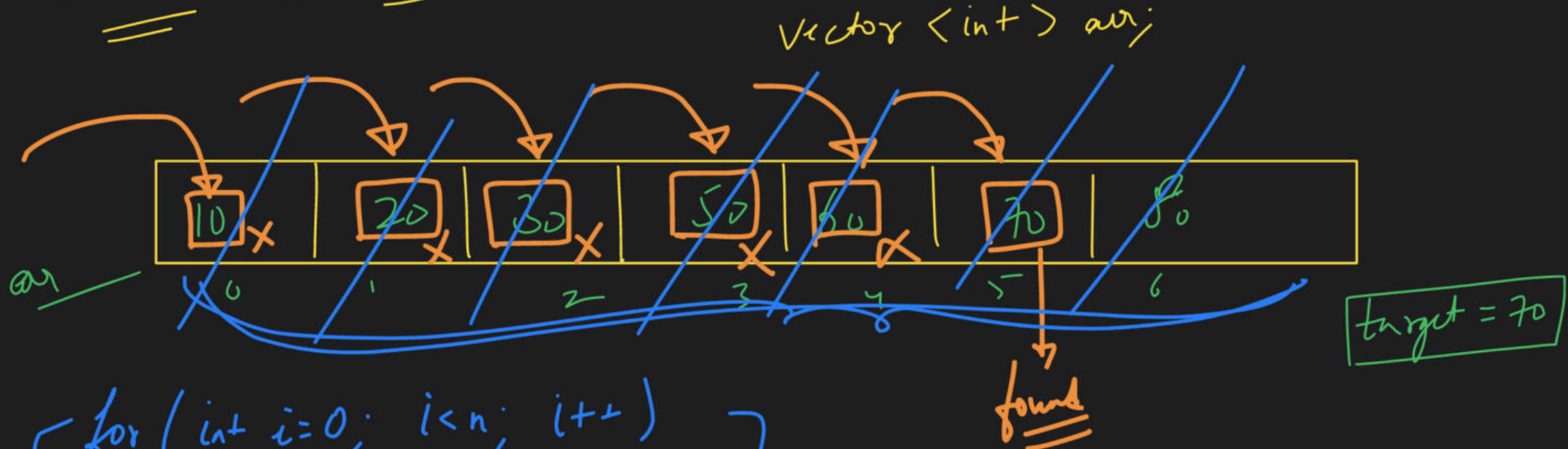
Searching && Sorting - Level 1

Special class

Love Babbar • Sept 13, 2023

Linear Search → 1-1 karke sabko check krdungi




```
for (int i=0; i<n; i++)
{   if (arr[i] == target)
        return true;
}
return false;
```

→ L.S → T.C → O(n)

