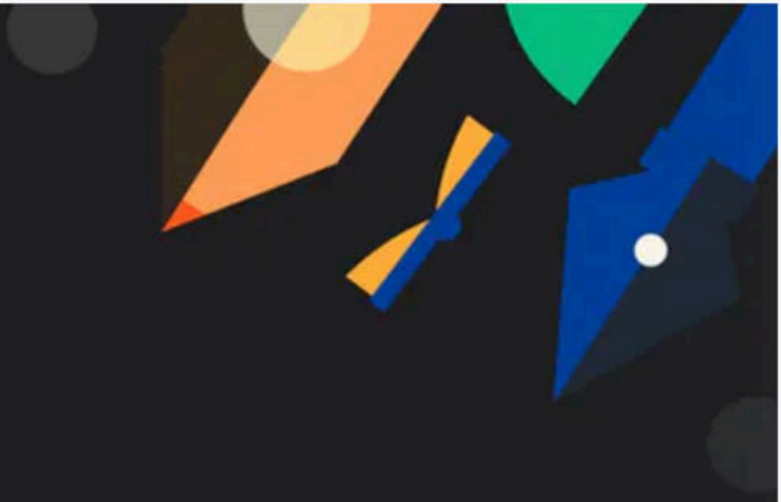


Doubt Clearing Session - Part IV

Foundation Course on Data Structures & Algorithm - III

Love Babbar • Lesson 4 • May 8, 2022

→ Doubts :-

3 classes

→ Binary Search
→ Classic ← Codes
→ 2D
→ Search Space
BA
PP
AC
Roti-P
EKO
LVC

→ Discord link in Chat

→ Implement B.S

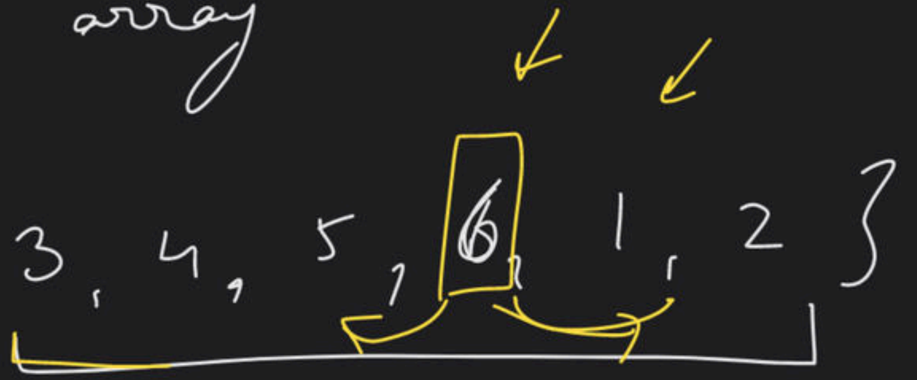
→ first / Last Occurrence

→ total no. of Occurrence → Last − first +1

→ fixed point → (value = index)

→ find pivot in an array

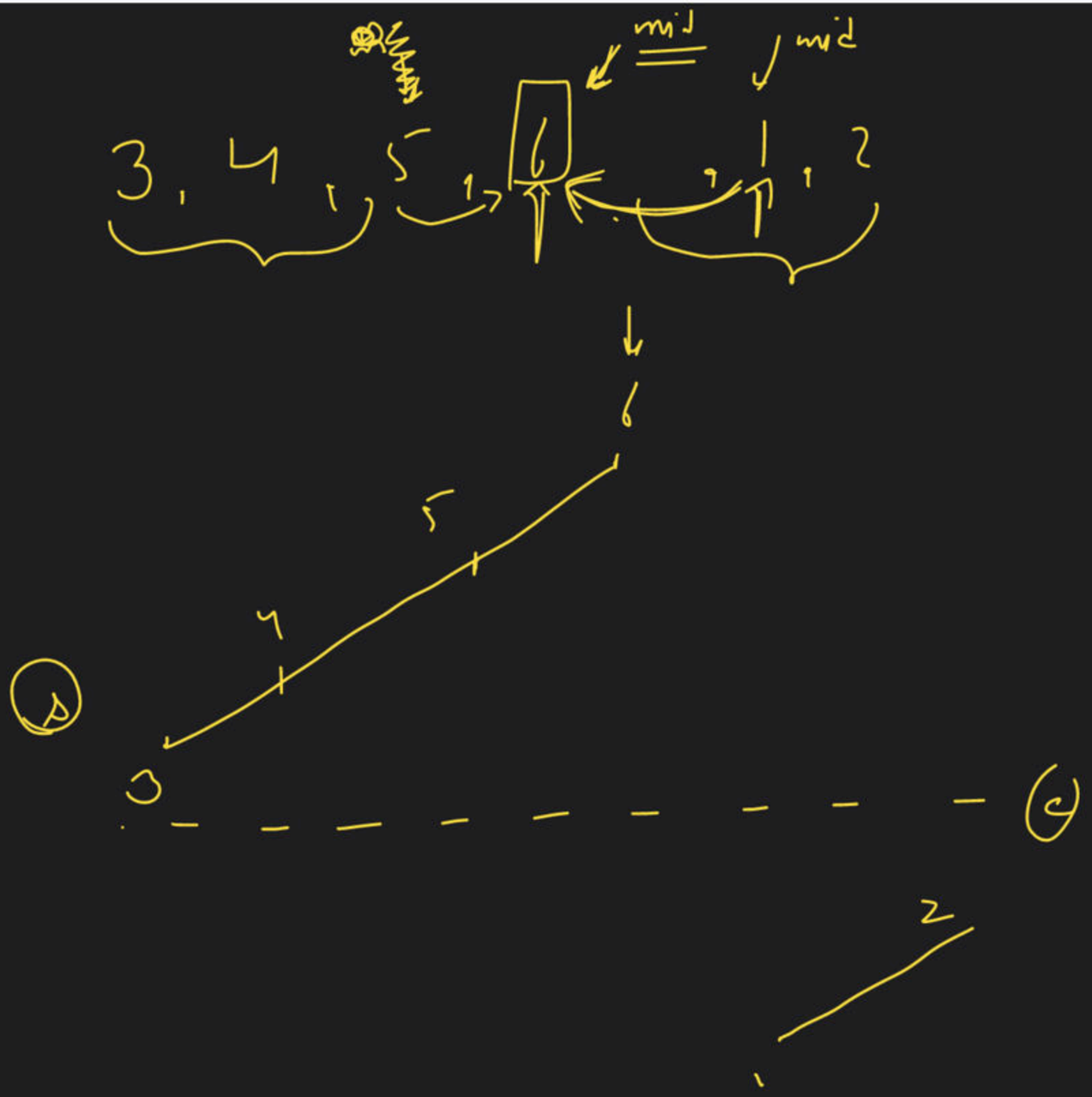
↳ i/p → { 3, 4, 5, 6, 1, 2 }

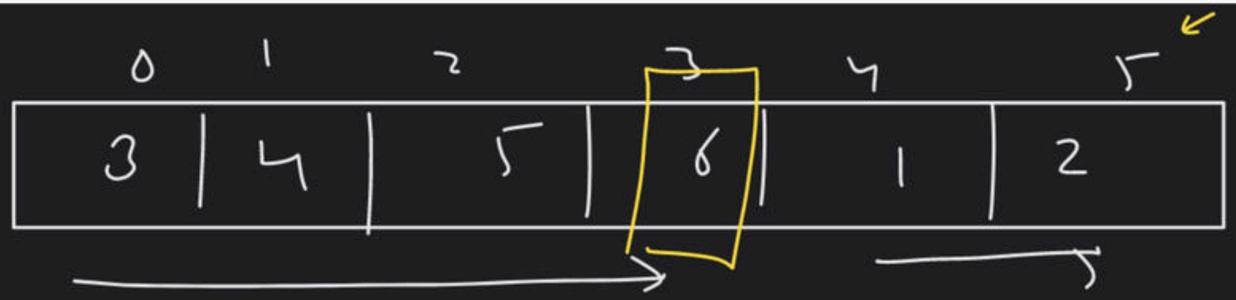


pivot → 6

#1 Brute force → linear search -> max element → O(n)

