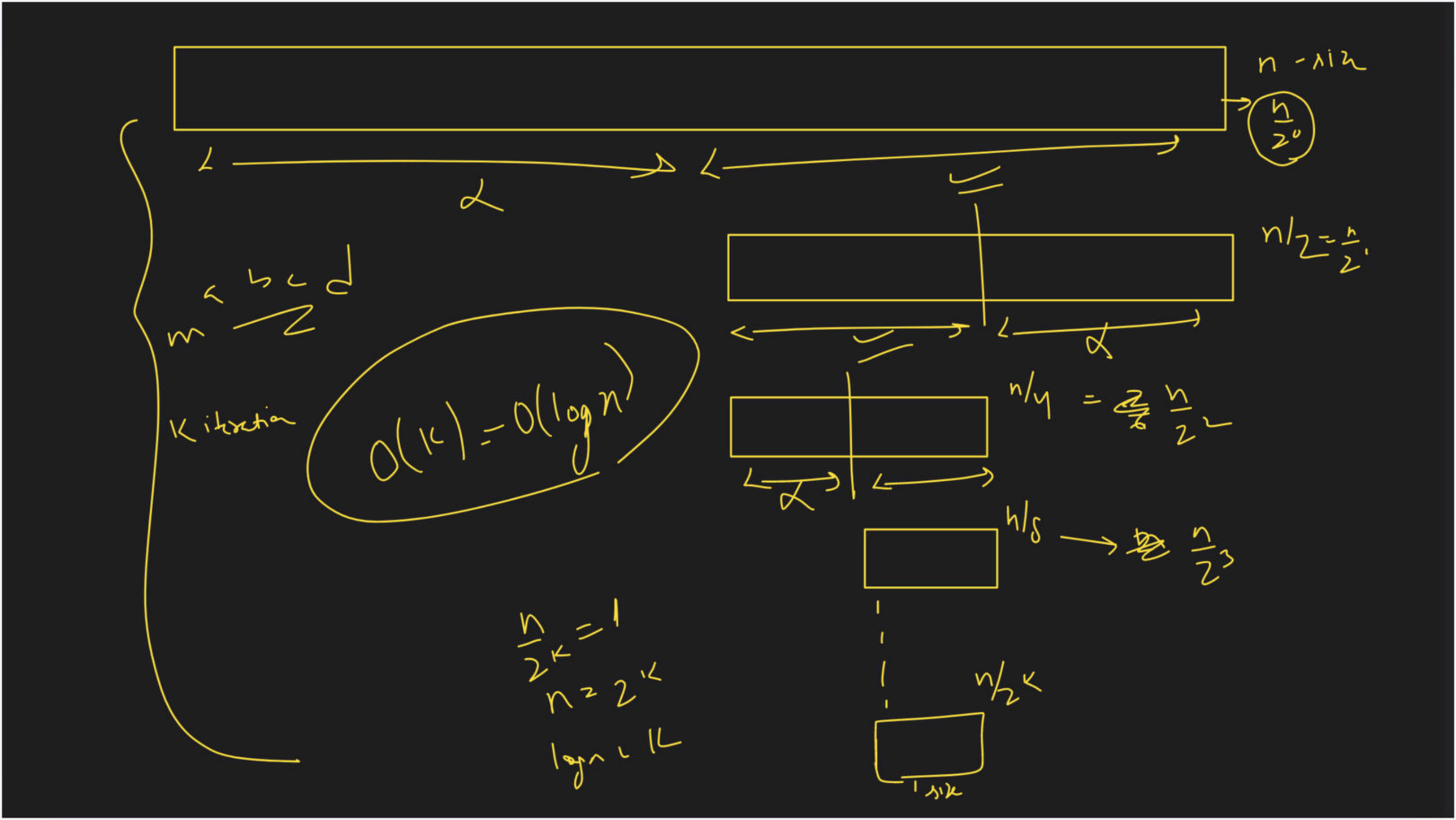

B.S
Element found
Left search
Right search
arr-1 th
104+1
108+1
112+1
111
arr
arr = 104
arr +1 = 104+1
108
108

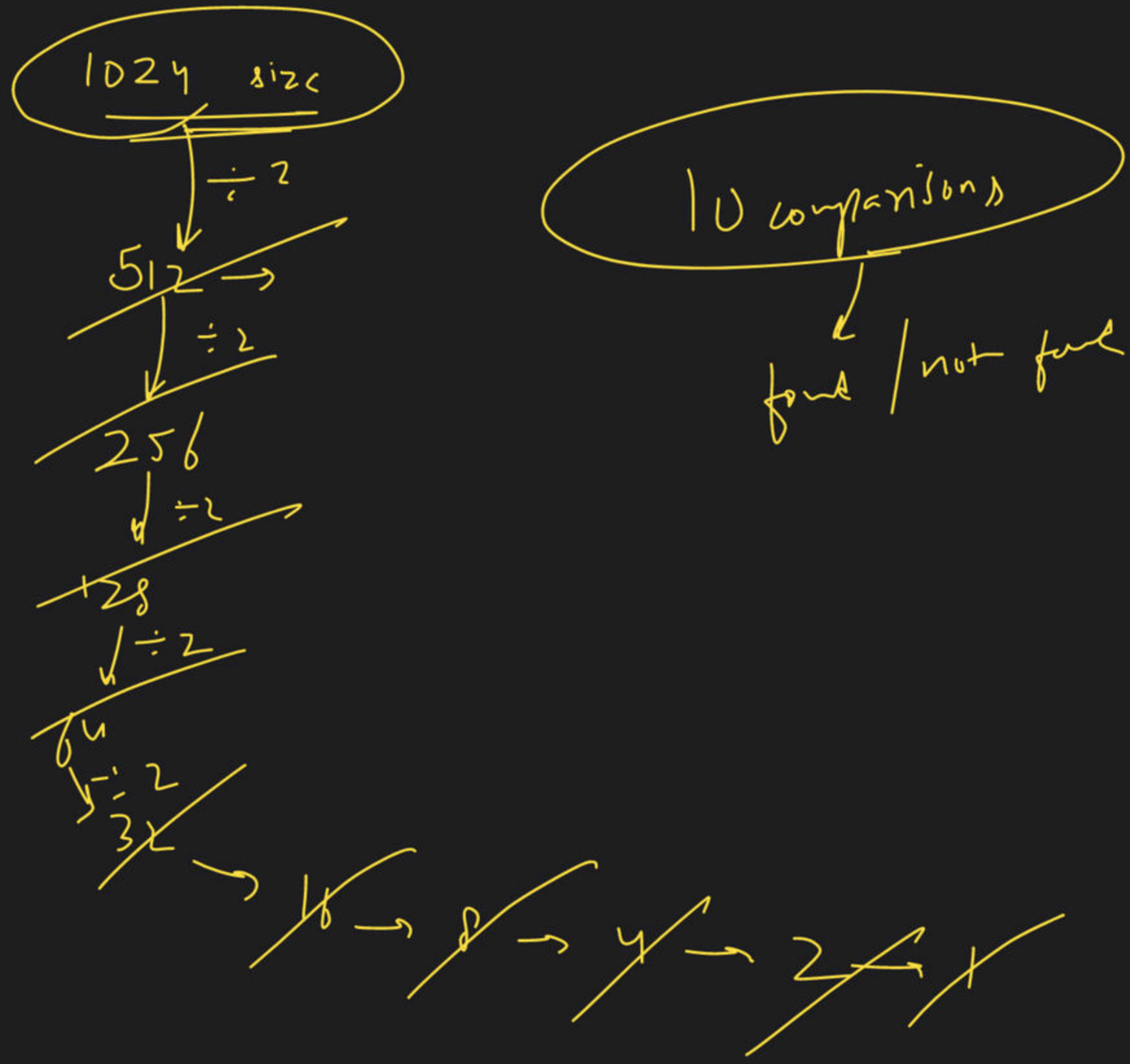

left search
e = mid - 1
right search
s = mid + 1
target = 13
s = 0
e = 6
mid = (0+6)/2 = 3
1 3 7 9 11 13
0 1 2 3 4 5 6
mid
13 < 7 → false
s = 4
e = 6
mid = (4+6)/2 = 5
9 11 13
4 5 6
s = mid + 1
13 < 11 → false
s = 6
e = 6
mid = (6+6)/2 = 6
13