n → size of array

Binary Search

cond^ → array

array → sorted → asc
sorted → dsc

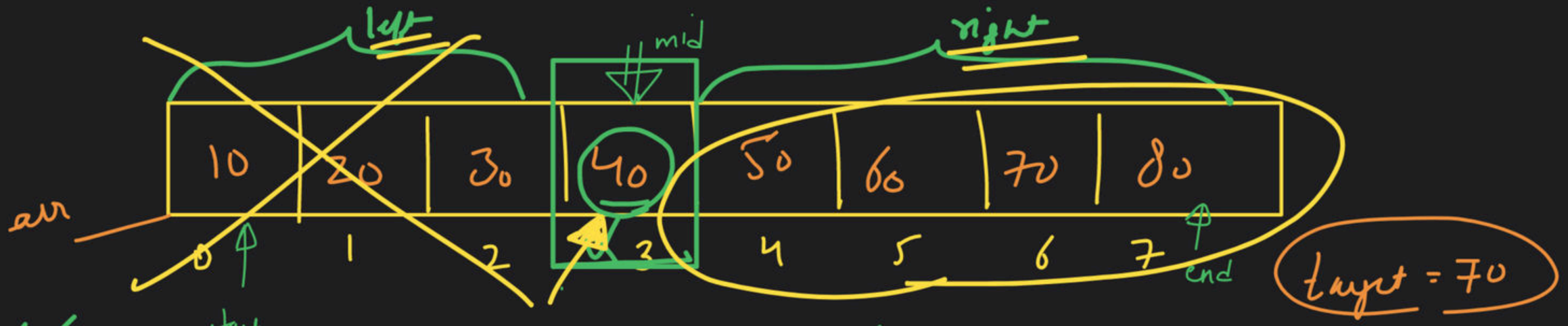
array → monotonic function

ascending

descending

Reload if

no awaz



(A) s, e ✓ ✓

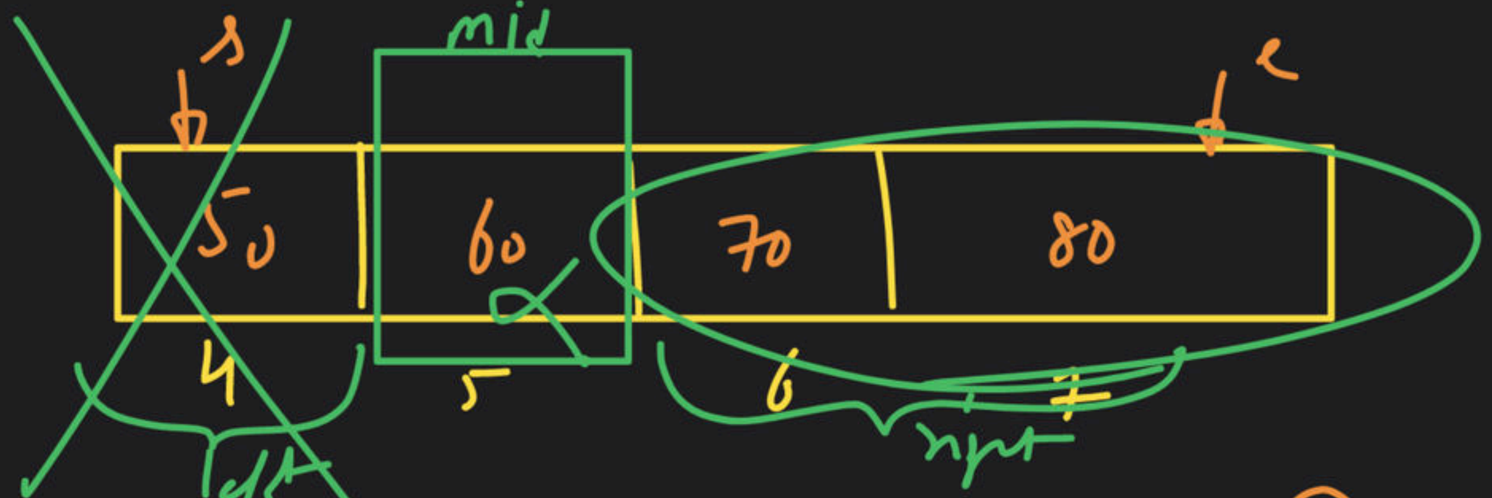
(B) mid = (s+e)/2 ✓

start = 0, end = 7 $\left(\frac{s+e}{2}\right)$

$mid = \frac{0+7}{2} = \frac{7}{2} = 3$

40 == 70 → F

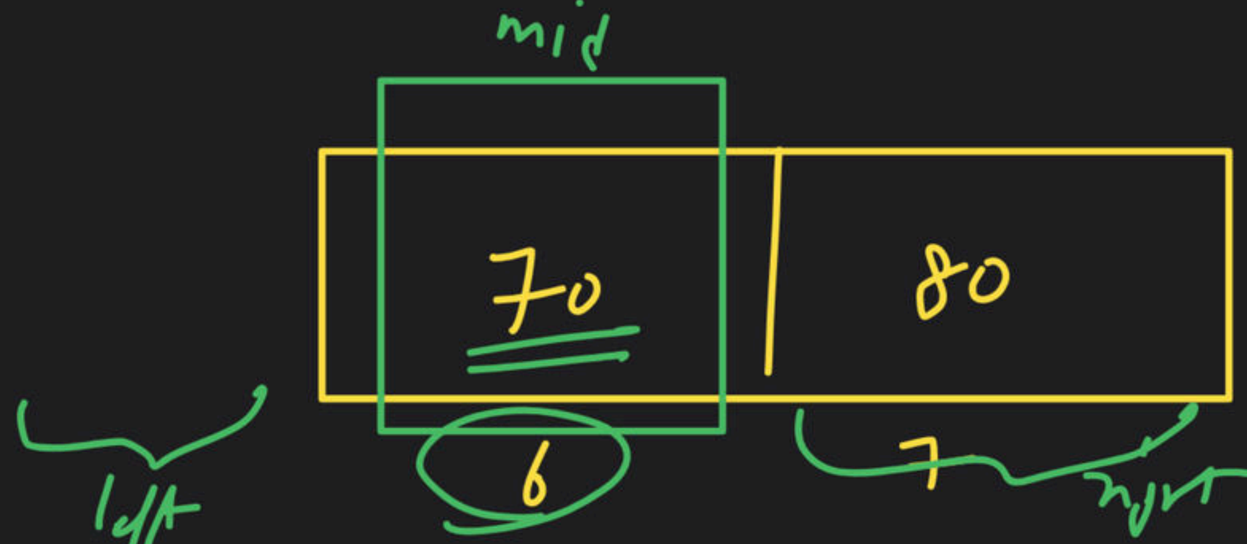
70 > 40 → right



(A) start = 4 , end = 7 , mid = $\frac{4+7}{2}$ = $\frac{11}{2}$ = 5

60 == 70 → F

70 > 60 → Right →

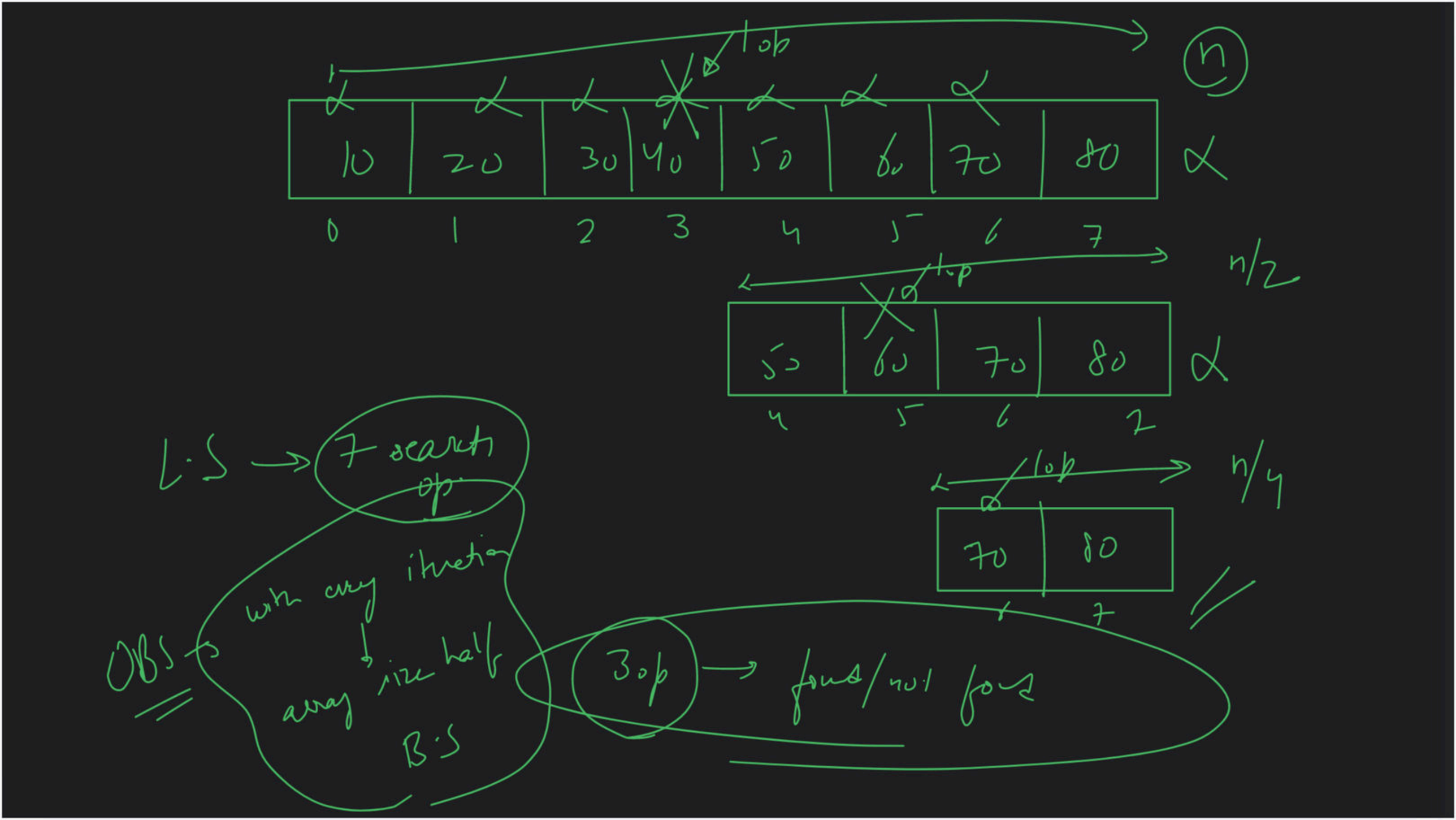


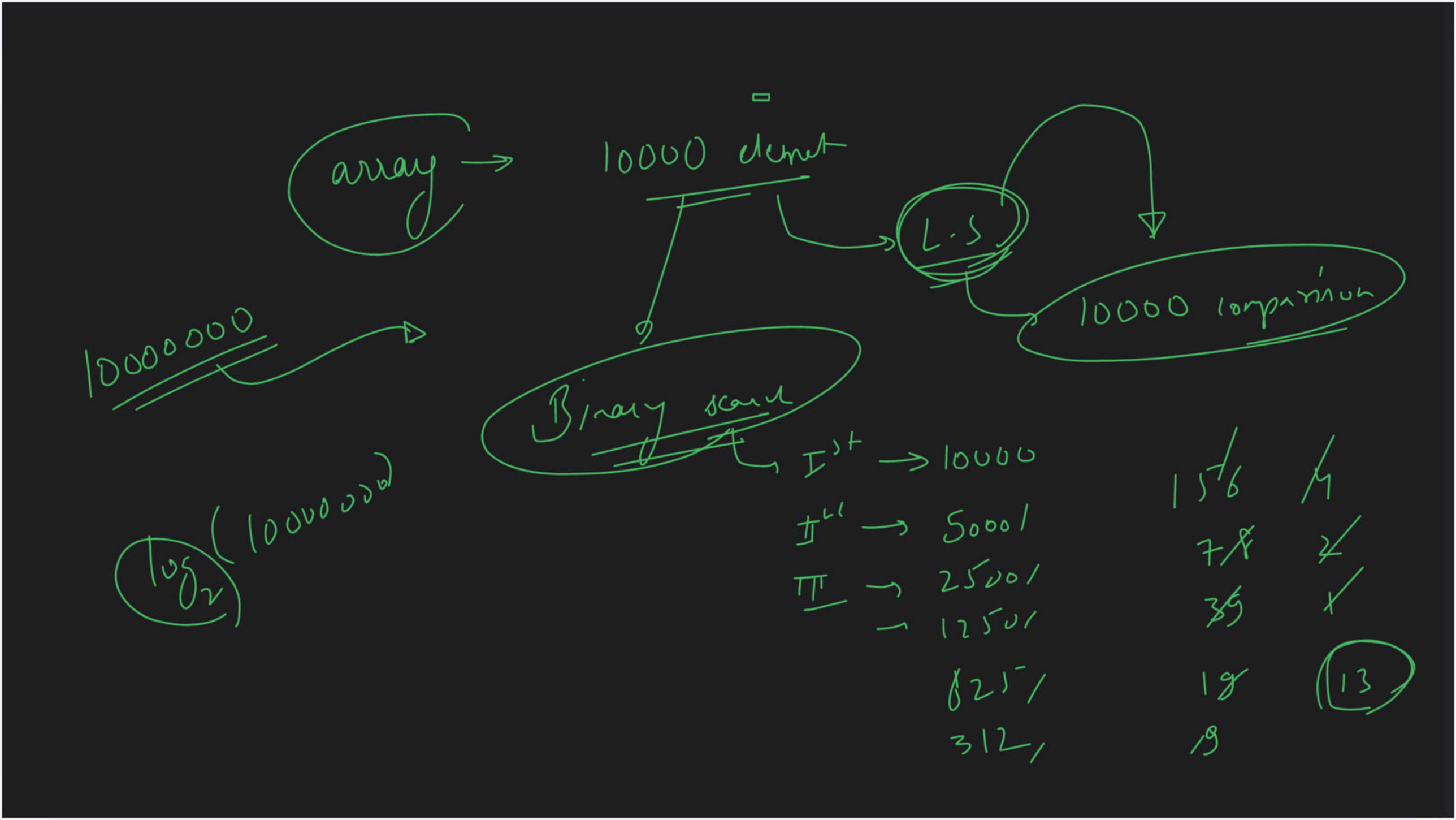
(A) start = 6 , end = 7 , mid = $\frac{6+7}{2} = \frac{13}{2} = 6$