#2 -> BS → O(log n) → How -> ?


mid
mid
3
4
5
6
7
8
9
6
5
6
5
3
2



[sorted & rotated]

i/p

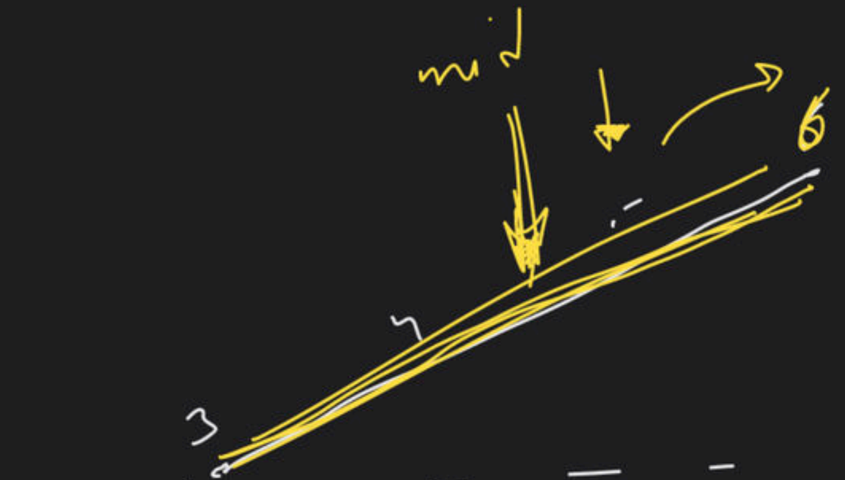
5 > 6 ✗

5 < 4 ✗

s = 0

e = 5

mid = 2



if (arr[s] <= arr(mid))

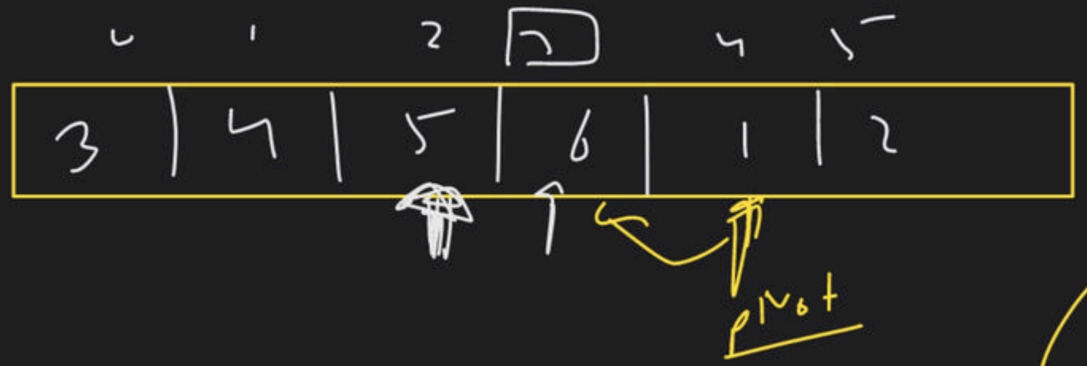
Left

next

```
if (arr[mid] > arr[mid+1])
    return mid;
```

prev

```
if (arr[mid] < arr[mid-1])
    return mid-1;
```



mid = (0 + 5)/2 = 2

mid < e

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

return mid

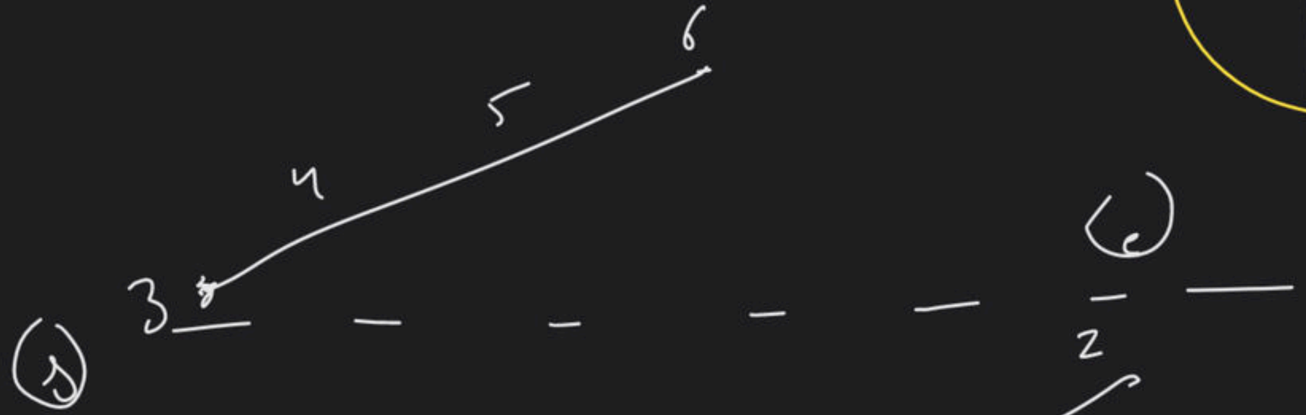
mid > 0

if (arr[mid] < arr[mid-1])

return mid-1

next

prev



Left / Right

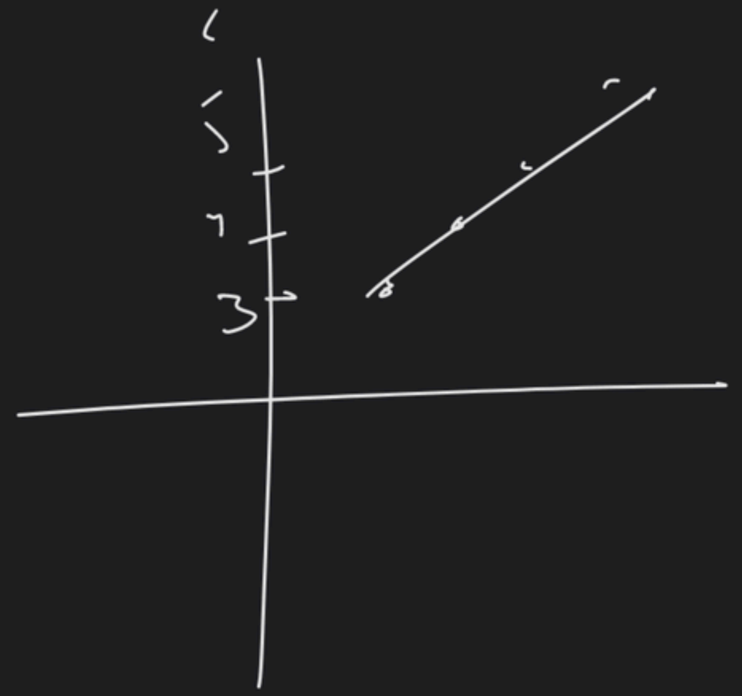
if (arr[s] ≠ arr(mid))