s=0
e=6
mid = 0+6/2 = 3
mid = 3
arr[mid] = 13
target = 22
1 | 3 | 9 | 13 | 17 | 21 | 24
0 1 2 3 4 5 6
22 < 13 → ✗ → don't go Right
mid
17 | 21 | 24
4 5 6
22 < 21 → ✗
don't go Right at all
mid
24
s 6 e
22 < 24
yes → 4-5
e = mid - 1
s <= e → loop chalao
s > e → loop nahi chalao
null
found
s = 4
e = 1
mid = ?
loop stop
null found
s > e
s = 6
e = 6-1 = 5
s = 6
e = 6
mid = 6+6/2 = 6


mid = (s + c) / 2
issue ?
-2^{31} → $2^{31} - 1$
$s + \frac{(c-s)}{2}$
mid = $s + \frac{(c-s)}{2}$
$s = 2^{31} - 1$
$c = 2^{31} - 1$
$= s + \frac{c}{2} - \frac{s}{2}$
$= \frac{s}{2} + \frac{c}{2}$
$= \frac{s+c}{2}$
Integer Overflow
$\frac{s}{2} + \frac{c}{2}$
$\frac{s+c}{2}$


n-size
n/2 = n/2^1
n/4 = n/2^2
n/8 = n/2^3
n/2^k
12
kth iteration
O(k) → O(log n)
n/2^k = 1
n = 2^k
log n = k


n - size
n/2^0
n/2 = n/2^1
n/4 = n/2^2
n/8 → n/2^3
n/2^k
1 size
m < b < d
k iteration
O(k) = O(log n)
n/2^k = 1
n = 2^k
log n = k


1024 size
÷ 2
512
÷ 2
256
÷ 2
128
÷ 2
64
÷ 2
32
10 comparisons
found / not found

①

What is Binary Search

↳ $\log n$ → search

T.C → $O(\log n)$ → why → devin

n = size of array

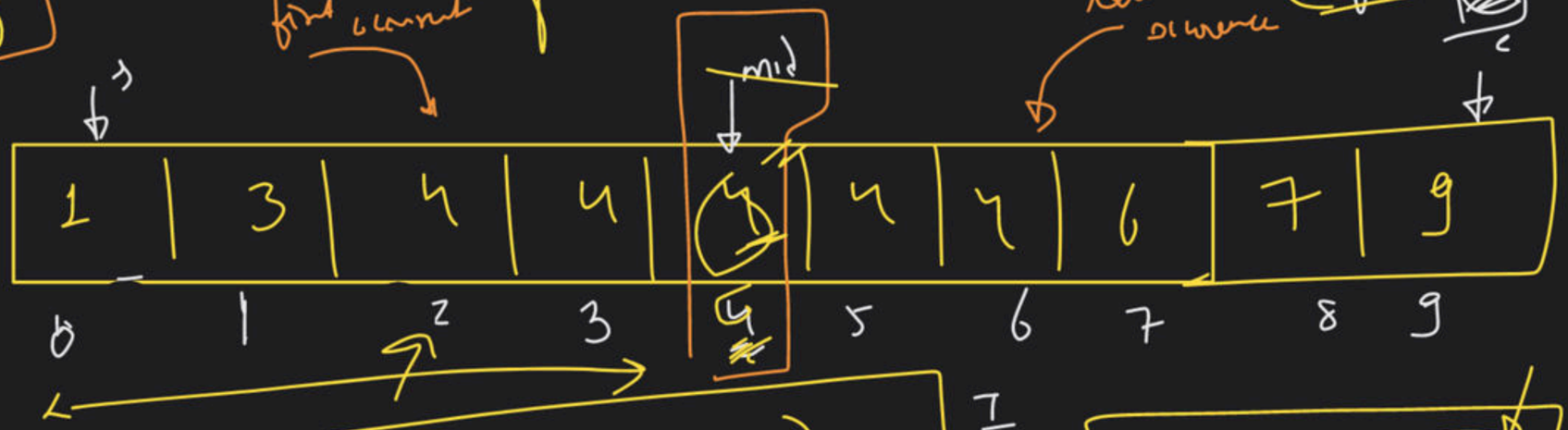
In-built

↳ yes → algorithm

binary_search(arr, arr+n, target)

binary_search(v.begin(), v.end(), target);

① find the first occurrence of an element

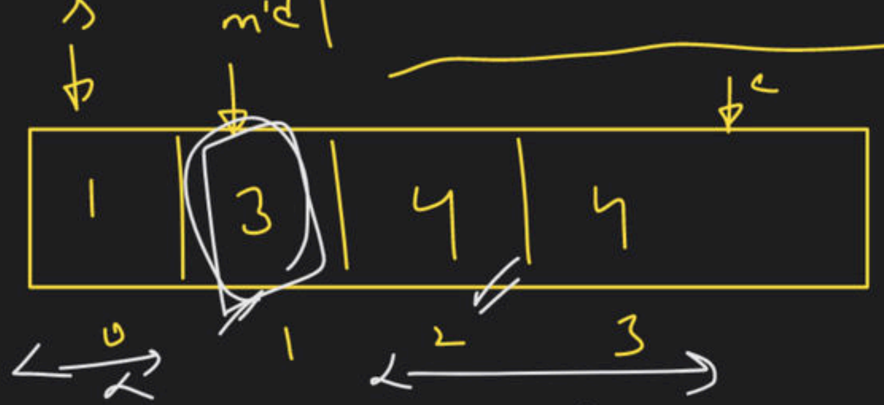


s = 0
e = 9
mid = (0+9)/2 = 4

I

if(arr[mid] == target)
4 == 4 → yes/true →
arr[ans] = index = 4
search in left
e = mid - 1

II

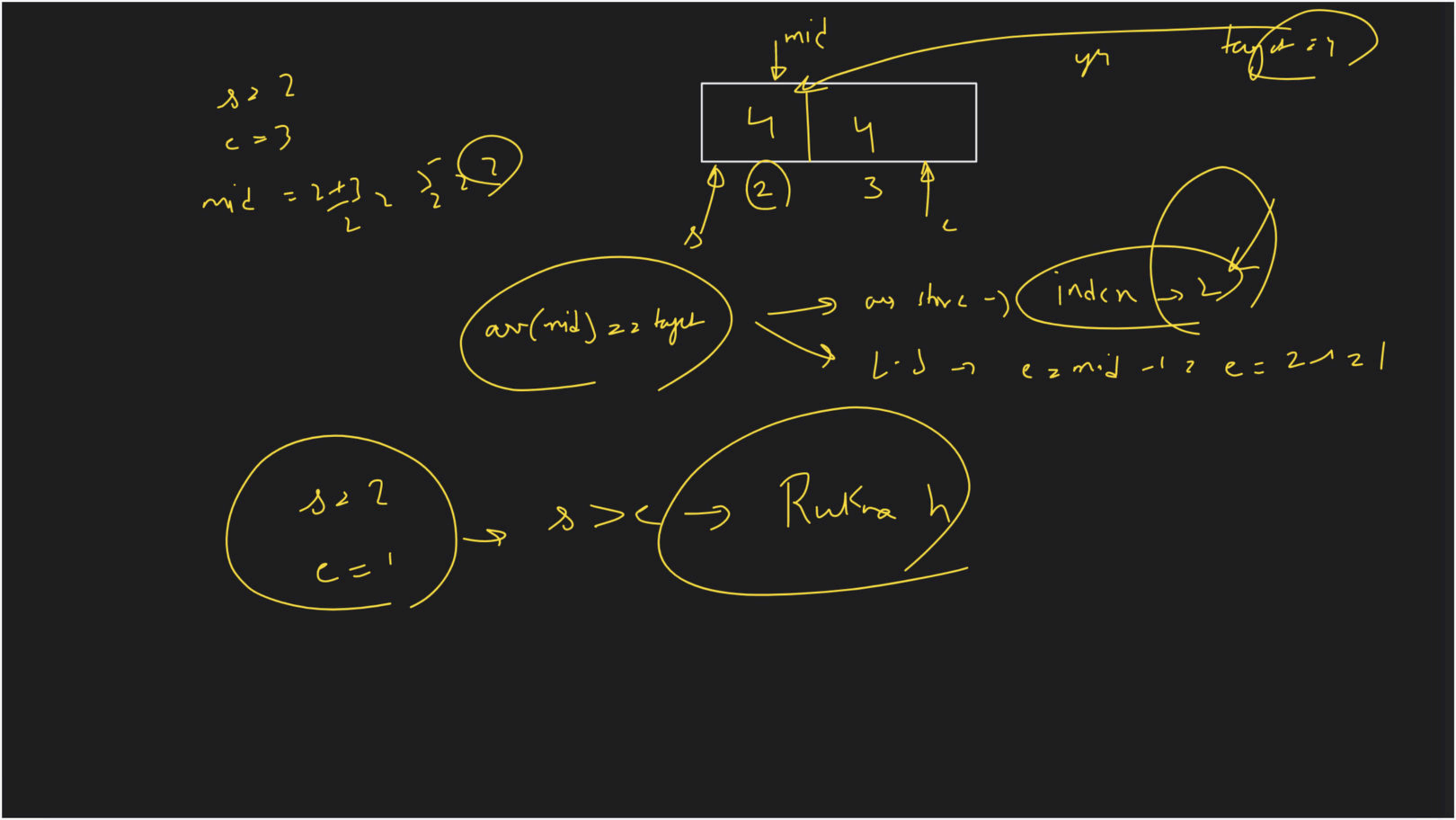


s = 0
e = 3
mid = (0+3)/2 = 1

4 > 3 →

target > arr(mid) → Right search
s = mid+1

target < arr(mid)
↳ left → e = mid - 1


s = 2
e = 3
mid = 2+3/2 = 5/2 = 2
mid
4
4
2
3
s
e
arr(mid) == target
arr.sort()
index -> 2
L.S -> e = mid - 1 = e = 2-1=1
s = 2
e = 1
s > e
Ruk na h