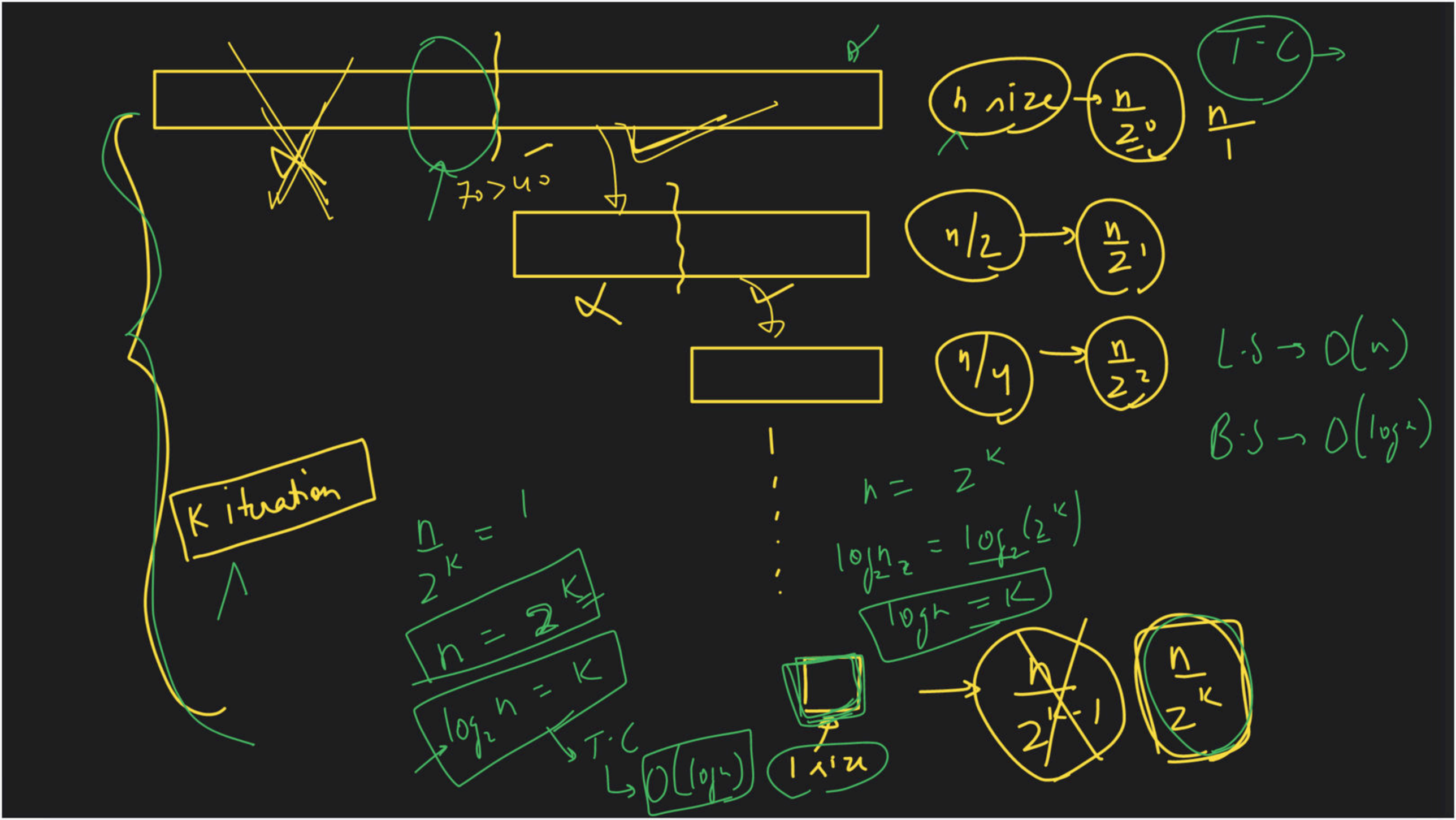
70 == 70 ⟶ Yes ⟶ True

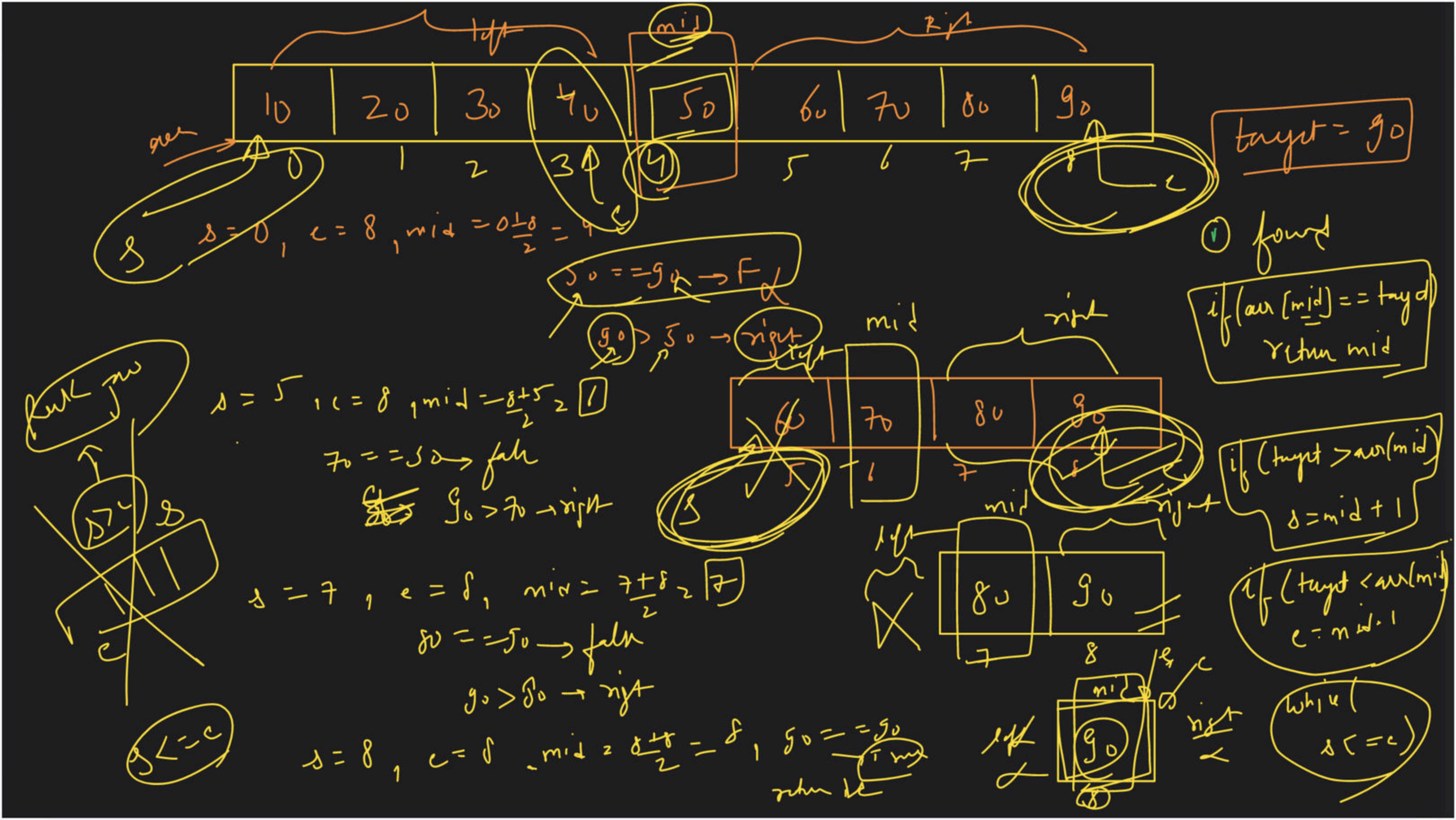
return index

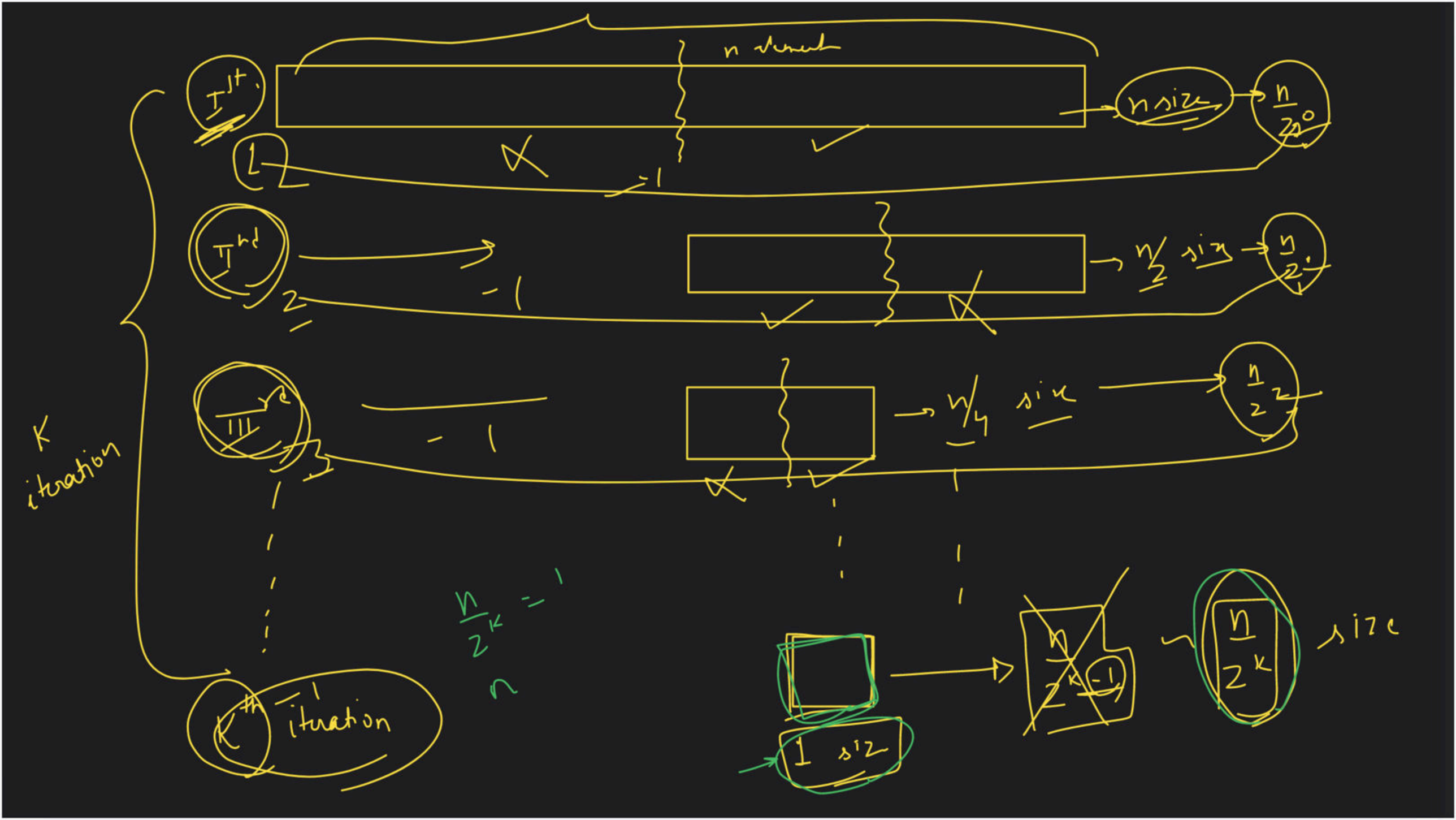
return 6


lob
n
10
20
30
40
50
60
70
80
0
1
2
3
4
5
6
7
α
n/2
lop
n/4
L.S → 7 search op.
with every iteration
array size half
B.S
OBS
3 op → found / not found


array
10000 elements
L.S
10000 comparisons
10000000
Binary search
$\log_2(10000000)$
I st → 10000
II nd → 5000
III rd → 2500
→ 1250
625
312
15
7
3
1
9
13


n size
n/2^0
T.C
n/1
n/2
n/2^1
n/4
n/2^2
L.S → O(n)
B.S → O(log n)
70 > 40
K iteration
n/2^K = 1
n = 2^K
log2 n = K
T.C
O(log n)
1 size
n = 2^K
log2 n2 = log2 (2^K)
log n = K
n/2^K - 1
n/2^K


left
mid
right
10 20 30 40 50 60 70 80 90
0 1 2 3 4 5 6 7
arr
s = 0, e = 8, mid = 0+8/2 = 4
target = 90
① found
if(arr[mid] == target)
return mid
s = 5, e = 8, mid = 5+8/2 = 6
70 == 50 → false
90 > 70 → right
s = 7, e = 8, mid = 7+8/2 = 7
80 == 50 → false
90 > 80 → right
s = 8, e = 8, mid = 8+8/2 = 8, 90 == 90
return 8
s <= e
if(target > arr(mid)
s = mid + 1
if(target < arr(mid)
e = mid - 1
while(
s <= e)


n elements
1st
n size
n/2^0
2nd
n/2 size
n/2^1
3rd
n/4 size
n/2^2
K iteration
Kth iteration
n/2^k = 1
n
1 size
n/2^k
size