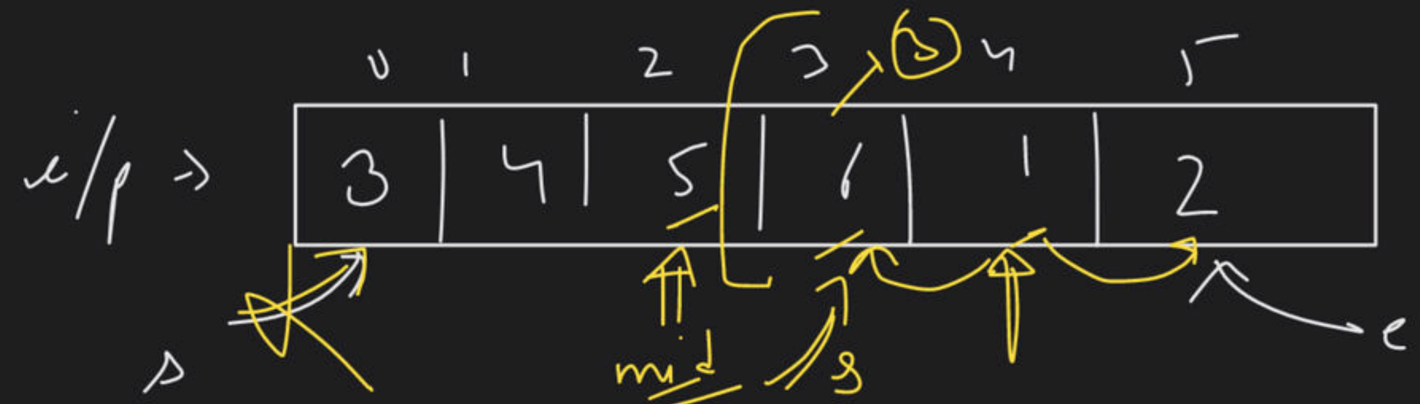
Left → e = mid - 1

else s = mid + 1;

if (arr[s] < arr(mid))

Right





Index of Pivot =>

$mid = (s+e)/2$ (floor)

$= \boxed{2}$

if (arr[s] >= arr[mid])