① if (arr[mid] == target)

ans = mid

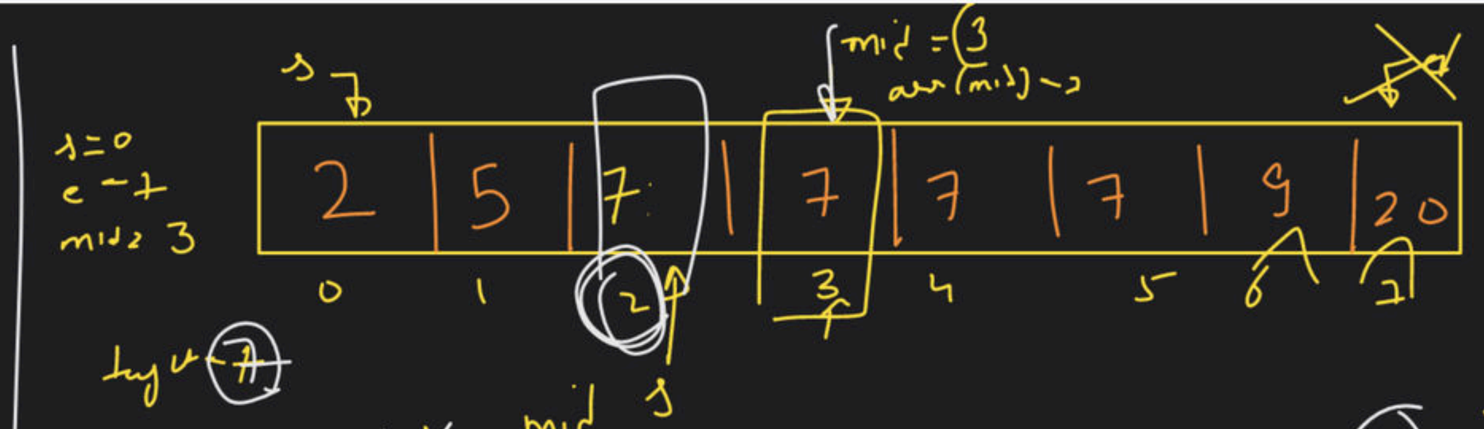
left → e = mid - 1

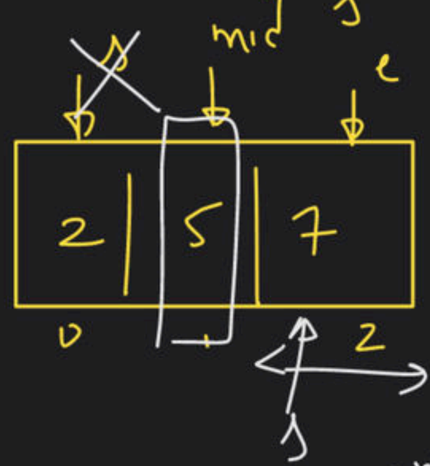
② if (target < arr[mid])

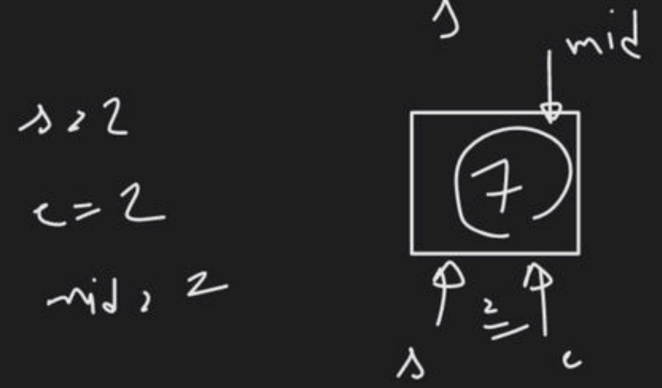
L.S → e = mid - 1

③ if (target > arr[mid])

