Q: arr = [2, 3, 5, 9, 14, 16, 18], target = 14

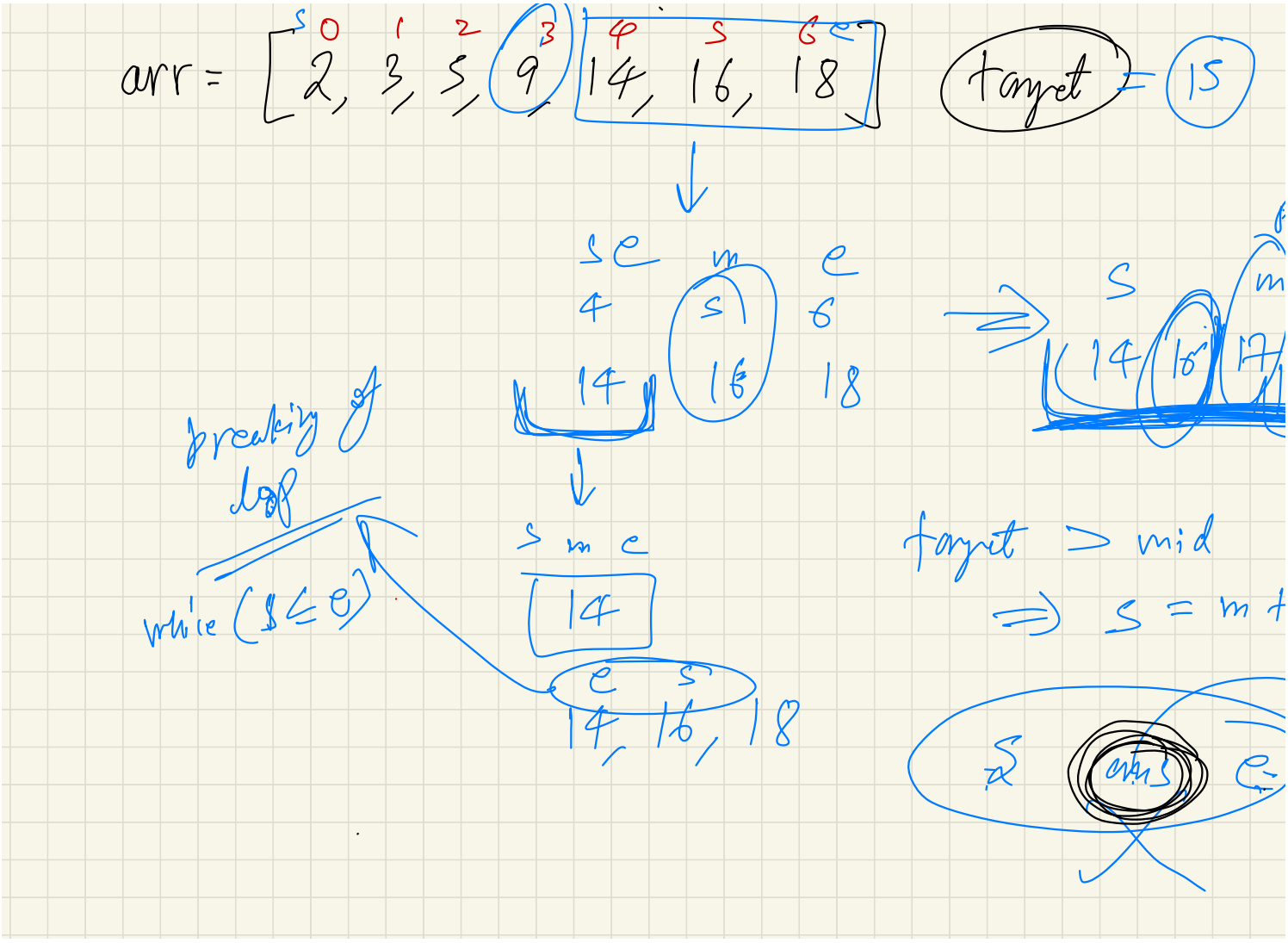
Ceiling = smallest element in array greater or = target.