$$\frac{n}{2^k} = 1$$

$$n = 1 \times 2^k$$

$$n = 2^k$$

apply both sides $\rightarrow \log_2$

$$\log_2 n = \log_2 (2^k)$$

$$\log_2 n = k$$

$\frac{a}{b} \Rightarrow c$

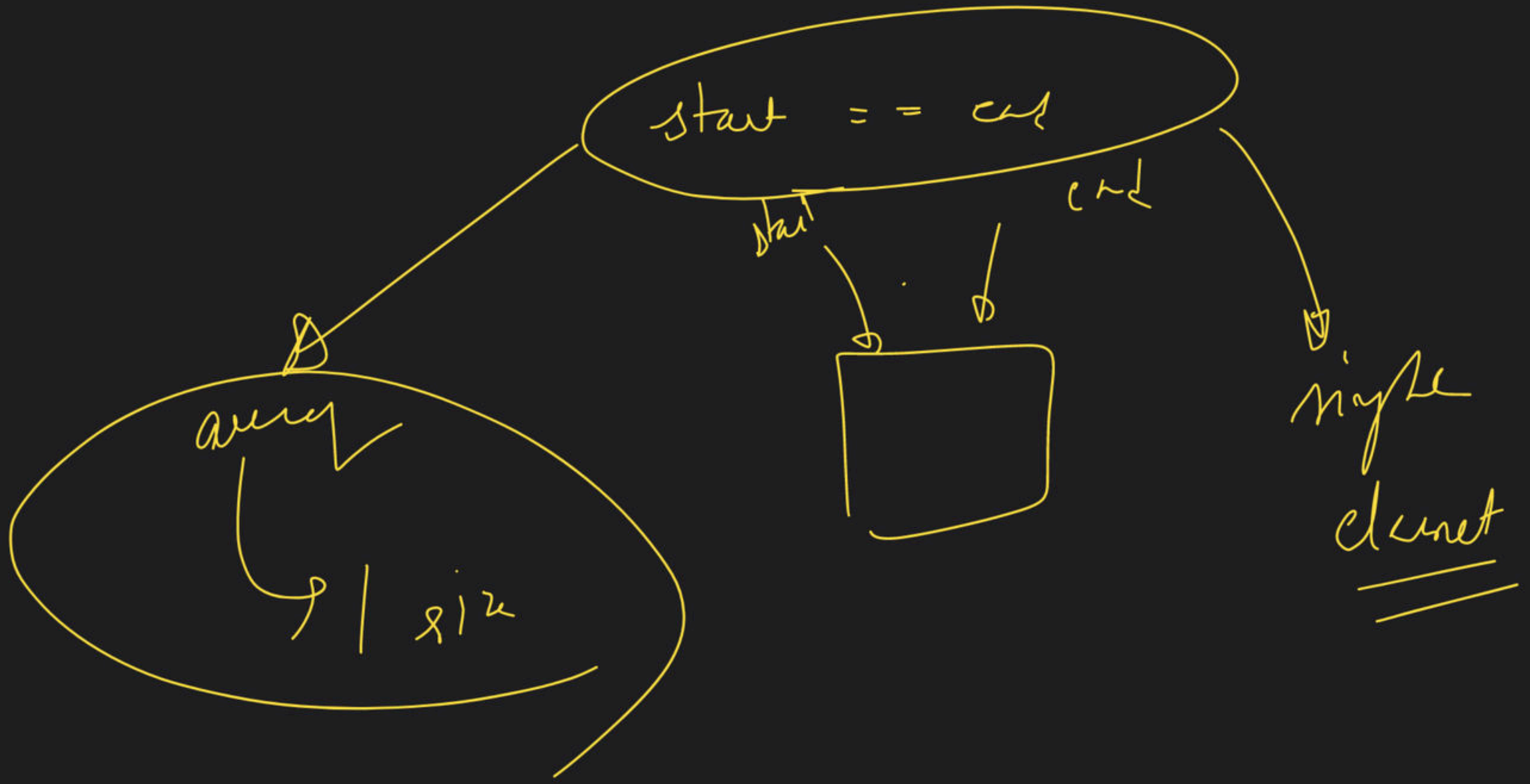
$a = c \times b$

$1 \times a \rightarrow a$

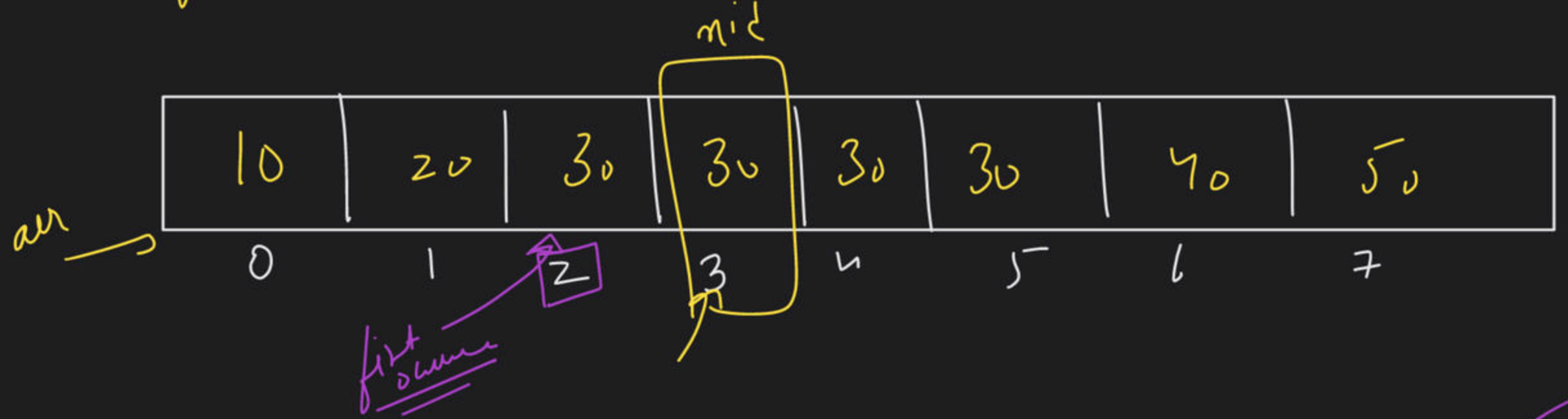
~~$k = ?$~~

T.C $\rightarrow O(k)$

$O(\log_2 n)$


start == end
start
end
right
element

→ find first occurrence of a number in a sorted array



target = 30

ans = 2

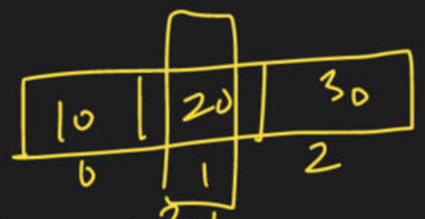
→ s=0, e=7, mid = 3

arr[mid] == target

30 == 30 → True → found

ans = 3

left → e = mid-1



→ s=0, e=3-1, e=2 → mid 2

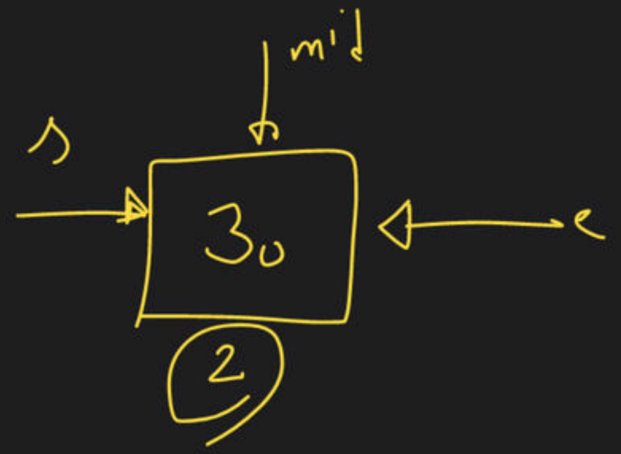
20 == 30 → ✗

30 > 20 → ✓ → right → s=mid+1 = 1+1 = 2

2 min

Sorted

found → ans store → left me chala gaya



int mid = s + (e-s)/2;