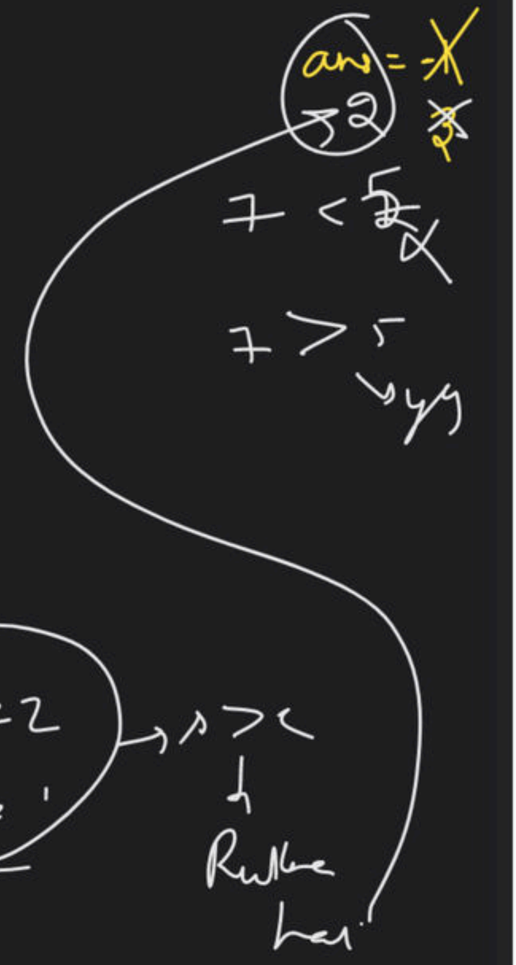
R.S → s = mid + 1



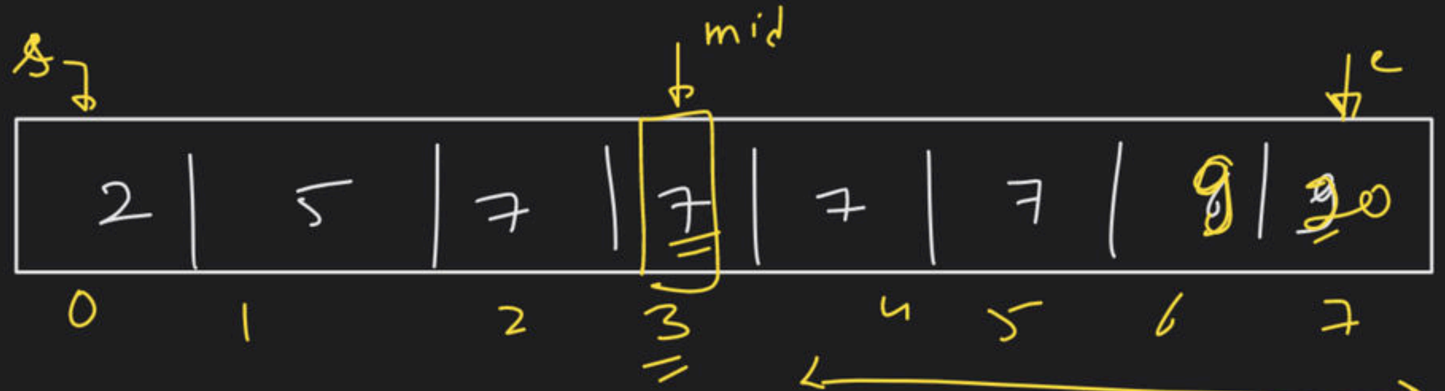




e = 2-1 = 1



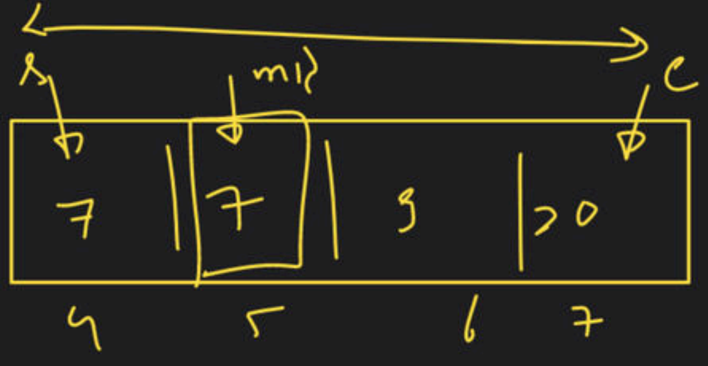
s = 0
e = 7
mid = 3



target = 7

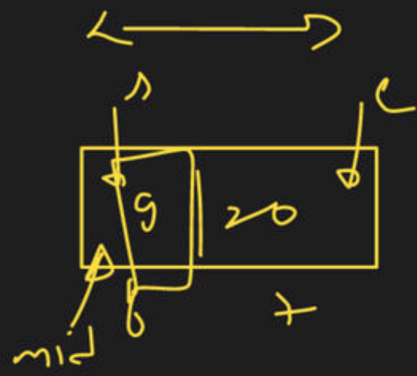
```
if (arr(mid) == target)
    ans = mid
    R.J → s = mid+1

if (target < arr(mid))
    L.J → e = mid-1

if (target > arr(mid))
    R.J → s = mid+1
```

s = 4
e = 7
mid = 5

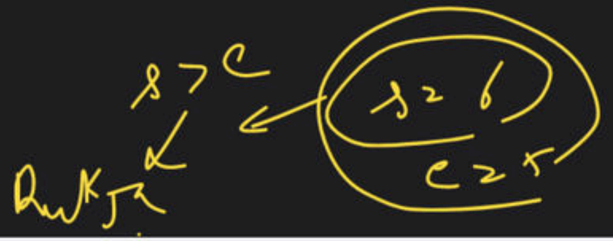


ans = 5

s = 6
e = 7
mid = 6



7 < 9

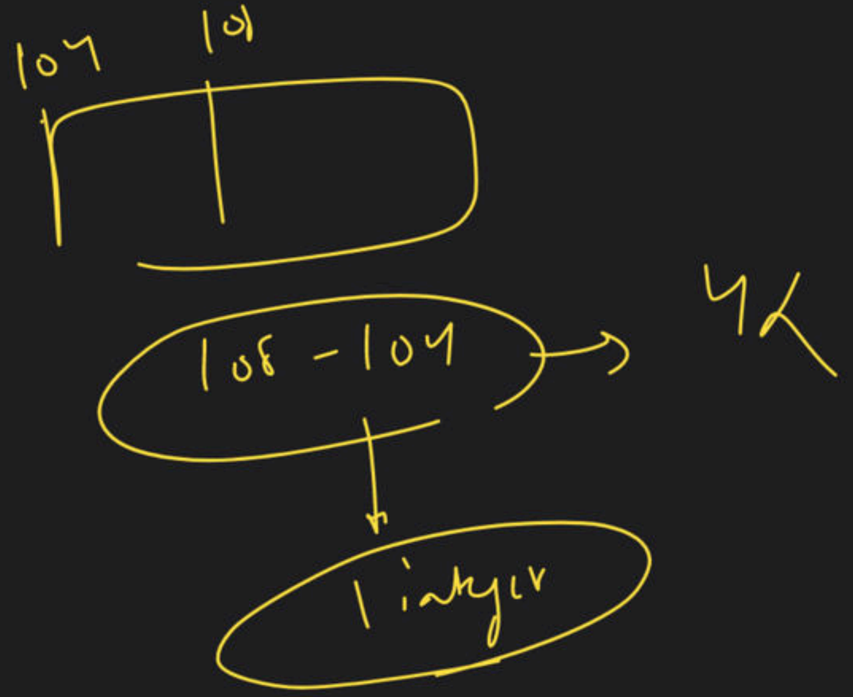


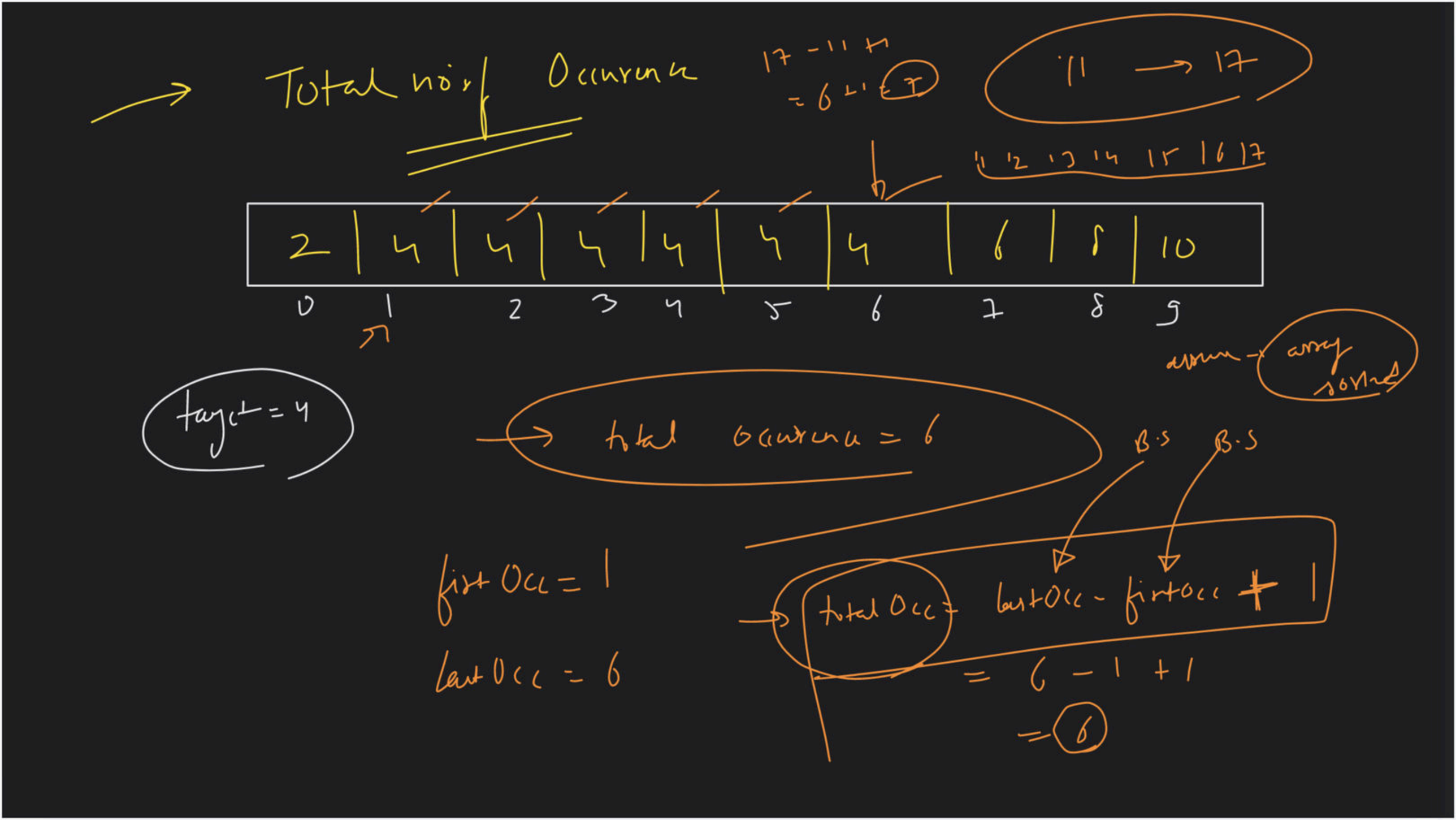
e = 6-1 = 5

Binary Search

first Occ

Last Occ




Total no. of Occurrence
17 - 11 + 1
= 6 + 1
= 7
11 → 17
11 12 13 14 15 16 17
2 4 4 4 4 4 4 6 8 10
0 1 2 3 4 5 6 7 8 9
target = 4
array sorted
total Occurrence = 6
B.S
B.S
first Occ = 1
last Occ = 6
total Occ = last Occ - first Occ + 1
= 6 - 1 + 1
= 6