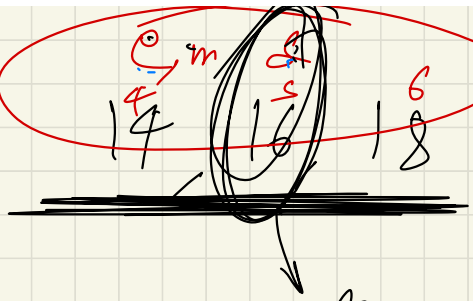
ceiling (arr, target = 14) = 14

ceiling (arr, target = 15) = 16

target = 4 = 5

target = 9 = 9


arr = [2, 3, 5, 9, 14, 16, 18]
0 1 2 3 4 5 6
target = 15
s e
4 14
m
s
16
e
6
18
breaking of loop
while (s <= e)
s m e
14
e s
14, 16, 18
target > mid
s = m
s
e



let's talk about start & end poi

Q: Find a n
≥ targe

& target e

⇓

→ e target s // condition for while loop w

start ≤ end ⇒ when while loop breaks

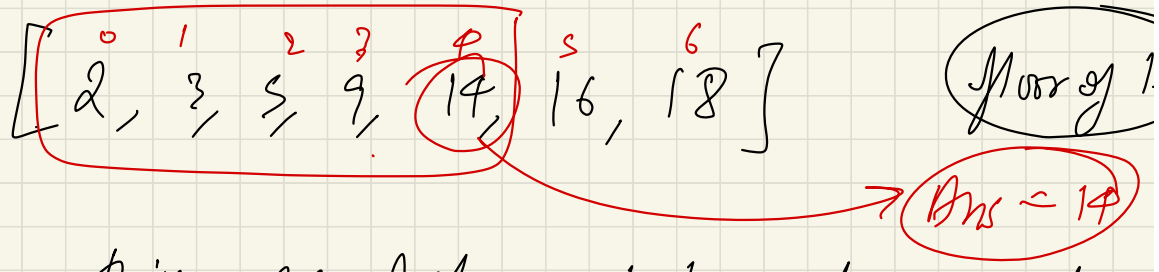
start = end + 1

next big no, when no ans found = start element. reas

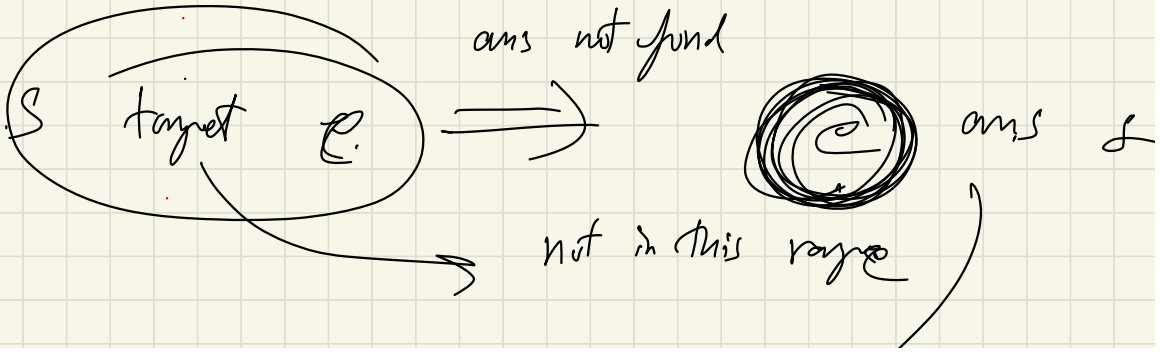
Q: Find the floor of a number.

floor = greatest number smaller or = target

arr = [2, 3, 5, 9, 14, 16, 18]



// same thing as before, just return end instead of -1, if no. not found.



When ~~condition~~ is violated,

$d = c + 1$ $\quad s > c$

3: ① Exact same approach for veiling of no.