s = 2 , e = 2 , mid = 2+2/2 = 2

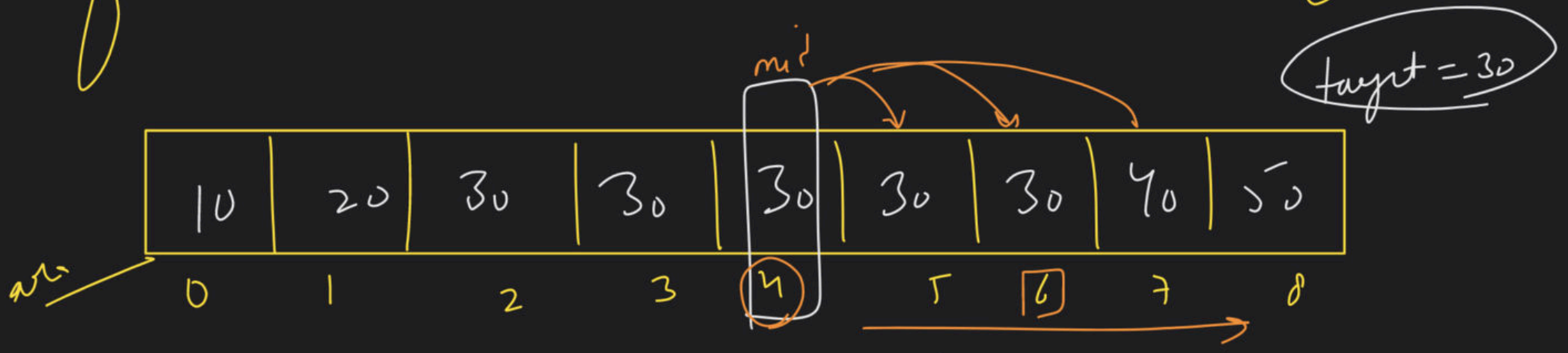
arr[mid] = = target

30 = = 30 → found → ans = 2

→ left me jao
e = mid - 1
e = 2 - 1
e = 1

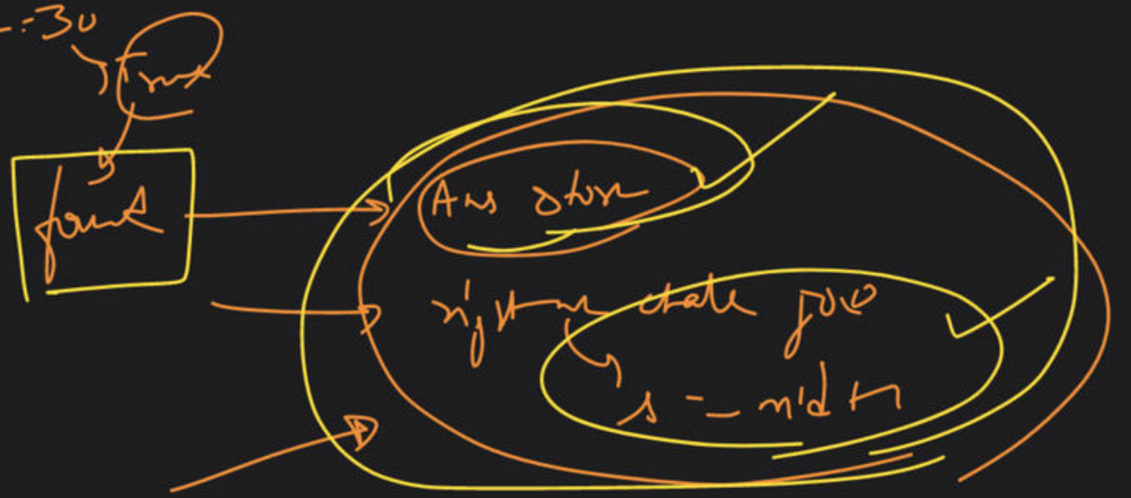
→ s = 2 , e = 1 → s > e → Bahar jao

→ find last occurrence → sorted array



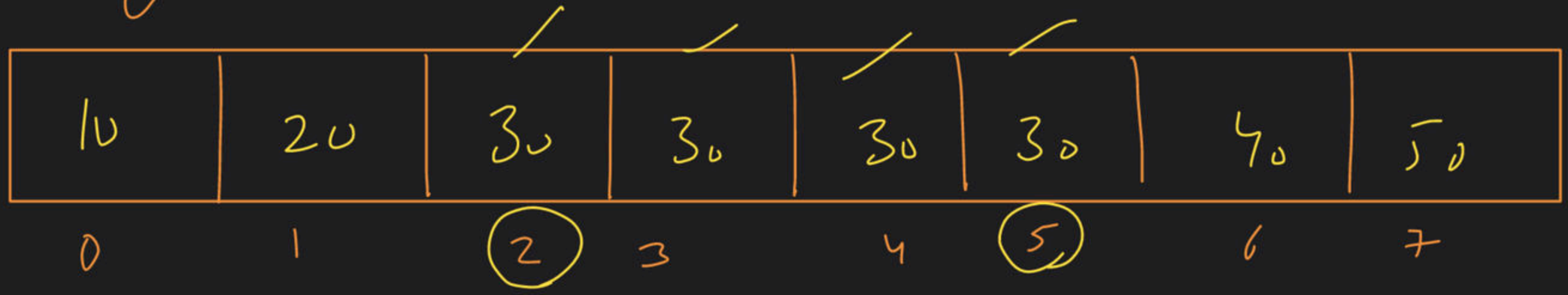
s = 0
e = 8
→ mid = 4

arr[mid] = 2 target
30 = 30



Find total occurrence

target = 30

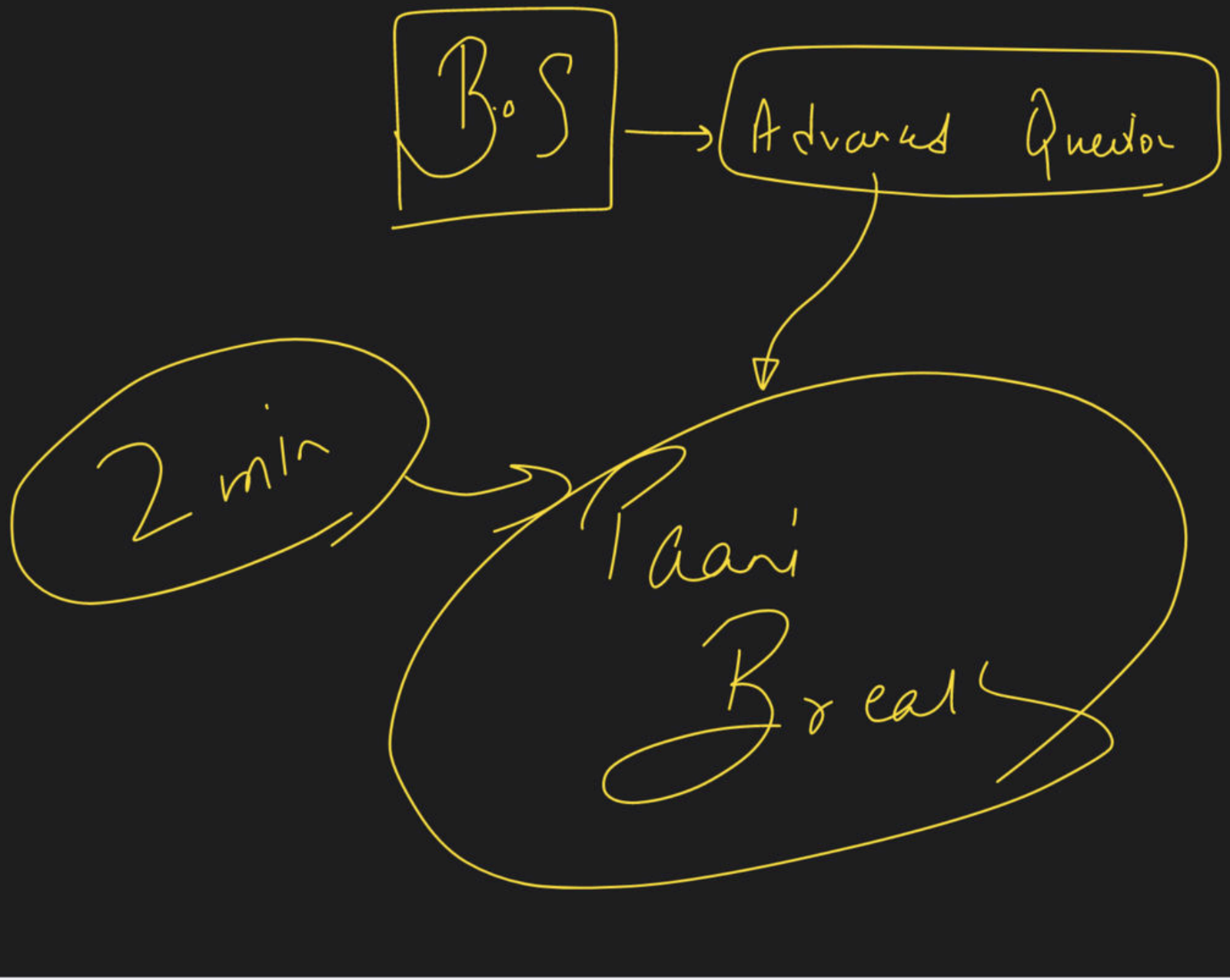


Total Occ → 4

first Occ = 2

last Occ = 5

Total Occ = last Occ - first Occ + 1


B.o S
Advanced Questions
2 min
Paani
Break

int mid = (s + e) / 2

INT_MAX + INT_MAX

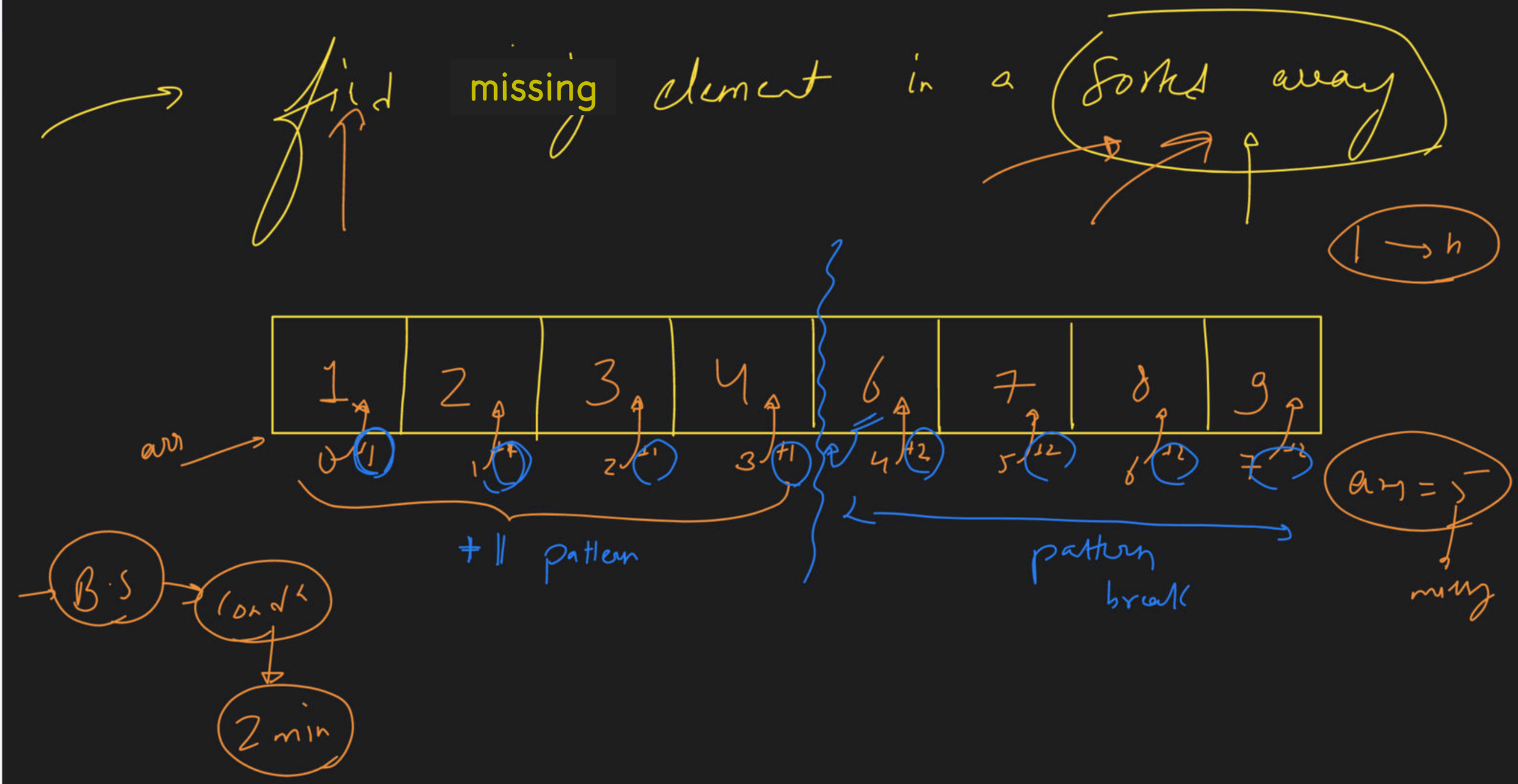
(s + e) / 2

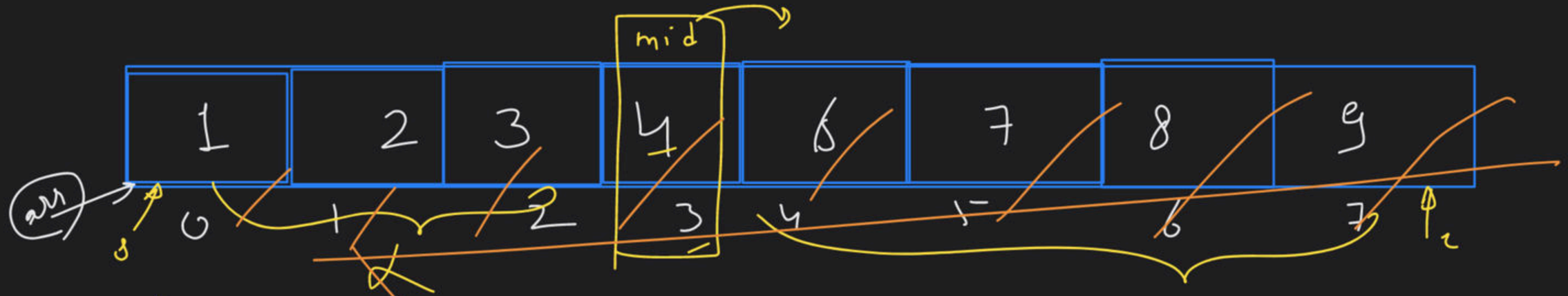
s + (e - s) / 2

s + (e - s) / 2 → s + e/2 - s/2

→ s/2 + e/2 → (s + e) / 2

find missing element in a sorted way





$s = 0$
$e = 7$
$\rightarrow$ $mid = \frac{0+7}{2} = 3$

$diff = arr[mid] - mid$
$= 4 - 3$
$diff = 1$

$diff == 1$ $\rightarrow$ right $\rightarrow s = mid + 1$

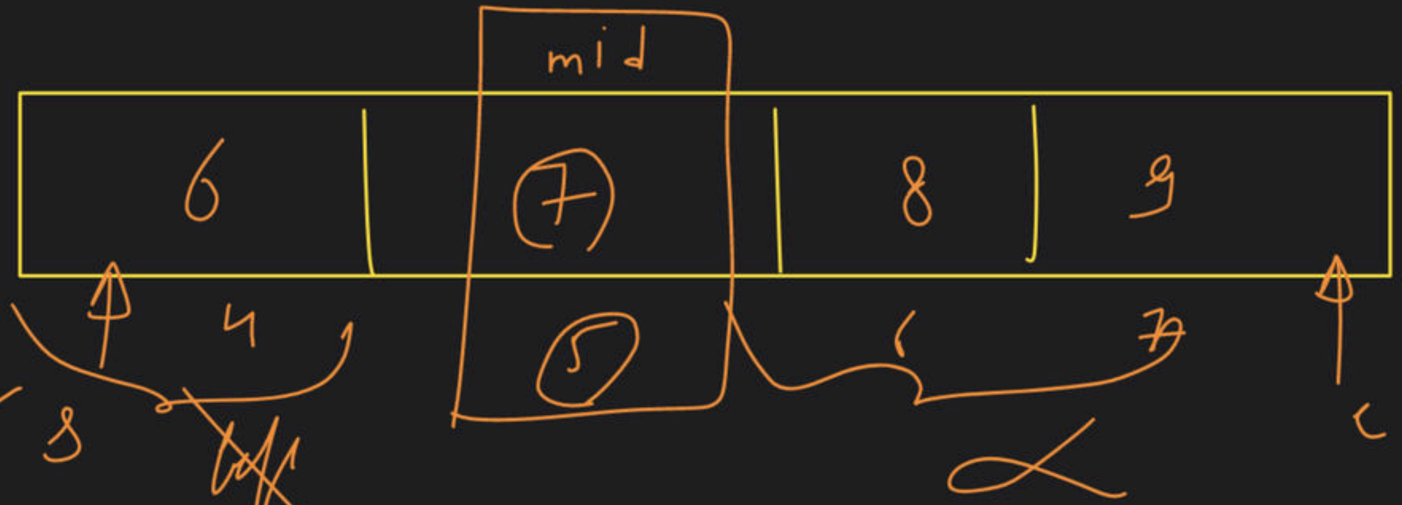
i/p $\rightarrow$ (n-1) integer

& integer

$1 \rightarrow y$

$1 \rightarrow n$

$1 - y$



s = 4

e = 7

→ mid = $\frac{4+7}{2}$ → 5

diff = arr[mid] − mid

= 7 − 5

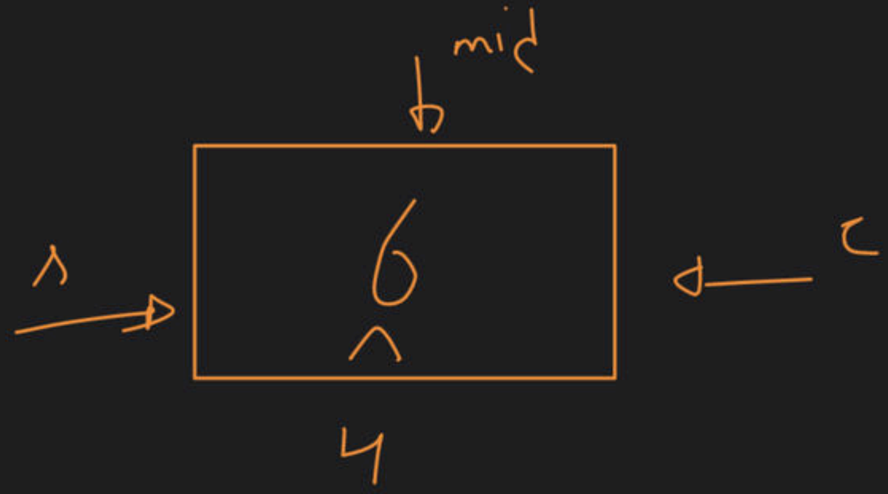
diff = 2

diff == 1 → false → ans = arr[mid]