3 >= 5 → F

(I) compare with mid element

$5 < 1$ ✗

(II) compare with prev elmt

$5 \leq 4$ ✗

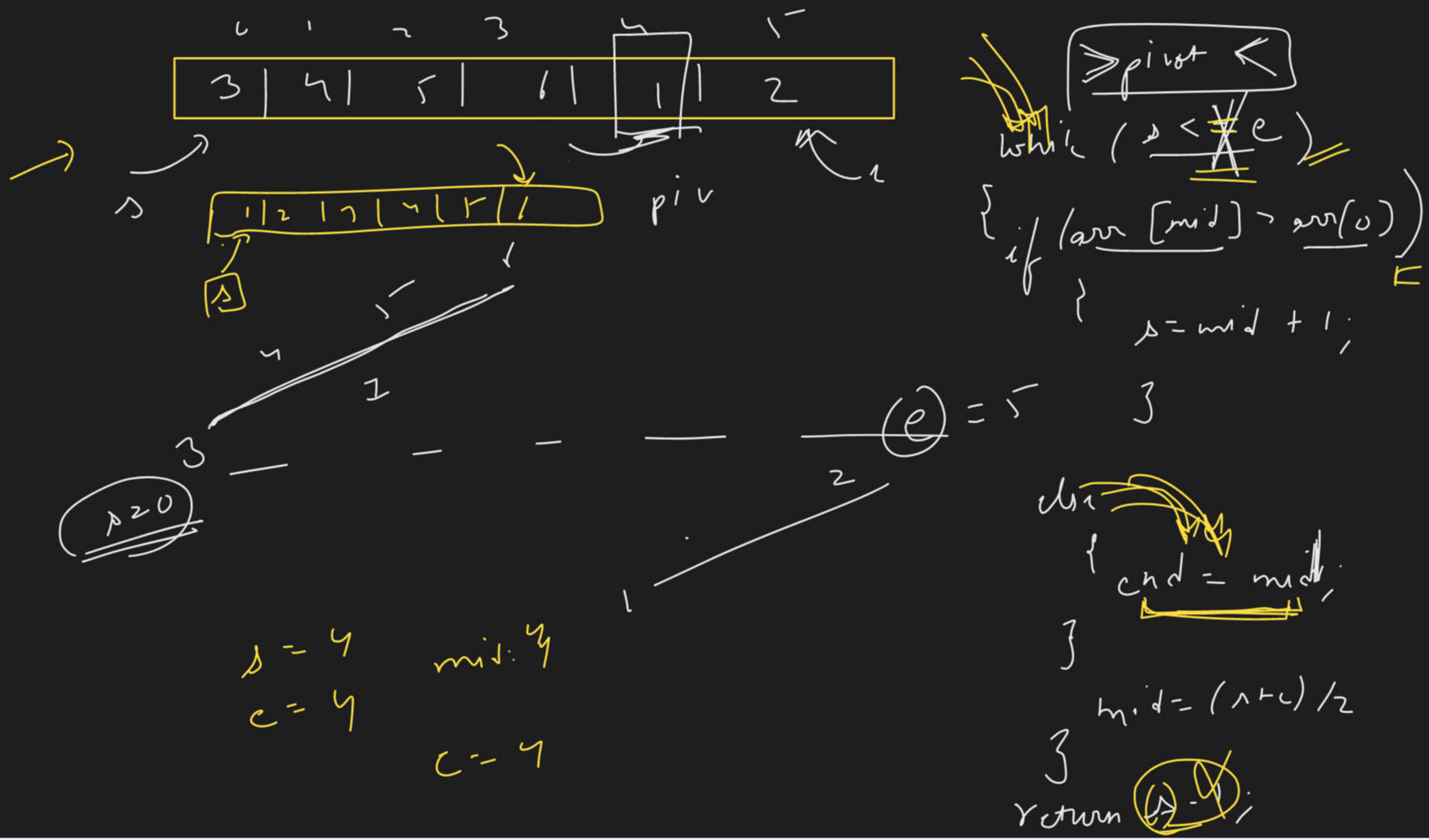
same approach

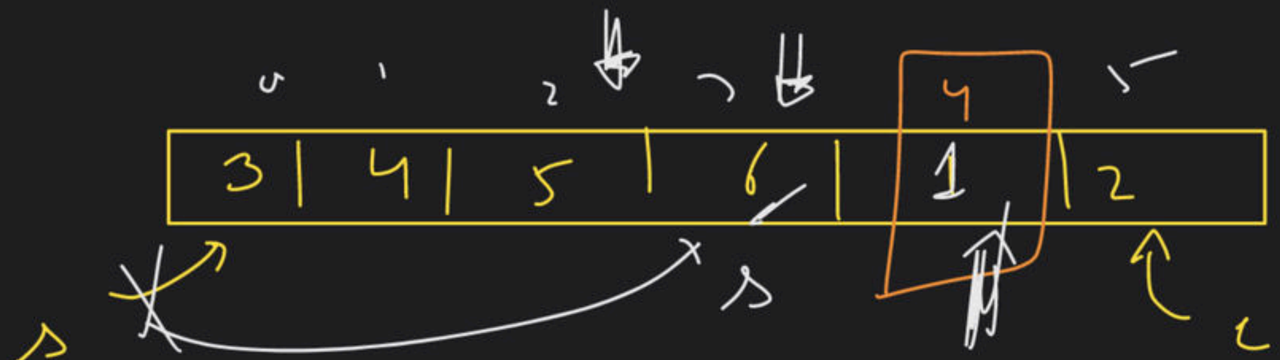
ans Yes or No

Dry Run

same

YT -> [L


3 4 5 6 1 2
piv
s
e
mid
pivot
while (s < e)
if (arr[mid] > arr[0])
s = mid + 1;
else
end = mid;
mid = (s+e)/2
return s;
s = 4
e = 4
mid: 4
e = 4

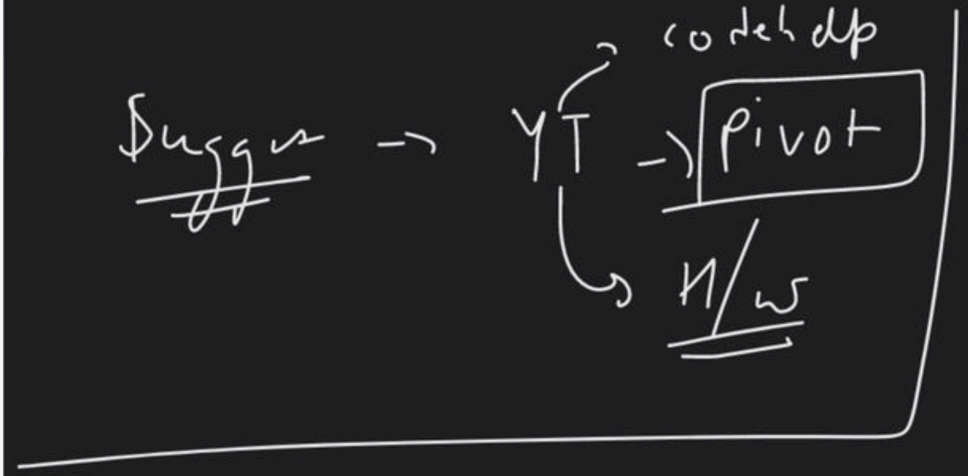


s = 0
e = 5
mid = 2

s = 3
e = 5
mid = 4

e = mid
e = 4

s = 3, e = 4
mid = 3
s = 4, e = 4
return 4



Peak in a mountain array

→ 30 min

→

SQRT - with BS (YT → codeforces)