B-1

LO

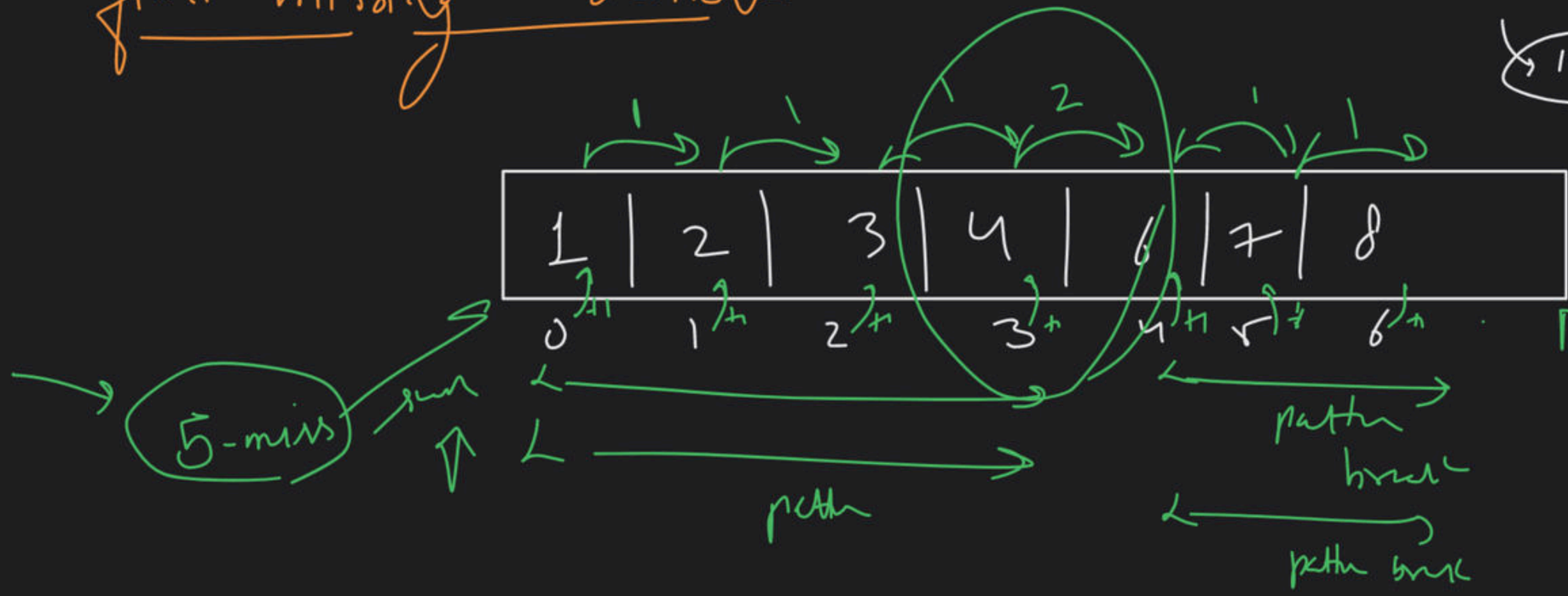
F-O

F-O

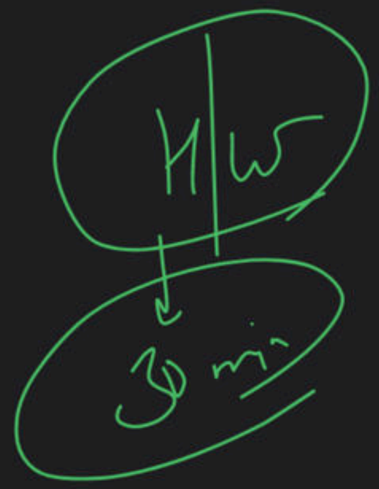
find missing element

1 → n

no miss

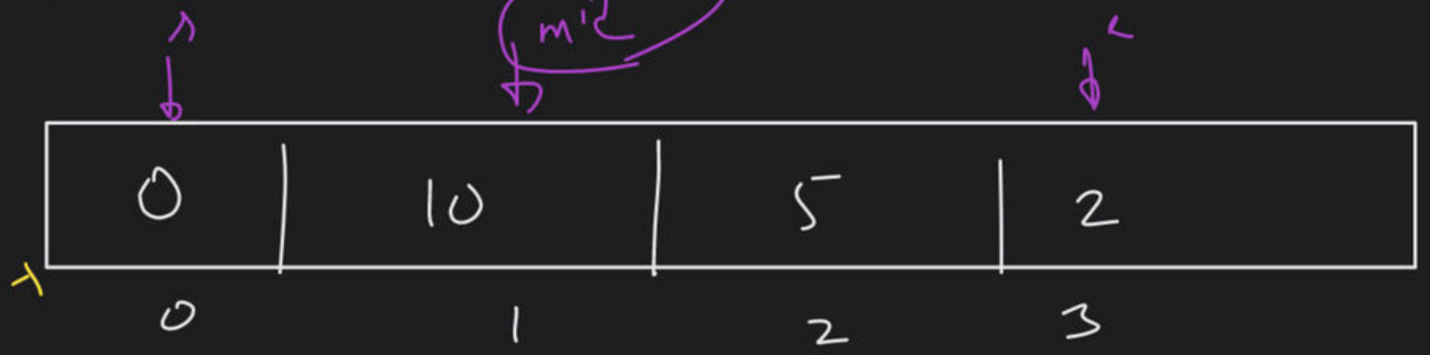


$1+2+3+4+5+6+7+8 \Rightarrow$

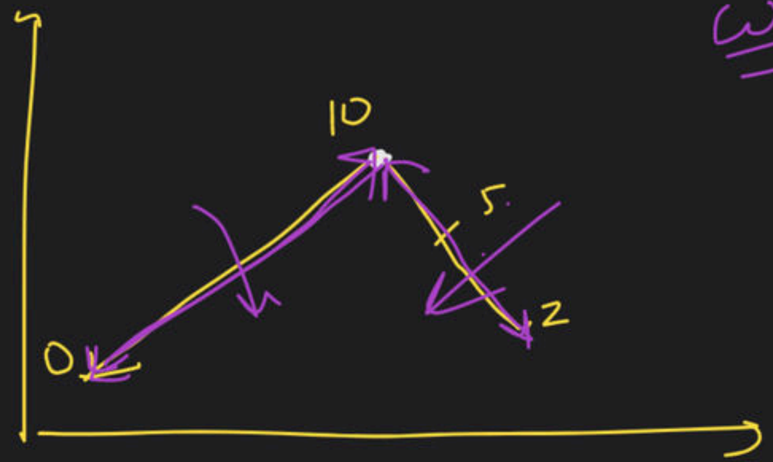


Peak element in a Mountain Array

arr = [1, 2, 7, 7, 3, 2]