ans = 7

→ left me jaye

e = mid − 1



s = 4
e = 4
→ mid = 4+4/2 = 4

diff = arr[mid] - mid
4 - 4

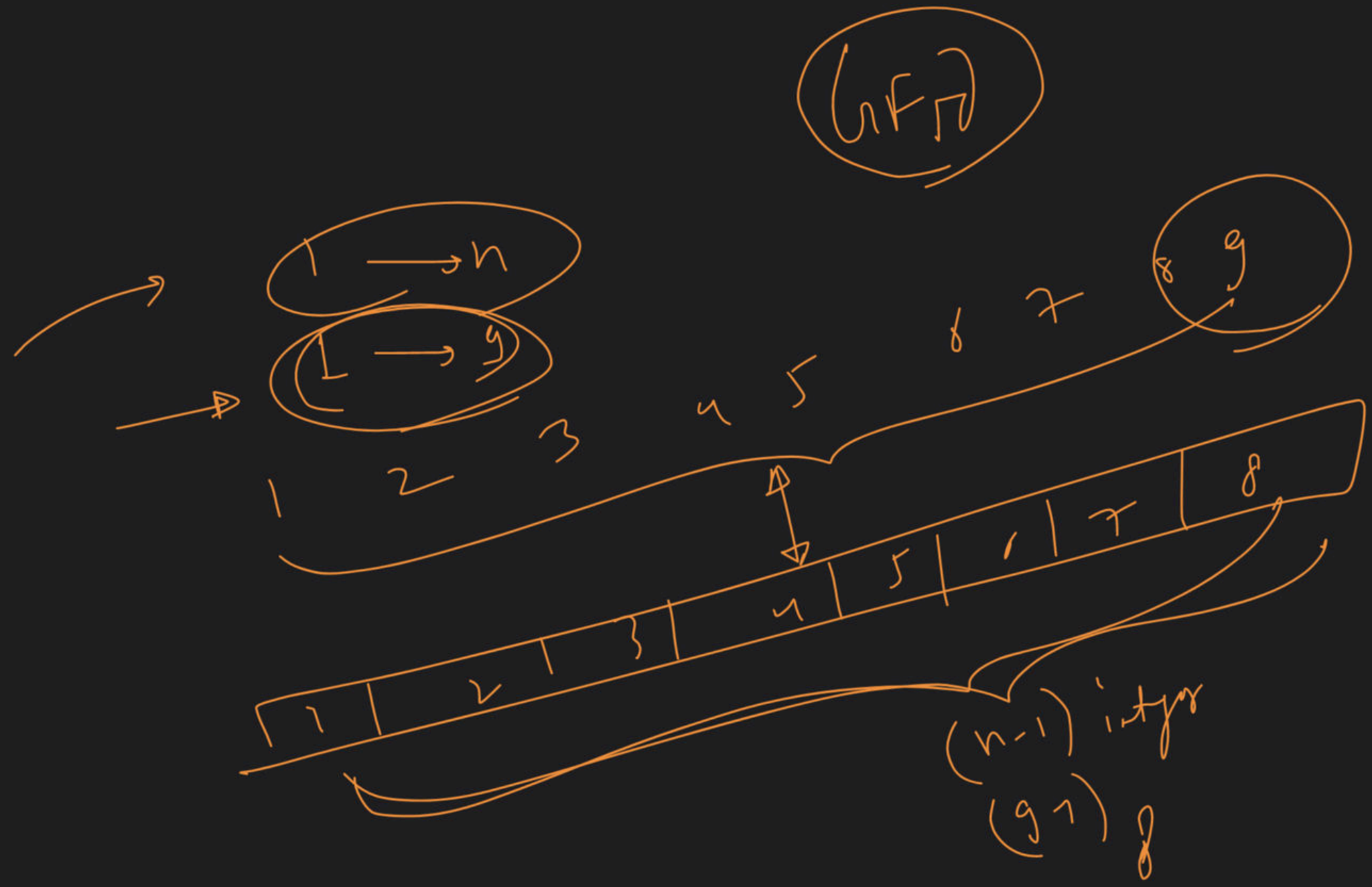
diff = 0

diff == 1 → No → ans = 4

left → e = mid - 1

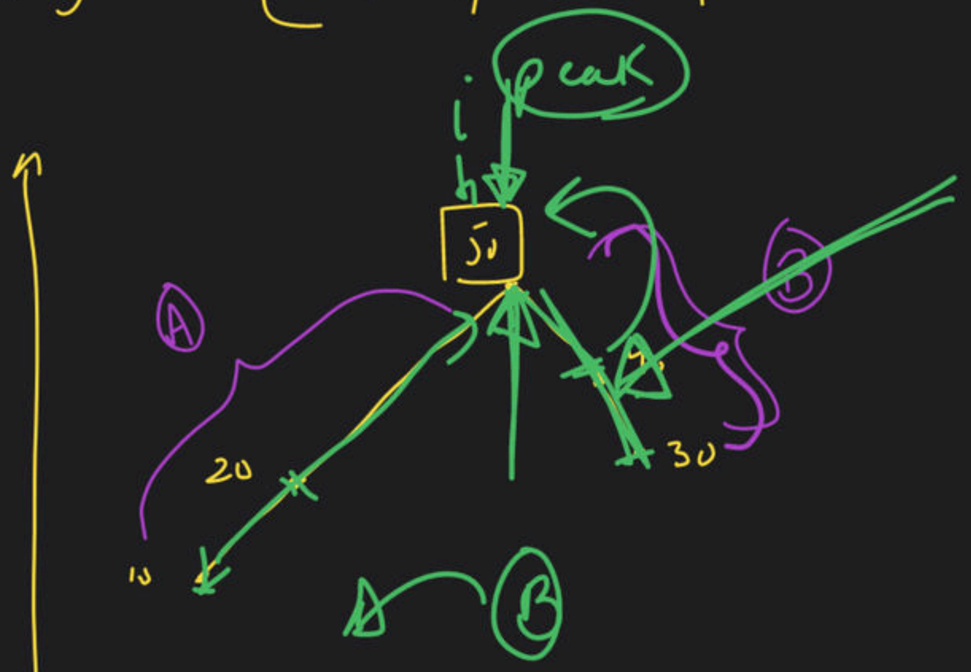
s = 4, e = 3 → s > e → while loop

findAns → ans


(n-1) integers
(97) B

Peak element in a Mountain Array

i/p → arr[] = { 10, 20, 50, 40, 30 }



ans = 50 → o/p

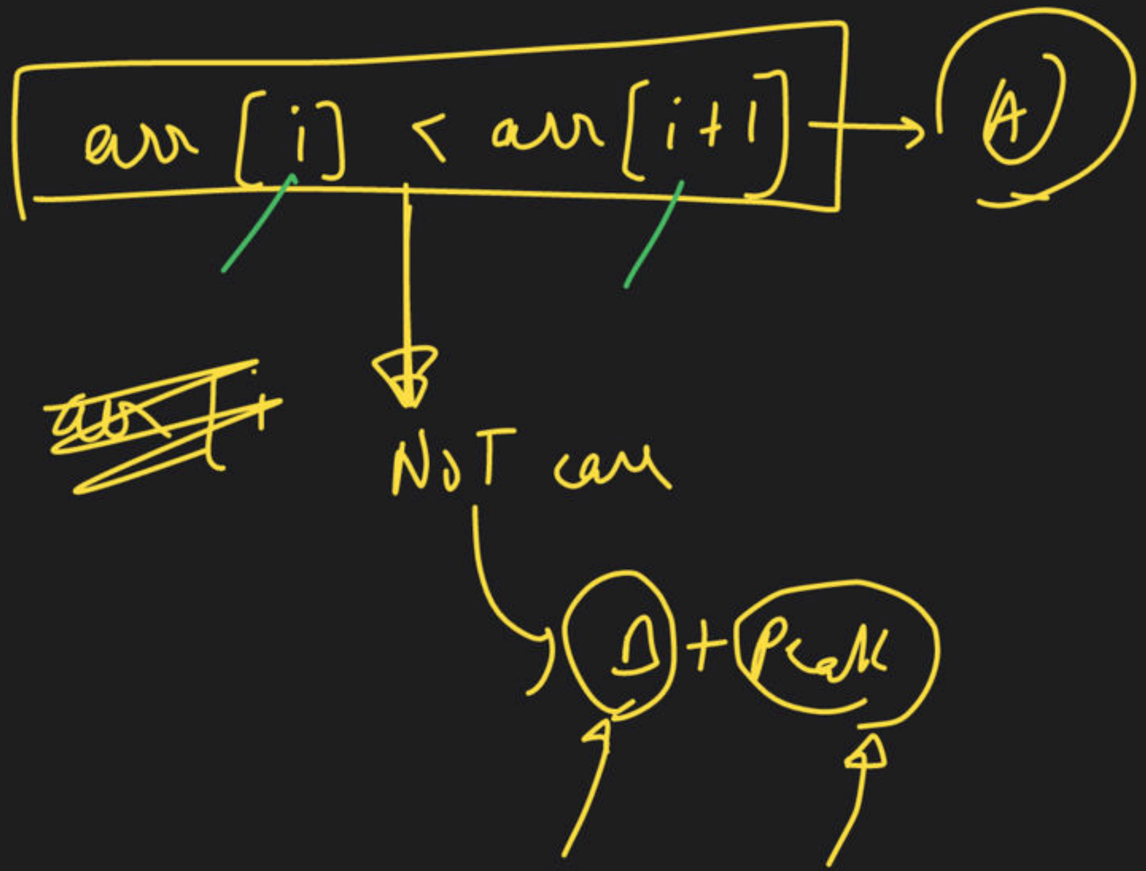
Kaun → ?

peak point

A
arr[i] < arr[i+1]

B
arr[i] > arr[i+1]

arr[i-1] < arr[i] > arr[i+1]



Observation :-

A

arr [i] < arr [i+1]

B

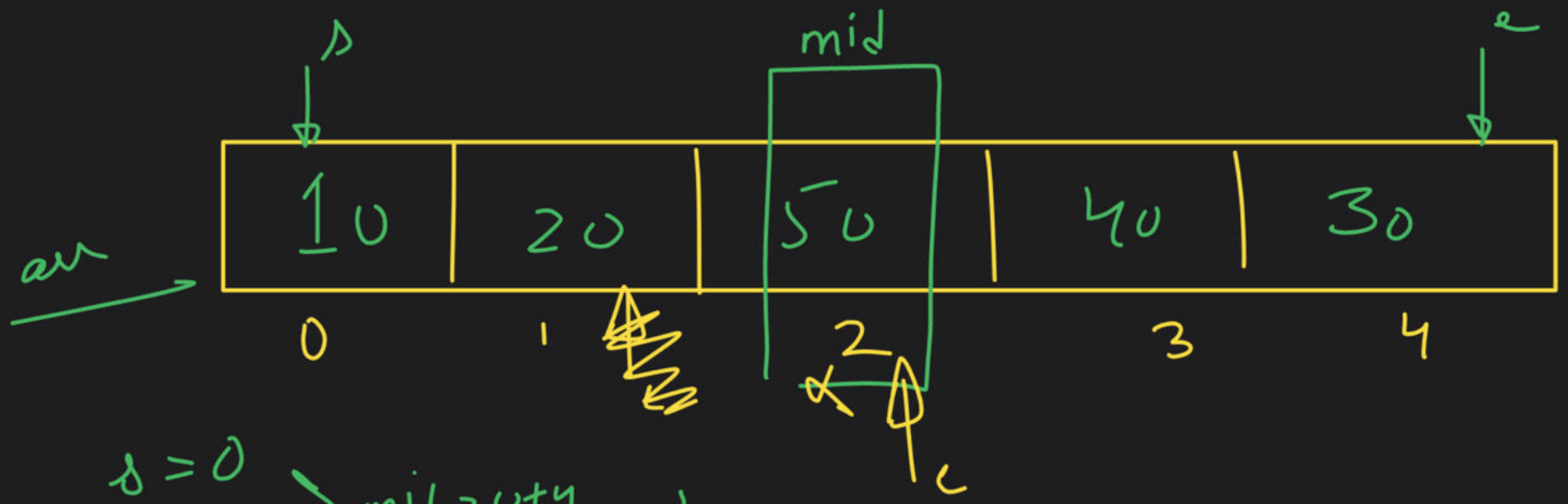
arr [i] > arr [i+1]

Peak

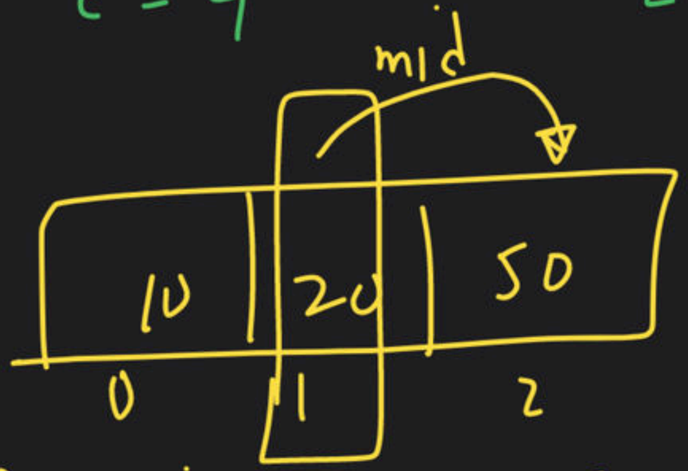
we can club

arr [i] > arr (i+1)

arr [i] > arr (i-1)



s = 0
e = 4
> mid = $\frac{0+4}{2}$ = 2



s = 0
e = 2
> mid = 1

20 < 50 → True

Right.