TRUE

FALSE

√2 → 1.414

B.S

1

Loop → 1.1, 1.2, 1.3, 1.4

→ 1.41, 1.42

→ 1.411, 1.412, 1.413, 1.414

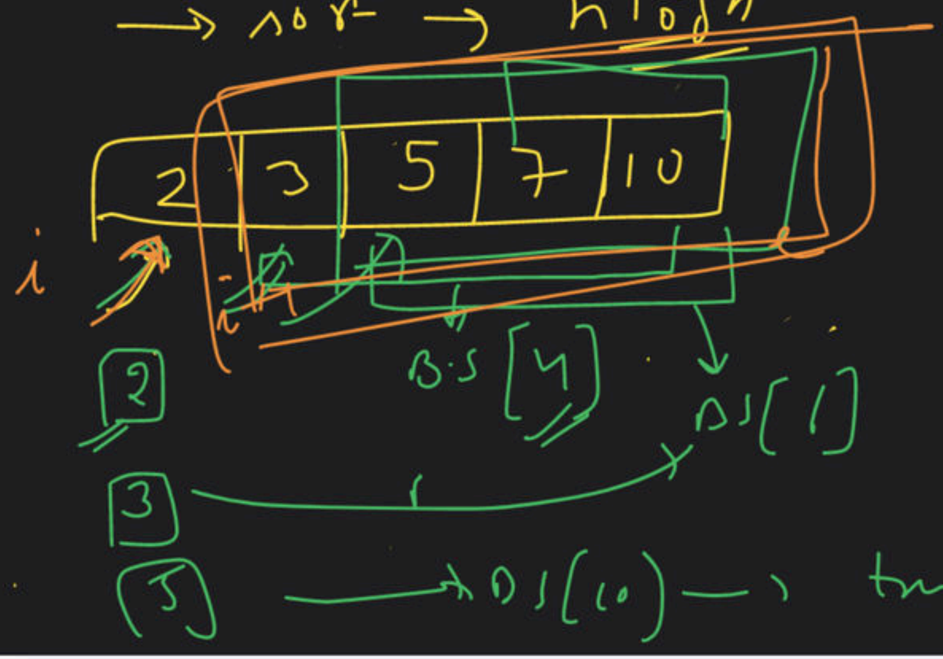
→ Check if N & its Double Exist

→ # Brute force -> sort → $O(n^2)$

→ # Map → $O(n)$

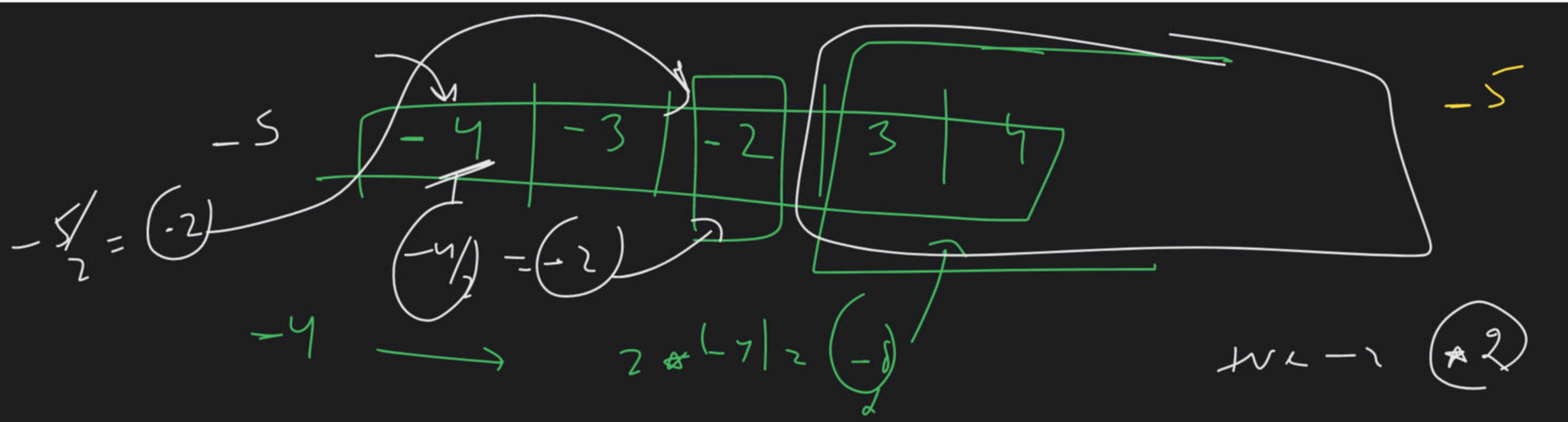
→ # B.S -> $n \log n$

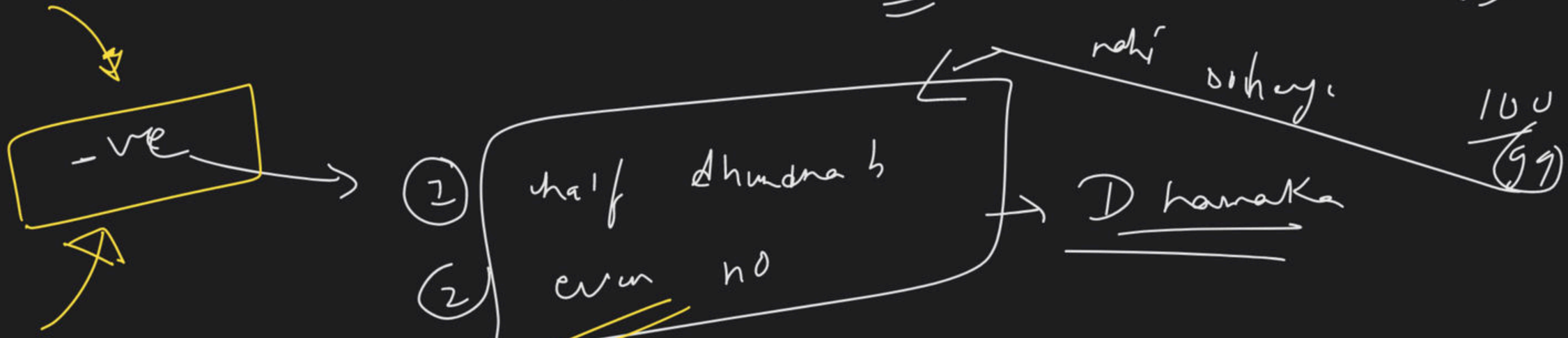
→ sort → $n \log n$



→ N, 2*N arr[i] = N
arr[j] = 2*N

→ i != j


-5
-4
-3
-2
3
4
-5
-5/2 = (-2)
-4/2 = (-2)
-4 → 2*(-4)/2 = (-4)
+ve → *2
-ve → 1/2
-2 → 2*(-2) = -4

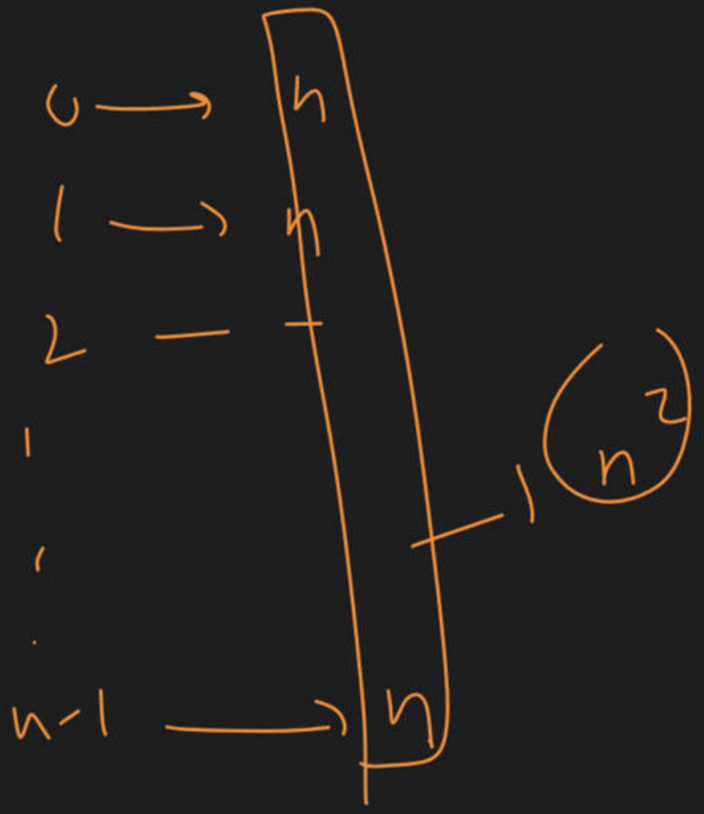

-ve
1
2
half dhundna h
even ho
Dharmatma
nahi milegi
100
99

14T.C

→ 10T.C

4T.C

→ -n



for(0 — n)