② Ignore the target = what we are looking for

③ arr = ['c', 'd', 'y', 'j'] target = 'c', s

condition violated : start = end + 1 ⟹ length
= 1

return S % N

2 % 4 = 2 // re

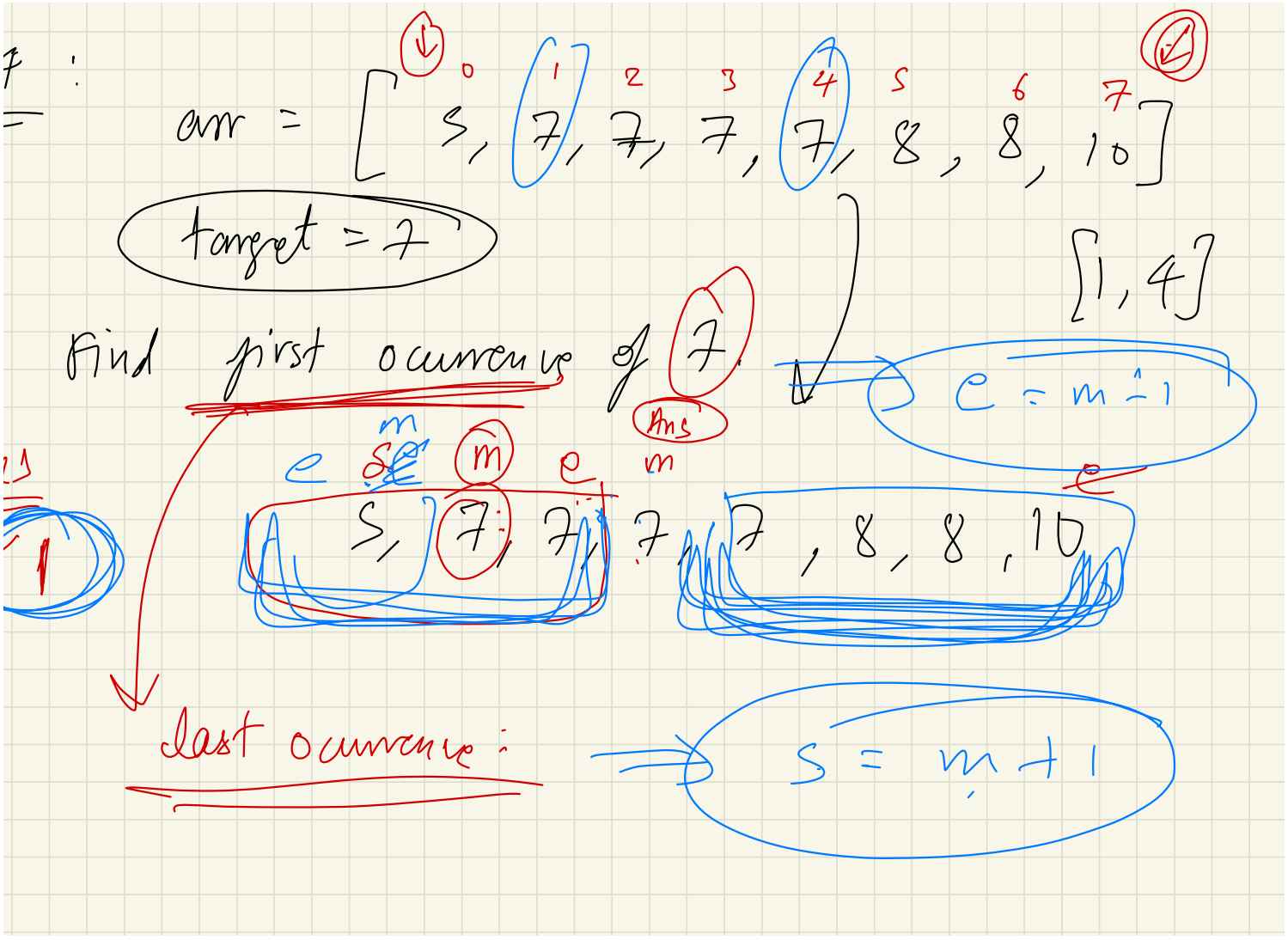
4 % 4 = 0

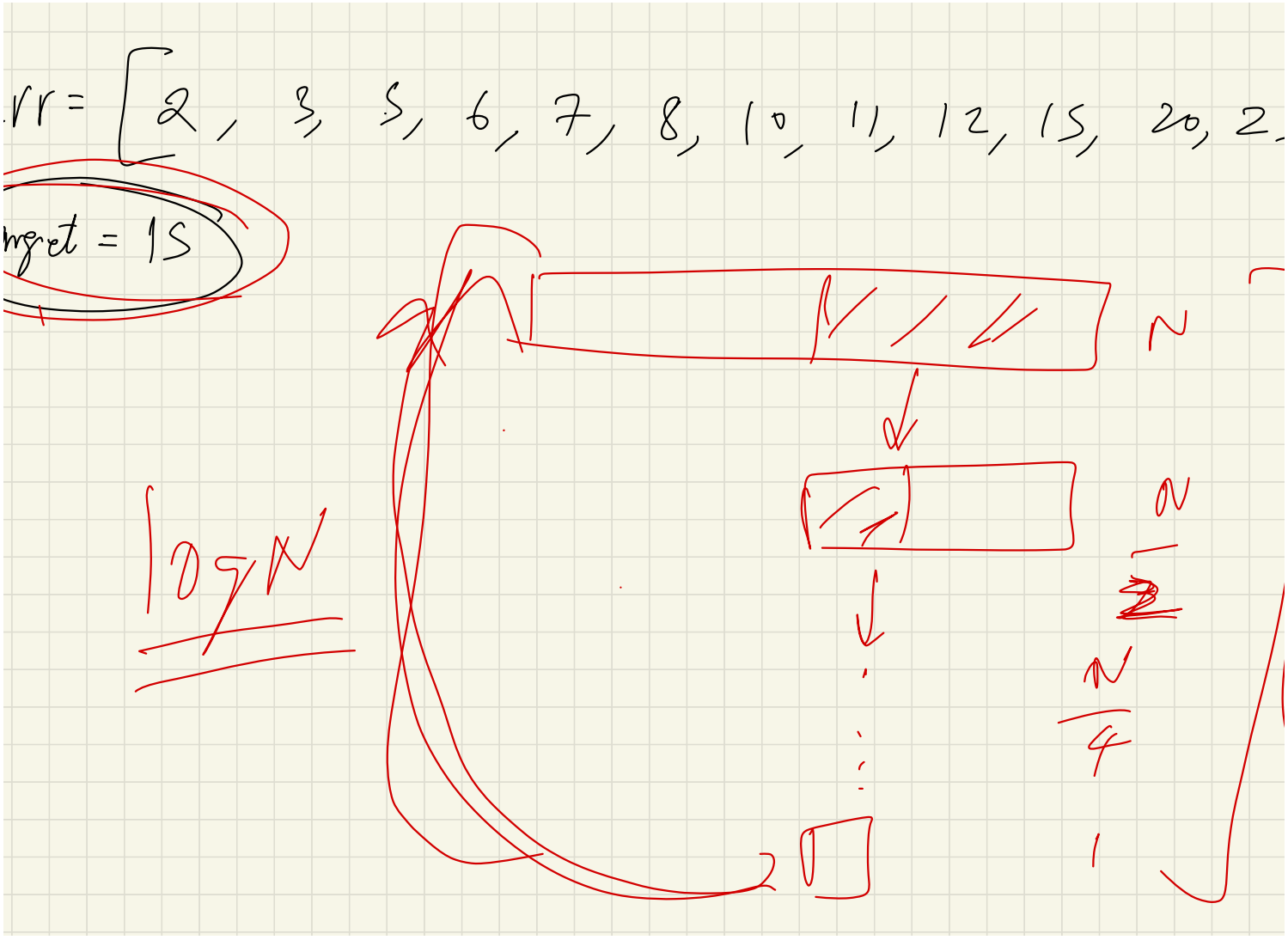