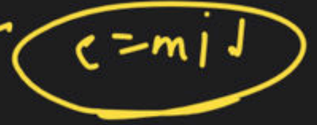
arr[mid] < arr[mid+1]

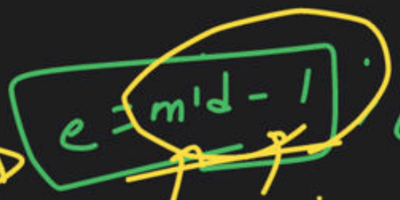
50 < 40 → false →

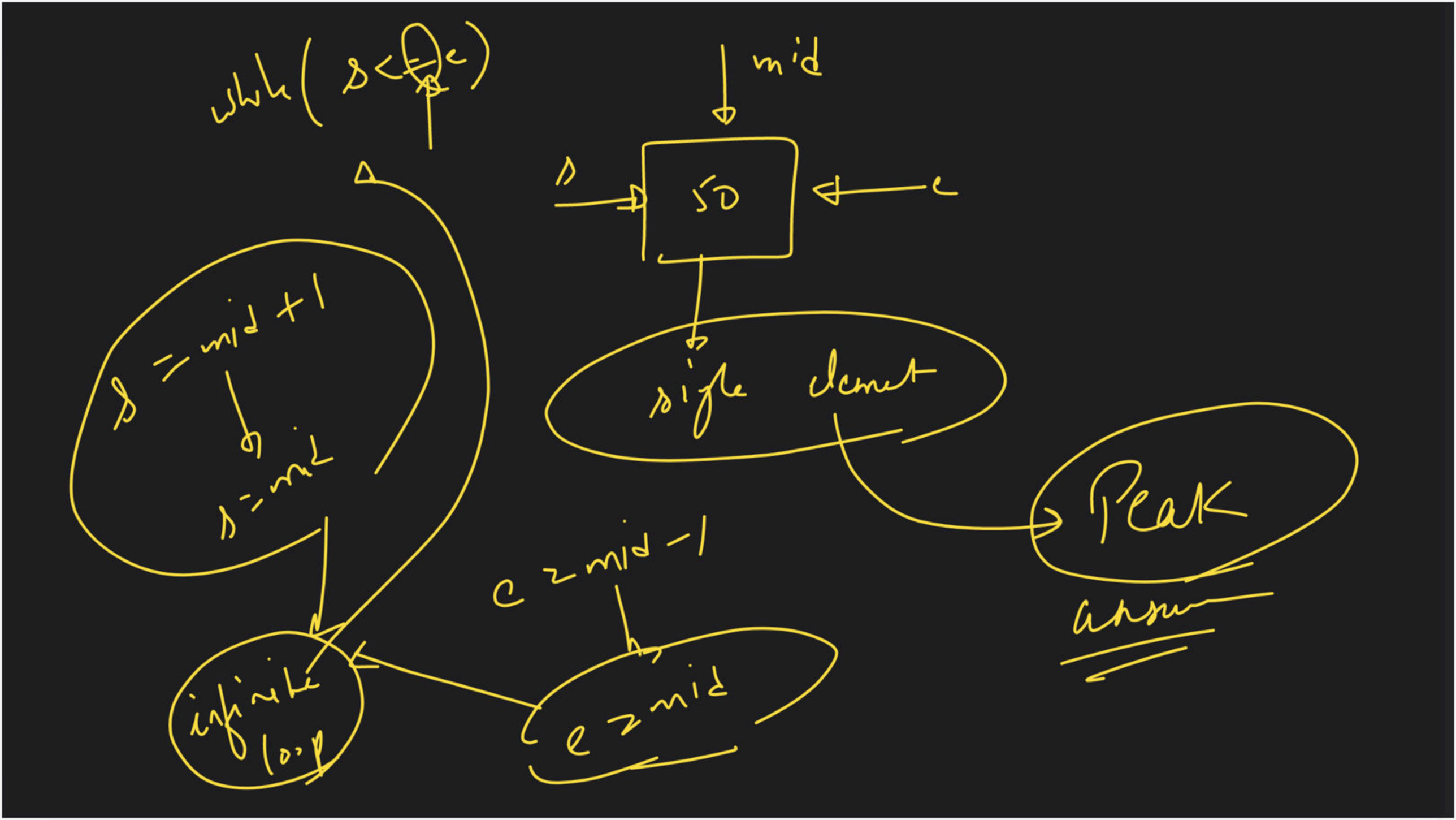
left

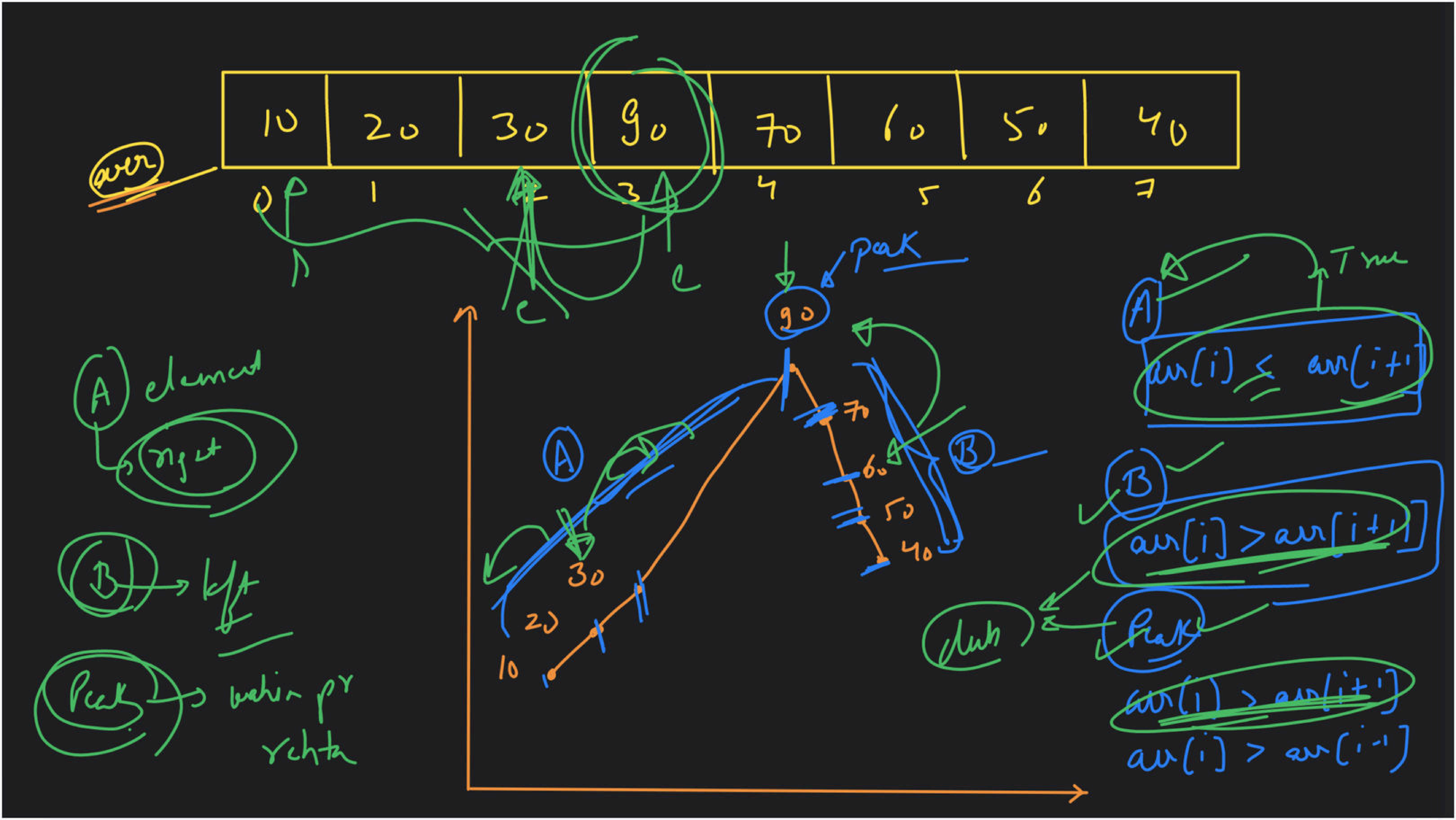
last mein hai

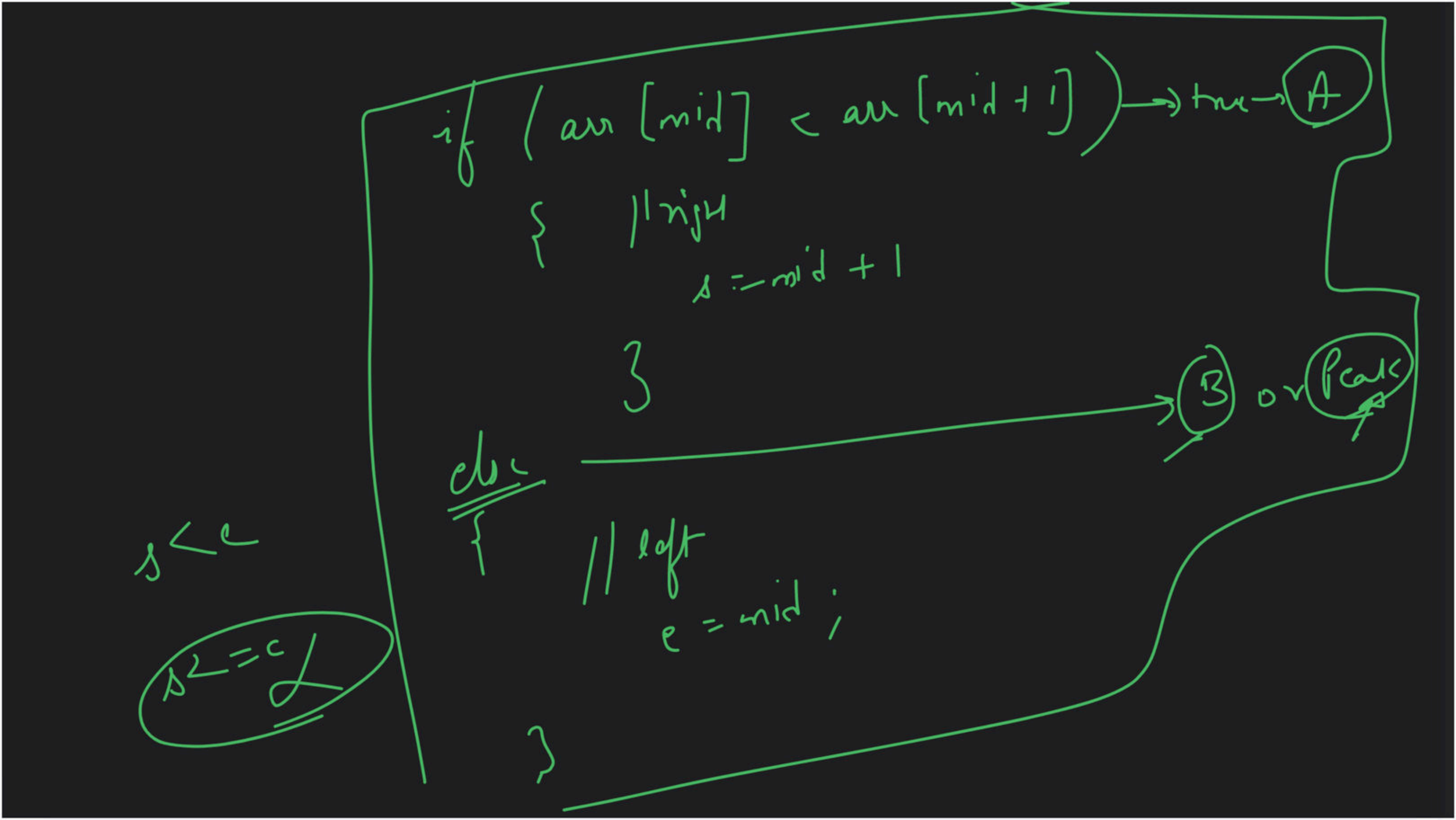
e = mid - 1

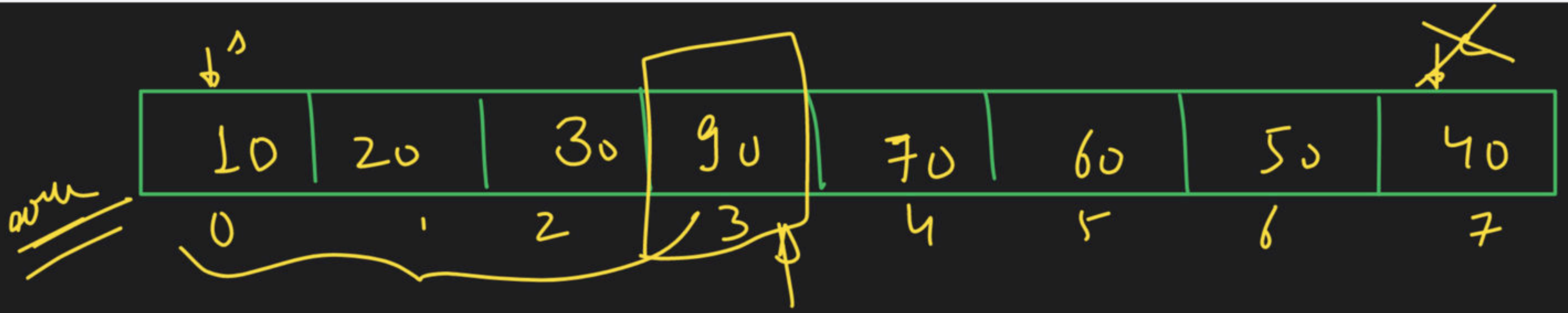





while(s<=e)
mid
s
isD
e
single element
s = mid+1
s=mid
e=mid-1
e=mid
infinite loop
Peak
answer


arr
10 20 30 90 70 60 50 40
0 1 2 3 4 5 6 7
Peak
90
70
60
50
40
30
20
10
A
B
A element
right
B left
Peak
arr[i] ≤ arr[i+1]
True
arr[i] > arr[i+1]
arr[i] > arr[i+1]
arr[i] > arr[i-1]


if (arr[mid] < arr[mid+1]) → true → A
{ //right
s = mid + 1
}
else
{
//left
e = mid;
}
B or Peak
s < e
s == e