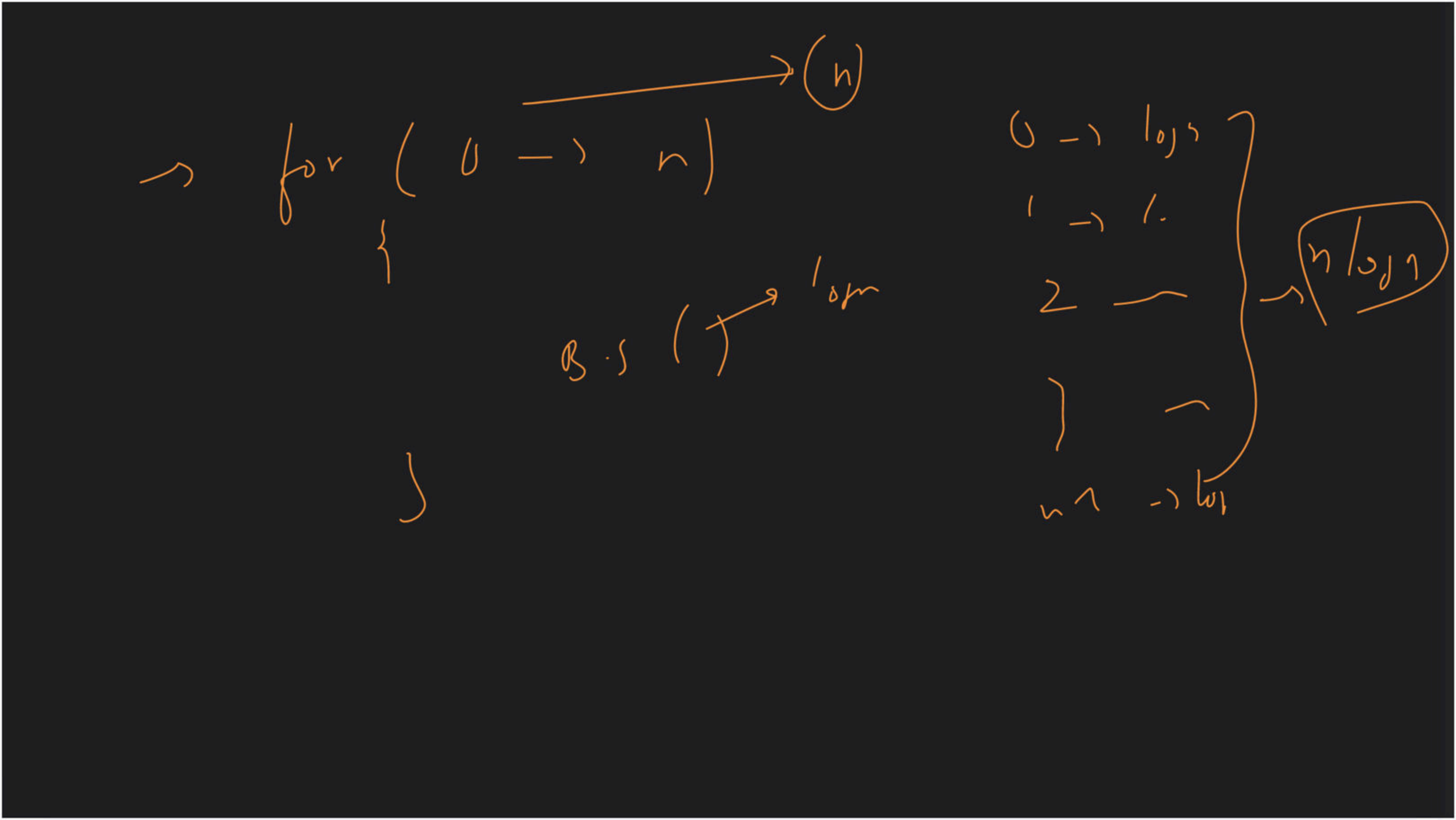
n

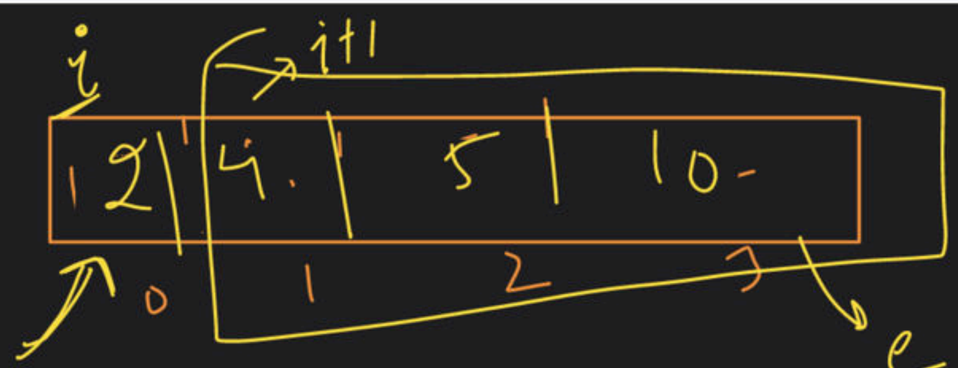
for(0 — n)

{

}

}


for (0 -> n)
{
}
n



n2 arr.size()

B.S

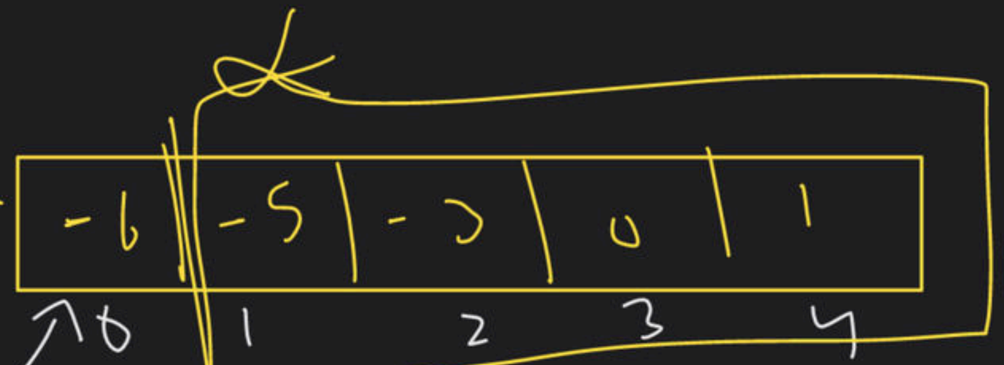
↳ OUT

for 2 to be an answer, 4 has to be present

arr[i] = 2

B.S (i+1, e, 4) → TRUE → Time

↳ false → 0



Even

flex

↓

B.S

(I) → sort

(II) for "-6" to be an answer,

$2 * (-1) = -12$ has to be present

divide →

$\frac{(-6)}{2} = -3$

$-6/2 = -3$

(-3, -3) → Yes

$-7/2 = -3$

(-7, -3) → No

-> HashMap ⟶ Hw -> STL -> Vector
Sort
Map
Hash
Queue
} -> Hw

```
map<int, int> m;
for (int i -> 0 -> <n)
{
    int element = arr[i];
    m[element]++;
}
```

loop

element

[0]

0 % 2 = 0

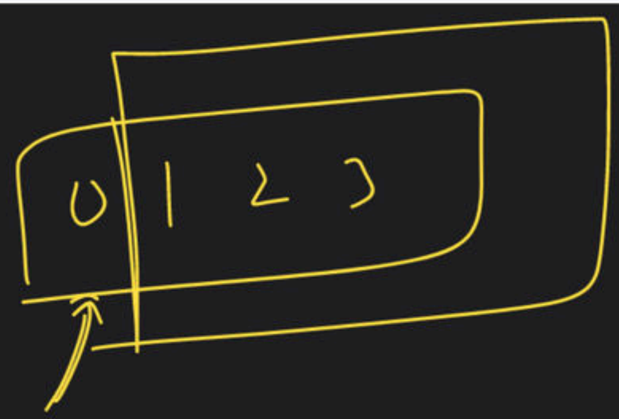
0 / 2 = 0

↳ T/F -> false

→ Google :-

No Index Out of Bound exception

Existed or not



0 1 2

arr[3]

infinite size array

[0 0] → TRUE

true no → 2

i/p arr

fun(arr[], size)

size of

s

e

→ target → 30 → Present or Not

B.S →

```
s = 0
e = 1
int val = arr[0] = 3
          why ?
while (val < target)
{
    s = e
    e = 2 * e
    val = arr[e]
}
```

arr(1)

n

n/2

n/4

n/8

Search space

range

arr[s] ≤ target ≤ arr(e)