o/p → 10 → Mountain ∩



Words

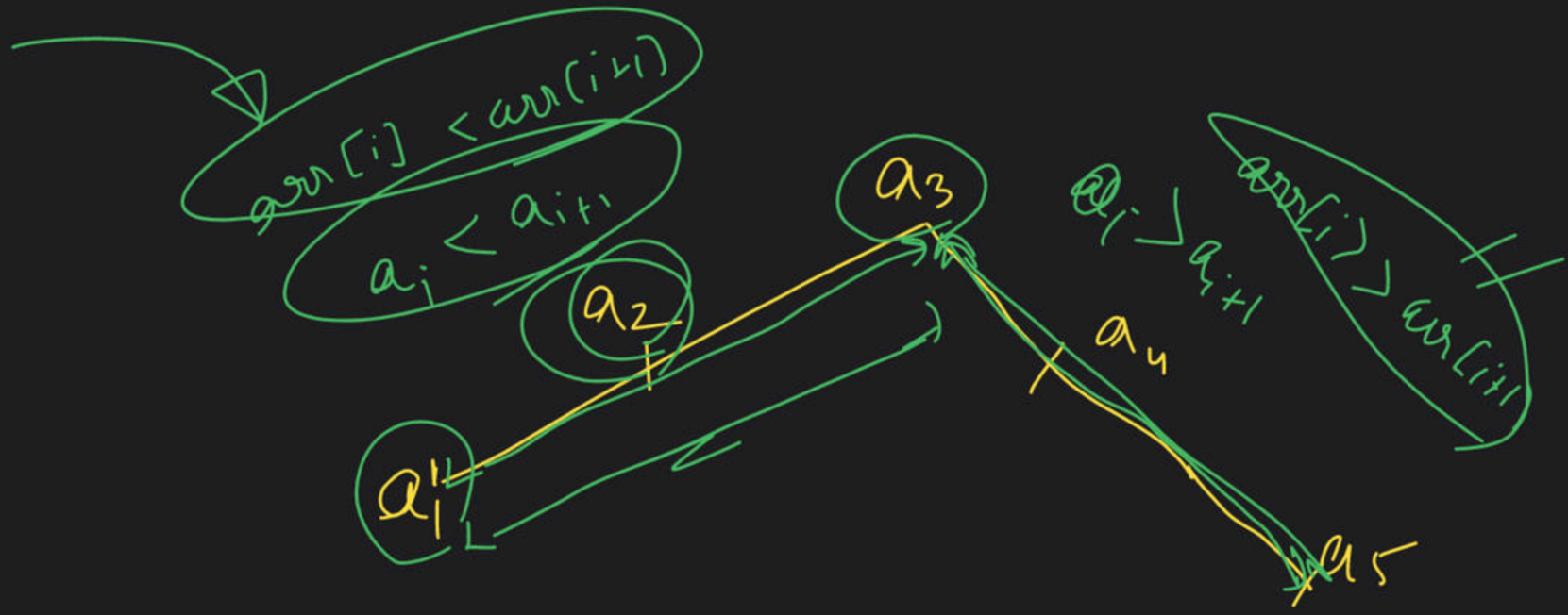
2 minute

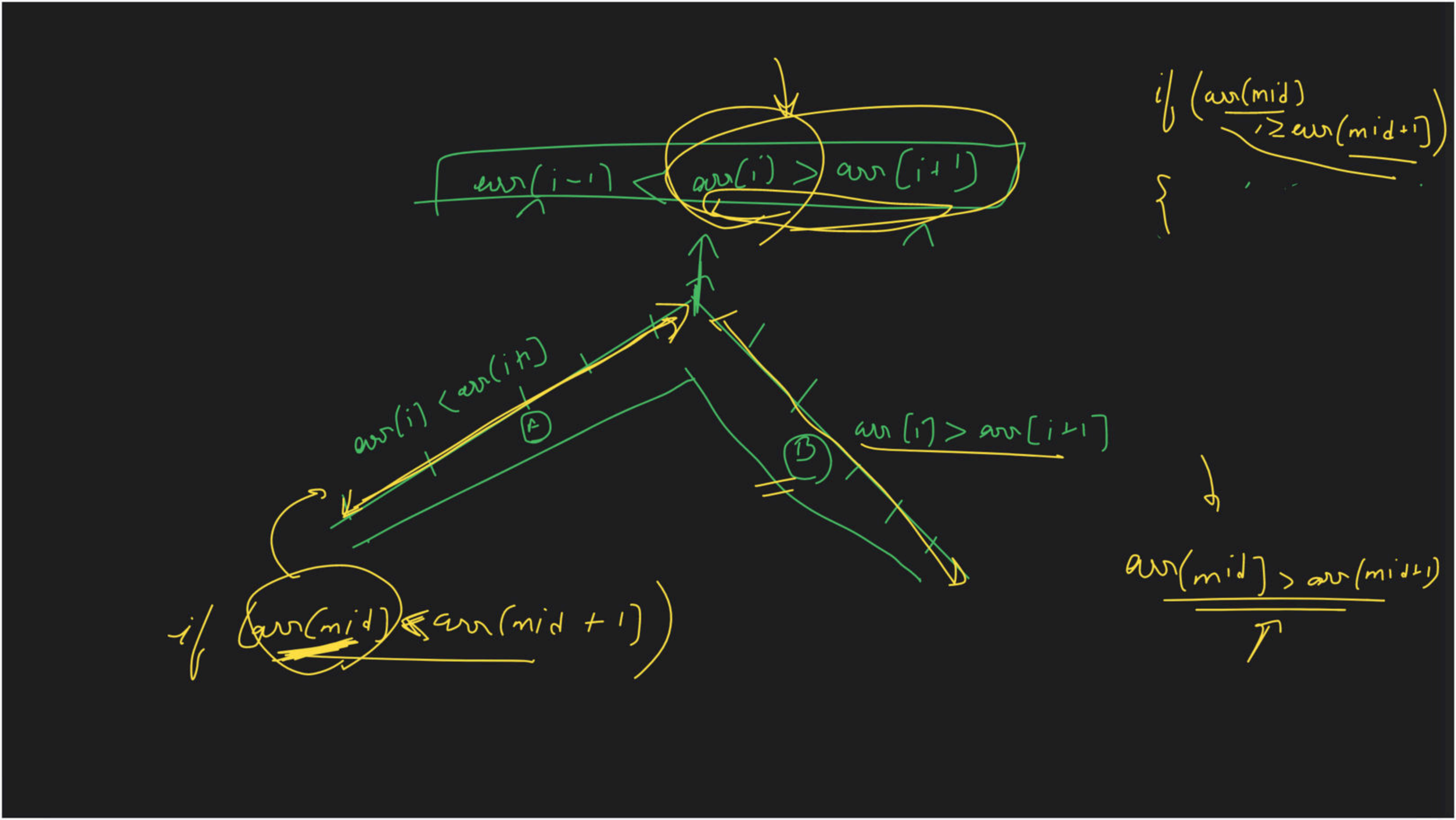
Brute force → L-S → max element

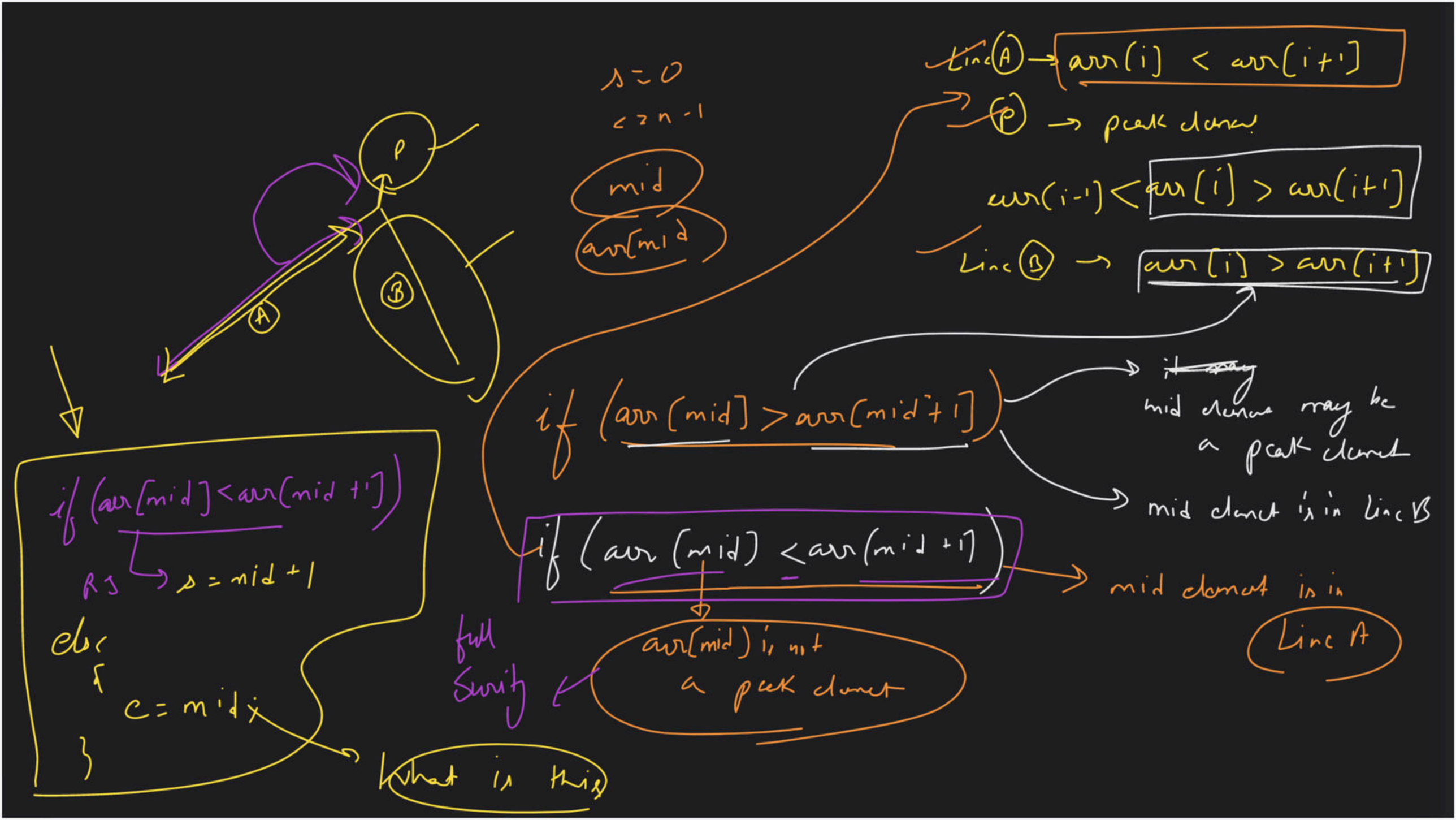
O(n)