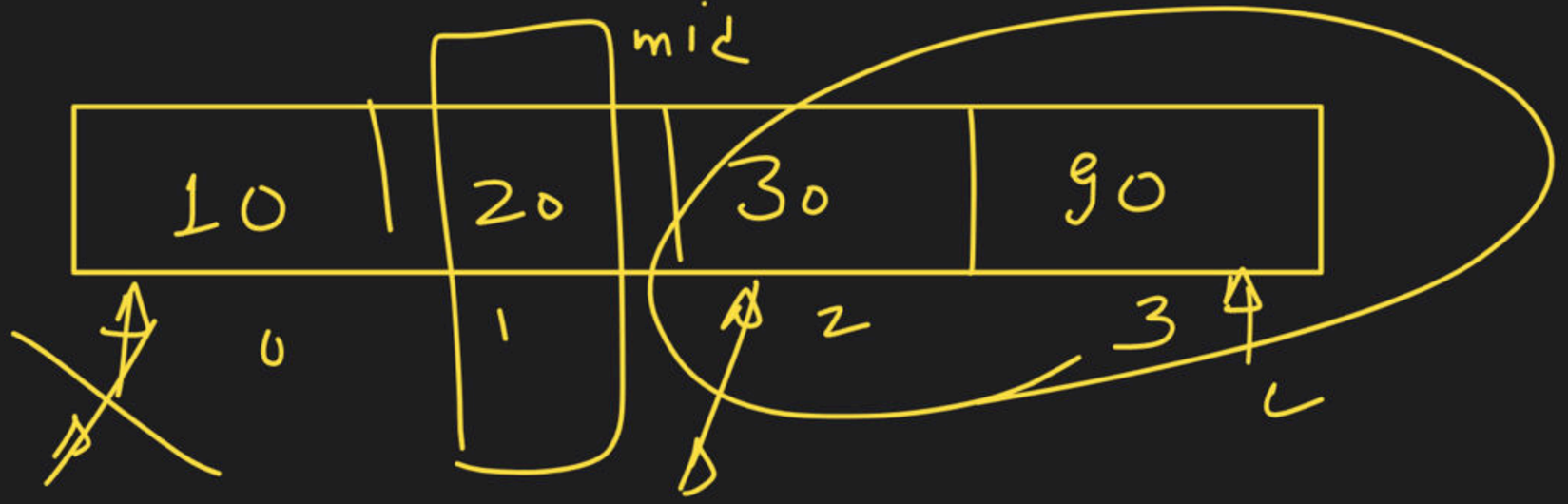


s=0
e=7

mid → $\frac{0+7}{2}$ = 3

arr[mid] < arr[mid+1]

90 < 70 → false → e=mid



s = 0

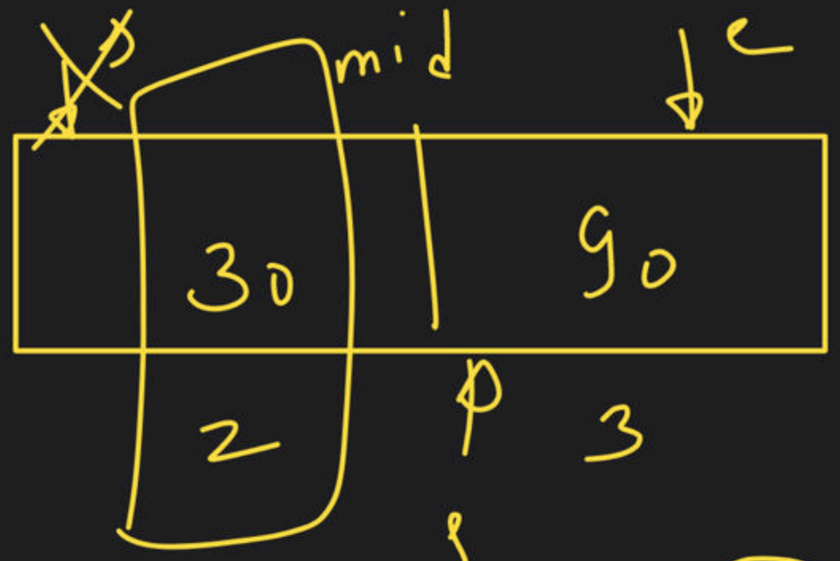
e = 3

mid = $\frac{0+3}{2}$ = 1

arr[mid] < arr[mid+1]

20 < 30 → true

↳ Right



s = 2

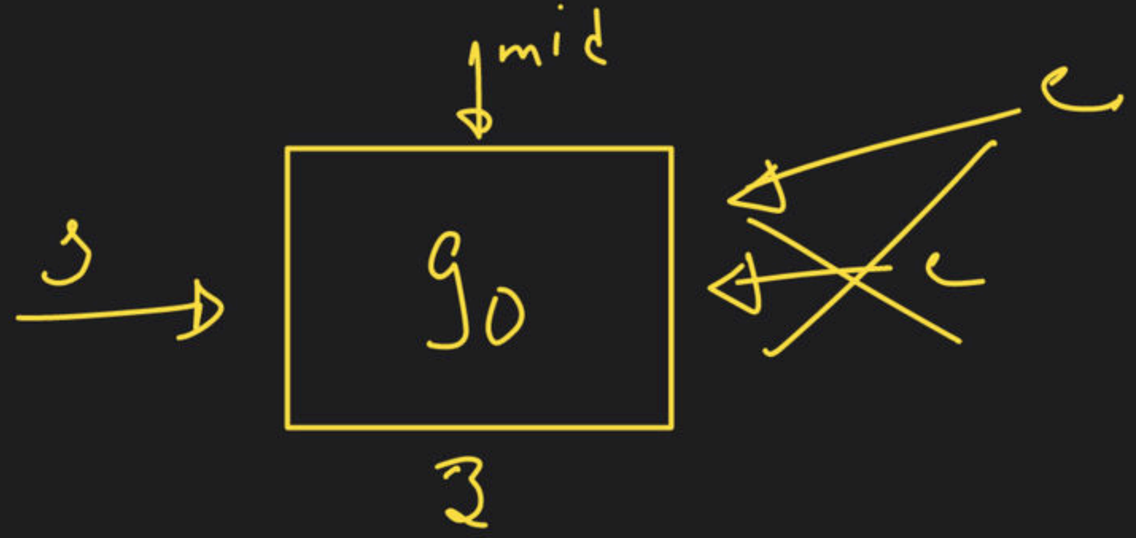
e = 3

$mid = \frac{2+3}{2} = 2$

arr[mid] < arr[mid+1]

30 < 50 → true

s = mid+1



s = 3

e = 3

mid = $\frac{s+e}{2}$ = 3

arr[mid] < arr[mid + 1]

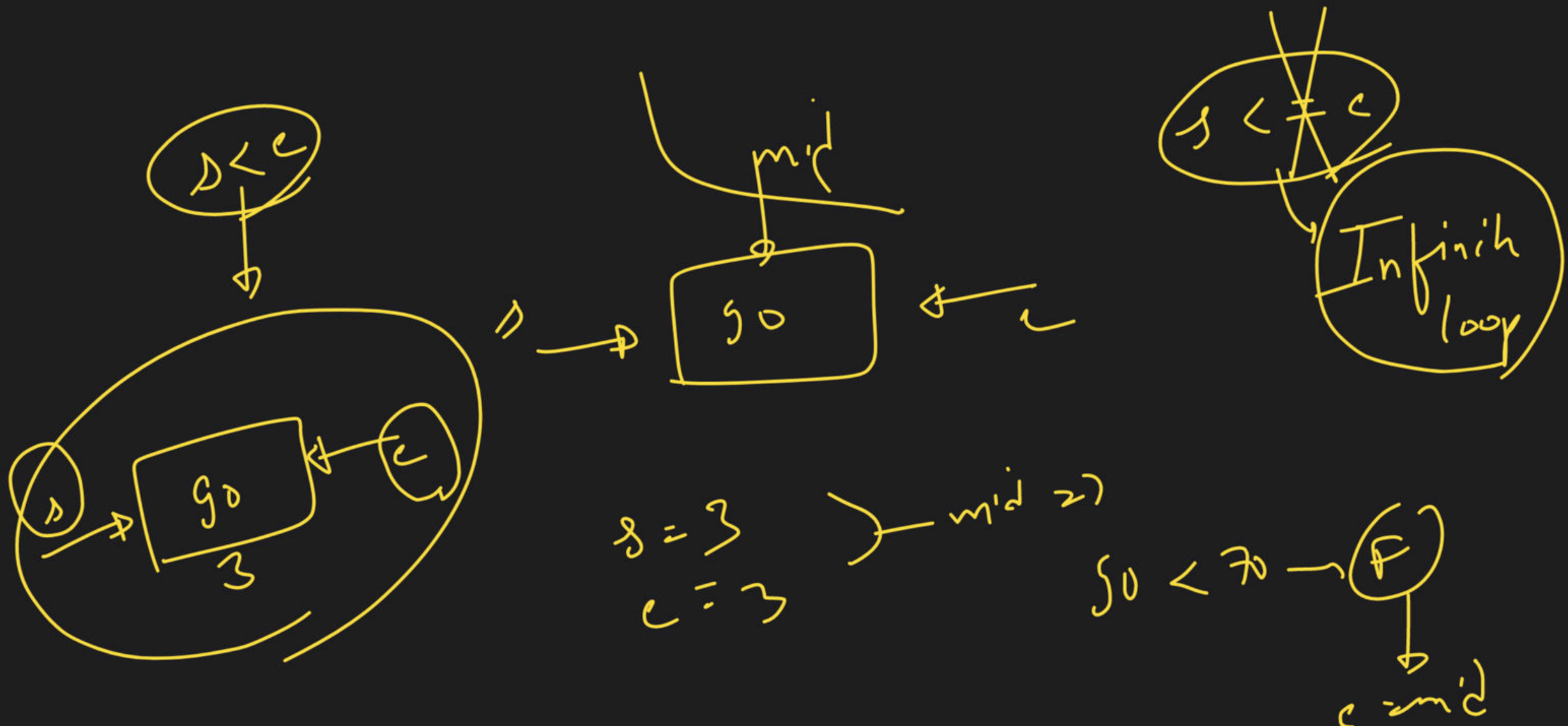
g_0 < 70 → false

e = mid

g = 3
c = 3

→ mid > 3 → g0 < 7 → F

c = mid


s<=e
mid
go
s
go
3
e
s=3
e=3
mid=2
go<70
F
e=mid
s<=e
Infinite loop


find pivot element
LFG