target = 30 ← i/p

arr

val = 7 13 24 30

3	7	13	18	24	30	34	37	40	...
0	1	2	3	4	5	6	7		

s

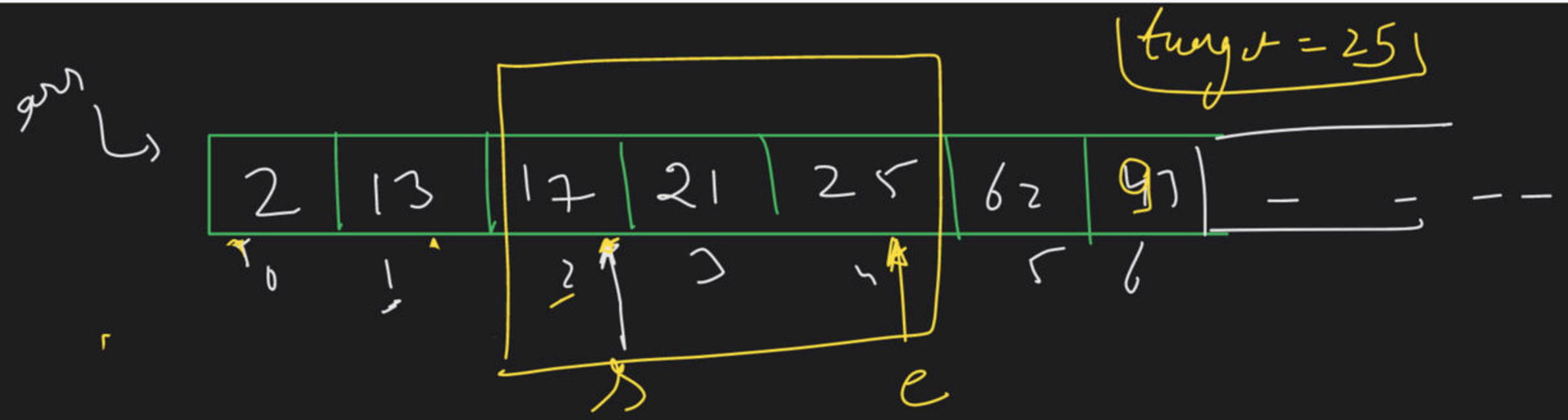
e

7 < 30

13 < 30

24 < 30

40 < 30 → F



s = 0, e = 1

val = arr[0] = 2

val < target

2 < 25 → T → new range find karo

s = 1

e = 2 - 1

s = 1, e = 2

val = 17

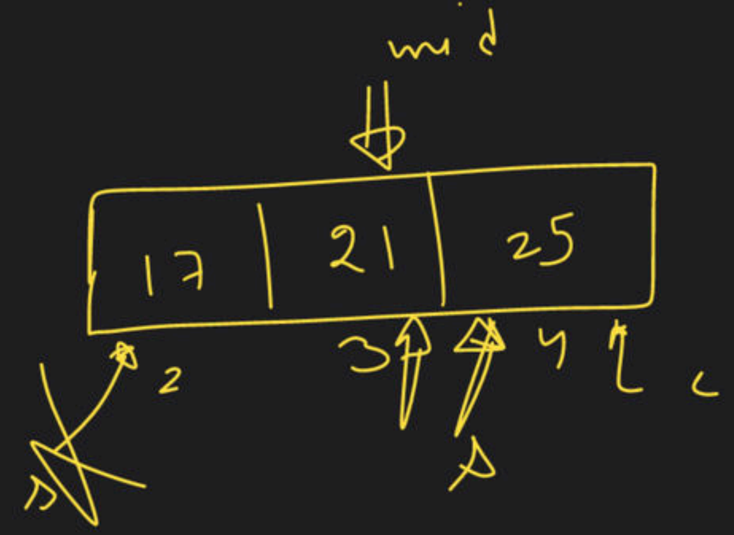
17 < 25 → T → new range

s = 2, e = 4

val = arr[e] = 25

25 < 25 → False

B.S in range me



s = 4, e = 7

mid = 7

25 == 25

T

return 4

s = 2, e = 4

mid = (2+4)/2 = 6/2 = 3

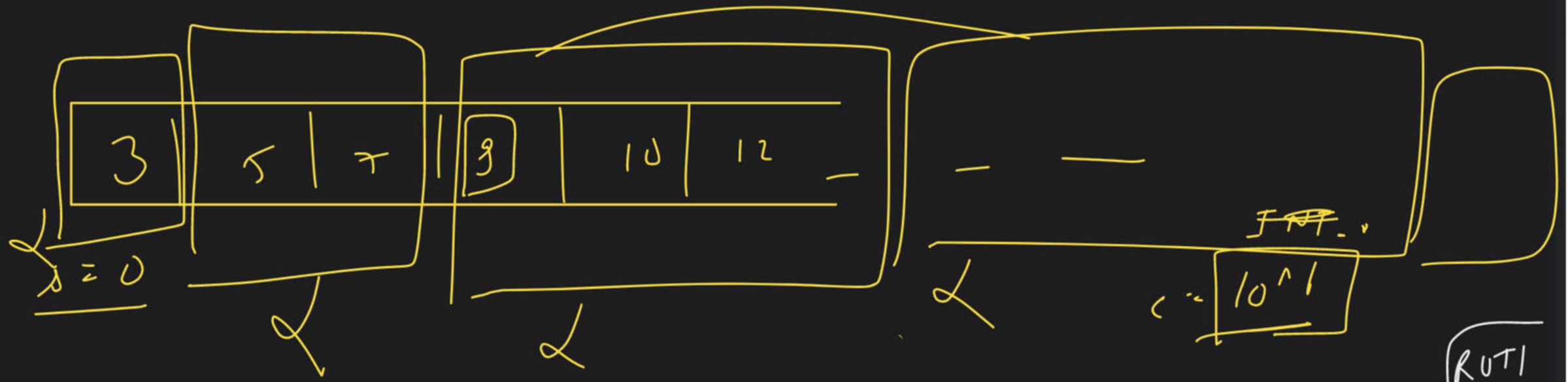
21 == target = 25 → false

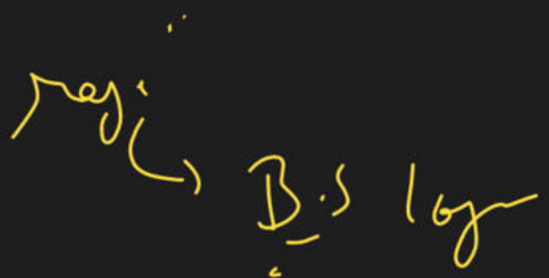
arr[mid] < target

21 < 25 → True → Right

s = mid + 1 = 3 + 1

s = 4





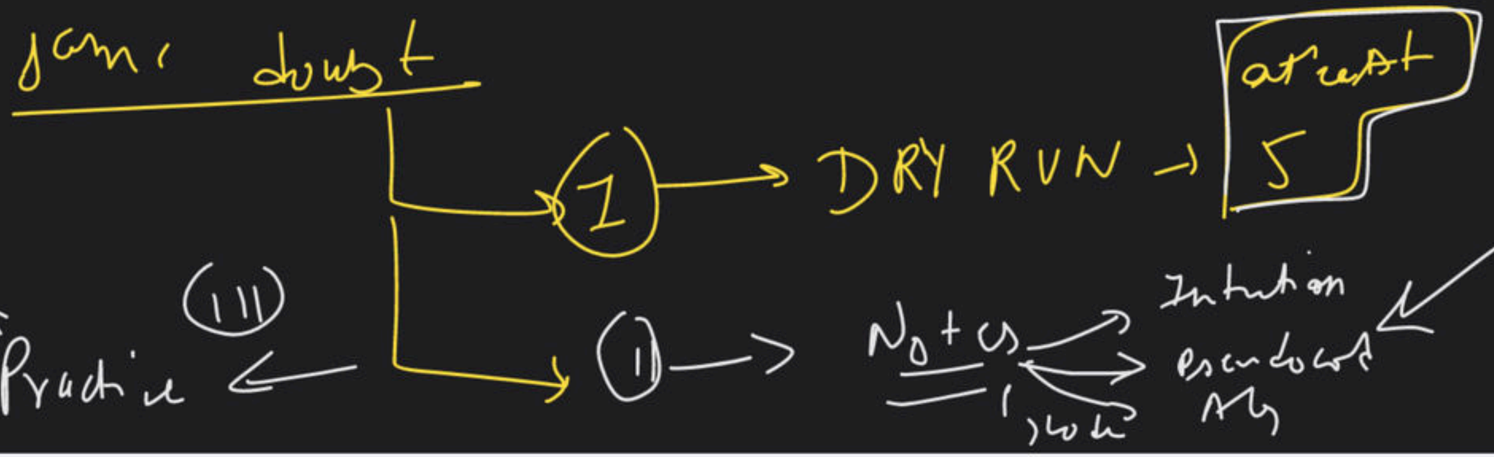
B.S

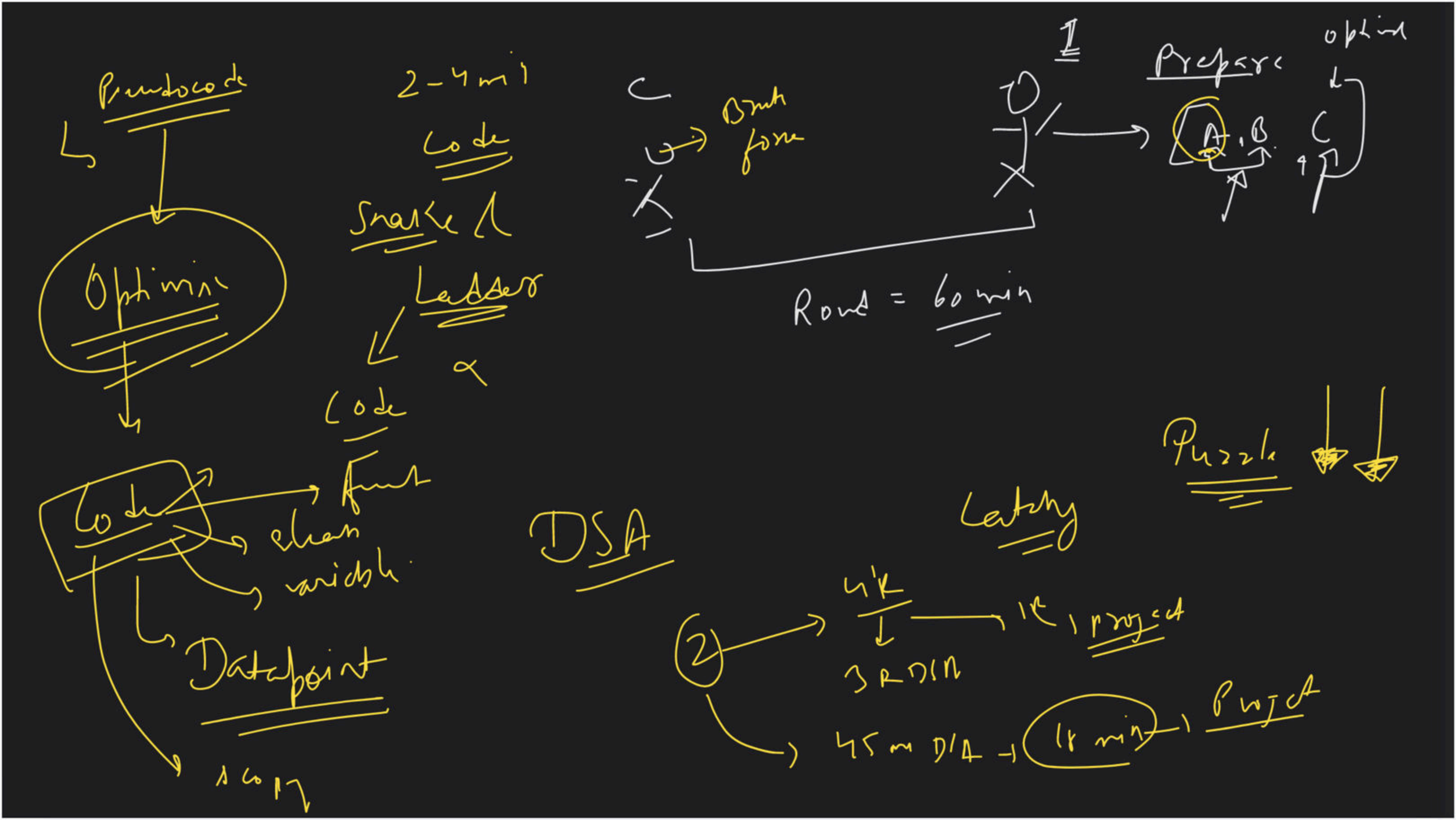
ROTI

2 hr

Sure

ROTI