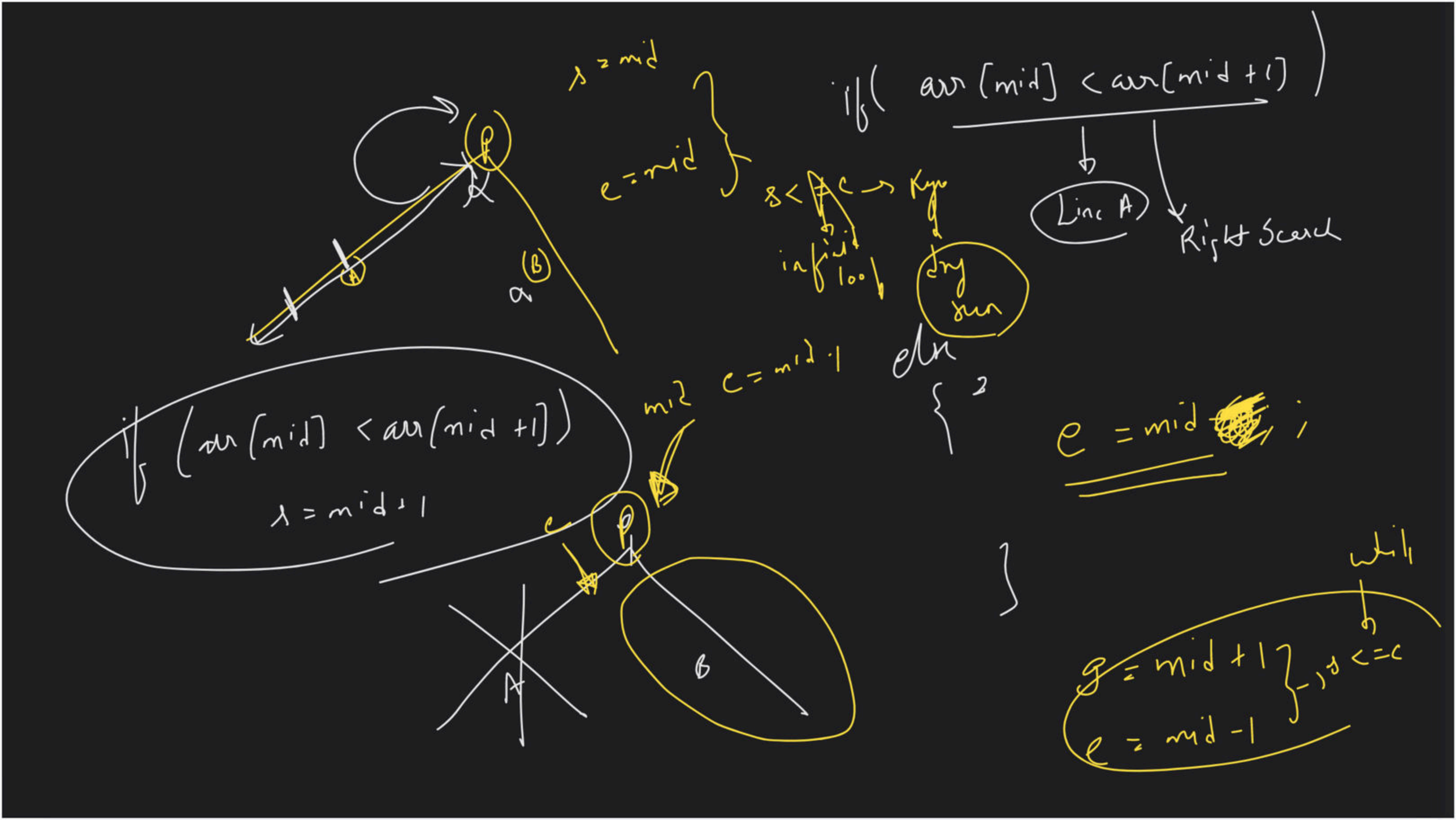
B.S → O(log n)

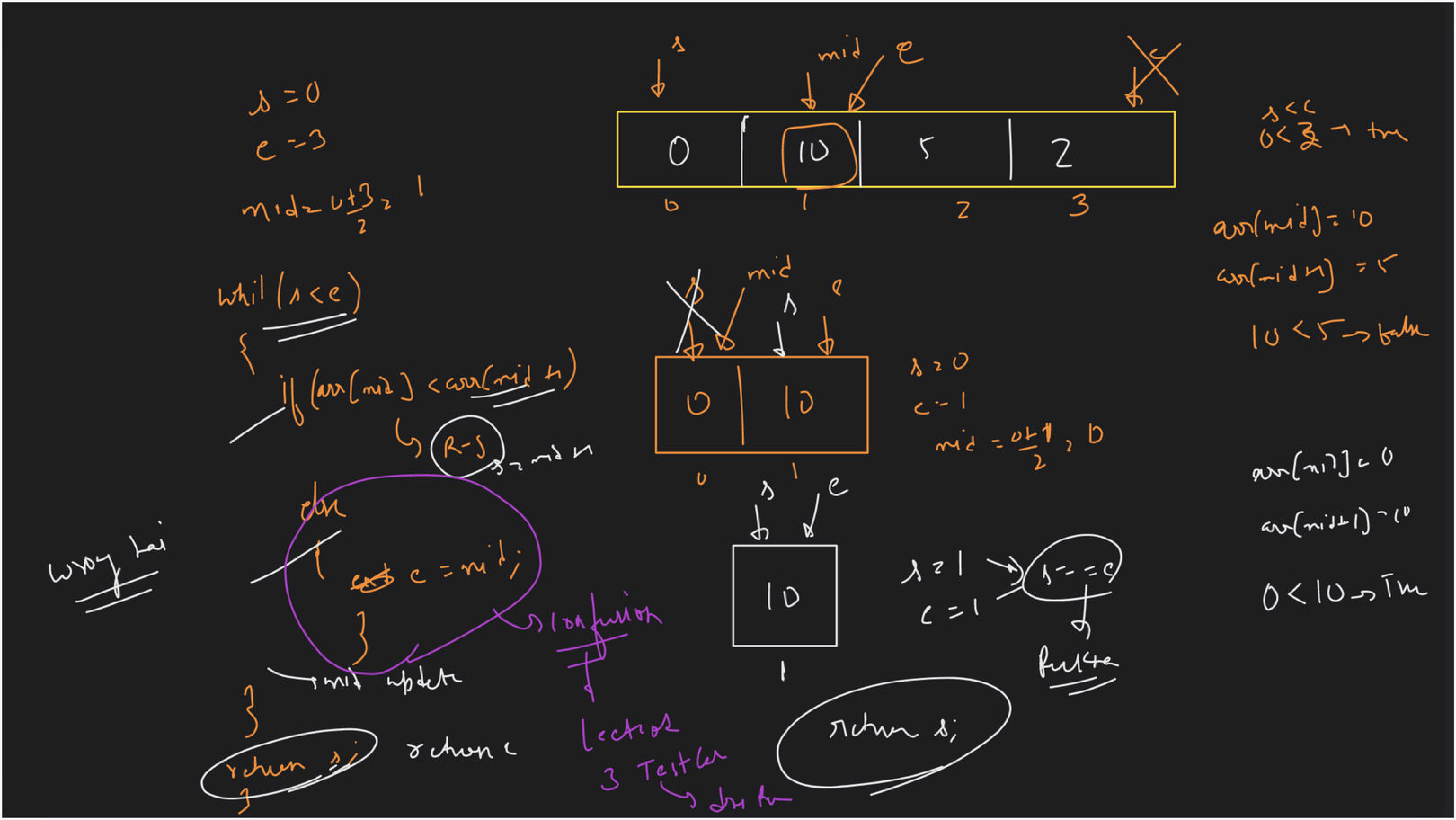
a_1	a_2	a_3	a_4	a_5



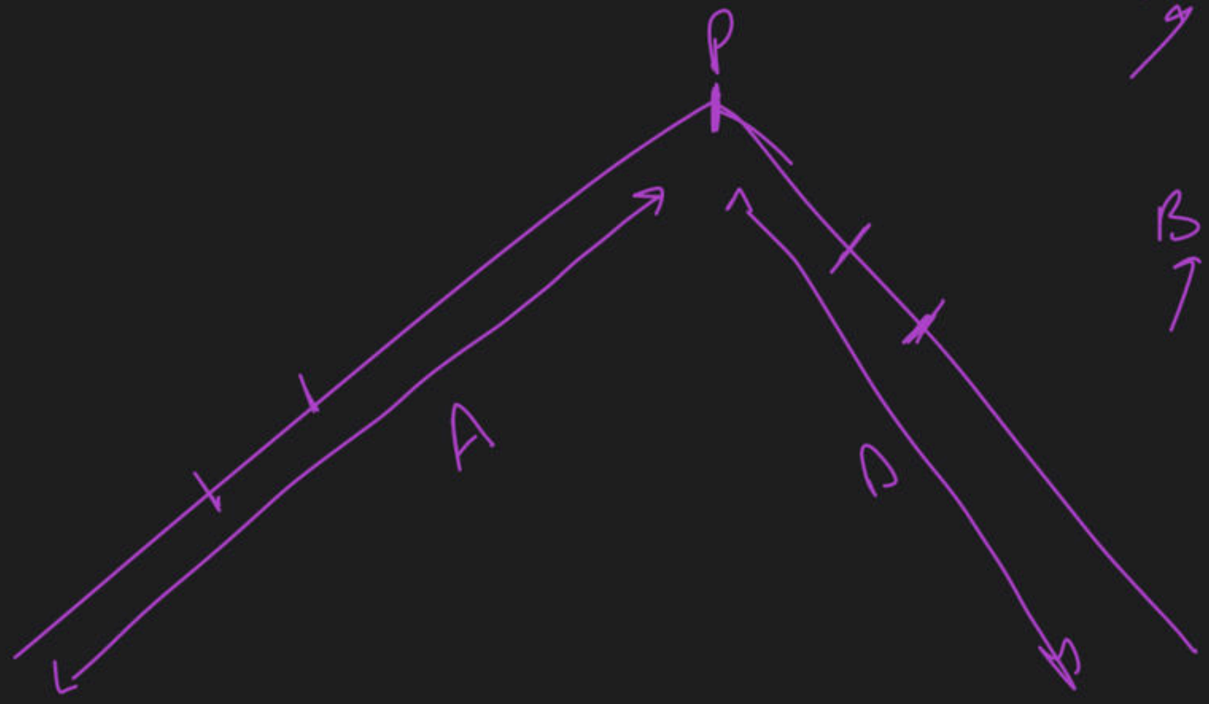

arr(i-1)
arr(i) > arr[i+1]
arr(i) < arr(i+1)
A
B
arr[i] > arr[i+1]
if (arr(mid) < arr(mid+1))
if (arr(mid) >= arr(mid+1))
{
arr(mid] > arr(mid+1)
T


s = 0
e = n-1
mid
arr[mid]
P
A
B
Line A → arr[i] < arr[i+1]
P → peak element
arr[i-1] < arr[i] > arr[i+1]
Line B → arr[i] > arr[i+1]
if (arr[mid] > arr[mid+1])
mid element may be a peak element
mid element is in Line B
if (arr[mid] < arr[mid+1])
mid element is in Line A
arr[mid] is not a peak element
full Sunity
if (arr[mid] < arr[mid+1])
RJ → s = mid+1
else
{
e = mid;
}
what is this


s = mid
e = mid
if(arr[mid] < arr[mid+1])
Linc A
Right Search
s < e → Kyo
infinite loop
dry run
else
{
mid
e = mid - 1
if (arr[mid] < arr[mid+1])
s = mid + 1
e = mid ;
}
while
s = mid + 1
e = mid - 1
s <= e
A
B
A
B



s = 0
e = 3
mid = (0+3)/2 = 1
while (s < e)
{
if (arr[mid] < arr[mid+1])
R-S
s = mid+1
else
{
e = mid;
}
mid update
}
return s;
return e
wrong hai
confusion
lecture
3 Testcase
dry run
s
mid
e
0
10
5
2
0
1
2
3
s<e
0<3 → true
arr[mid] = 10
arr[mid+1] = 5
10 < 5 → false
mid
s
e
0
10
s = 0
e = 1
mid = (0+1)/2 = 0
arr[mid] = 0
arr[mid+1] = 10
0 < 10 → true
s
e
10
1
s = 1
e = 1
s == e
false
return s;



A → arr(i) < arr[i+1]

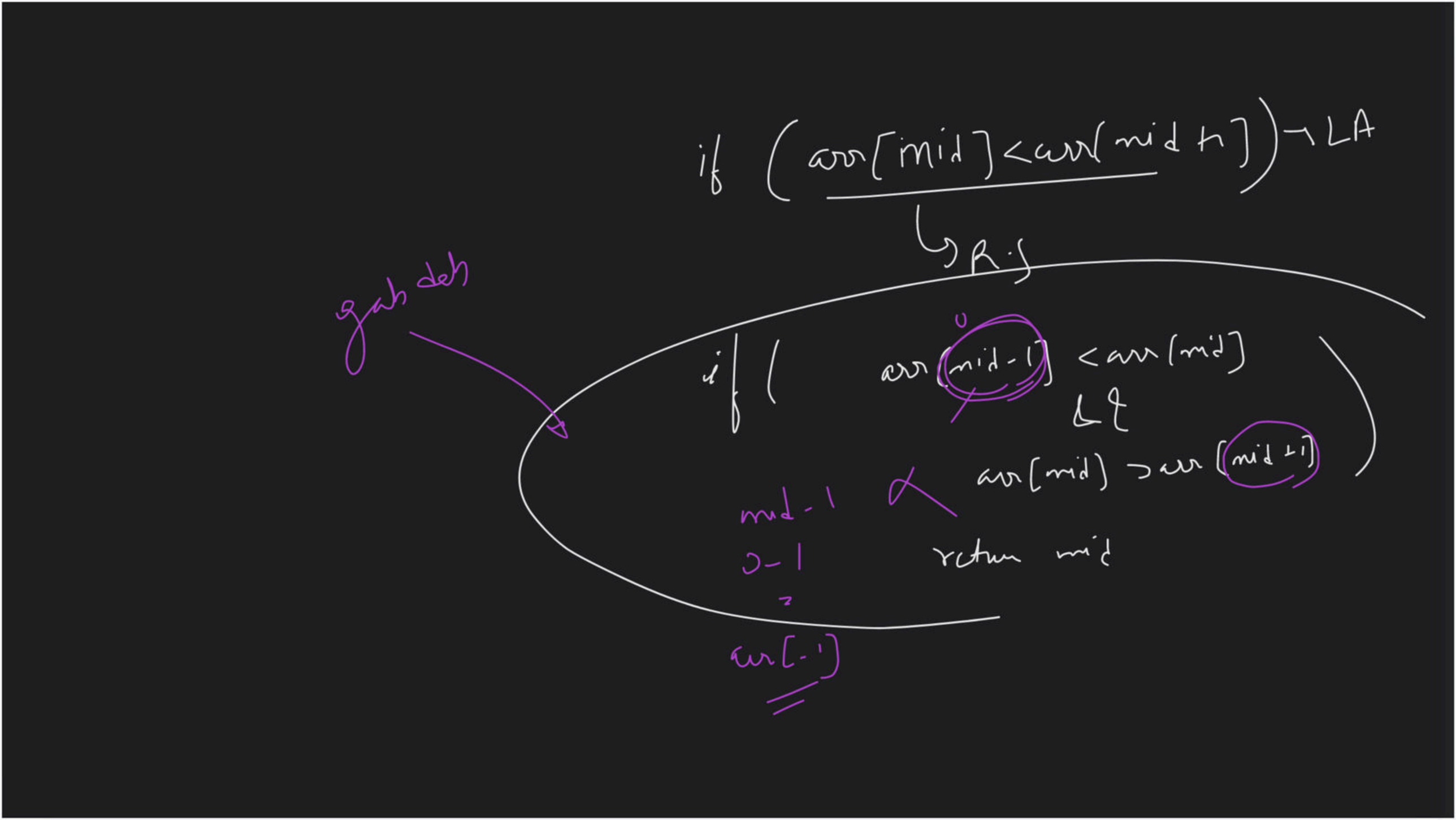
P → arr[i] > arr(i-1)
> arr[i+1]

B → arr(i) > arr[i+]

① find pivot using Binary Search

② Search in a rotated & sorted Array

③ SquareRoot using Binary Search


if (arr[mid] < arr[mid+1]) → LA
↳ R:j
grab deh
if (arr[mid-1] < arr[mid]
Lt
arr[mid] > arr[mid+1]
mid-1
∝
return mid
arr[-1]