ENG
BA
EP
AL




Pseudocode
2-4 min
Code
Optimise
Snake & Ladder
Code
Code
fast
clean
variable
Datapoint
1 Code
C
Brute force
1
Prepare
optimal
Round = 60 min
DSA
Puzzle
Lately
2
4'K
1K | project
3 RDIN
45 min DIA
15 min
Project

Google

↳ infinite sorted array

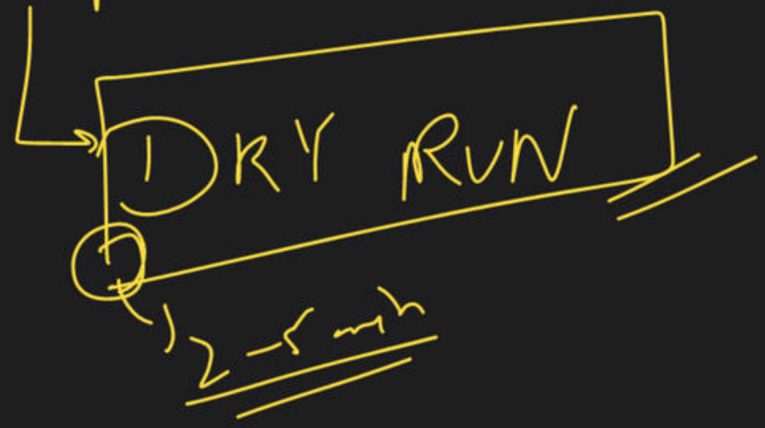
↳ target

↳ T.C → ?

Right

log n

Optimise



n -> position of target

O(n) -> ? Wrong

O(n log n) -> ? Wrong

O(n²) -> Wrong

n -> ? no. of elements

Part I :- range find Out

Part II → B.S on range → n

2 * log p → O(log p)

→ log p → How ? Why ?

B.S → log n → ? How Why

n log n → dry run

→ Google → 5K+

∞ (Unknown size)

array sorted

target -> ?

1.5hr → 2hr

Party = Code

10K+

15K+

T.C -> ?

Bubble sort
Insertion sort → padhke
Selection sort → CodeHelp → YT

HW

30 min
30 min

ignore

STL → 1 video