st

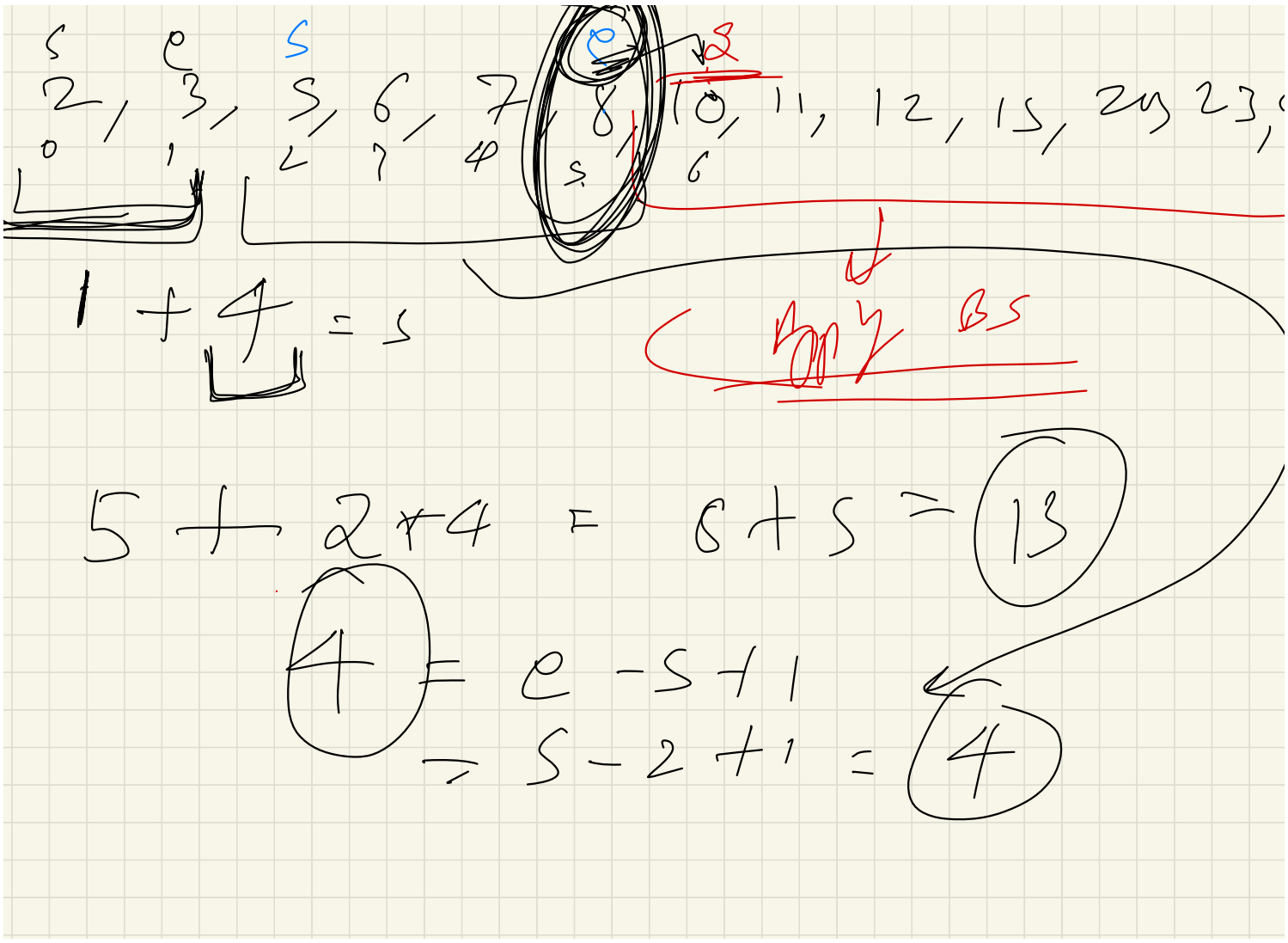
=

if S == N :

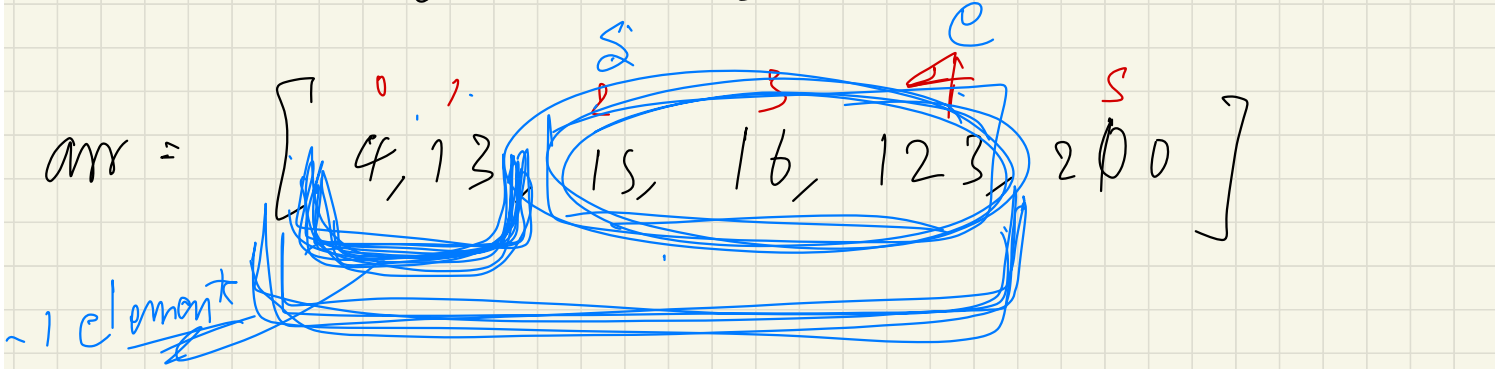
return 0


arr = [5, 7, 7, 7, 7, 8, 8, 10]
0 1 2 3 4 5 6 7
target = 7
[1, 4]
Find first ocurrence of 7
ans
e = m - 1
s
m
e
5, 7, 7, 7, 7, 8, 8, 10
last ocurrence :
s = m + 1


rr = [2, 3, 5, 6, 7, 8, 10, 11, 12, 15, 20, 2
rget = 15
N
N/2
N/4
1
log N


S
e
S
e
2, 3, 5, 6, 7, 8, 10, 11, 12, 15, 24, 23,
0 1 2 3 4 5 6
1 + 4 = 5
5 + 2 * 4 = 6 + 5 = 13
4 = e - s + 1
= 5 - 2 + 1 = 4

Finding size of array by indices:

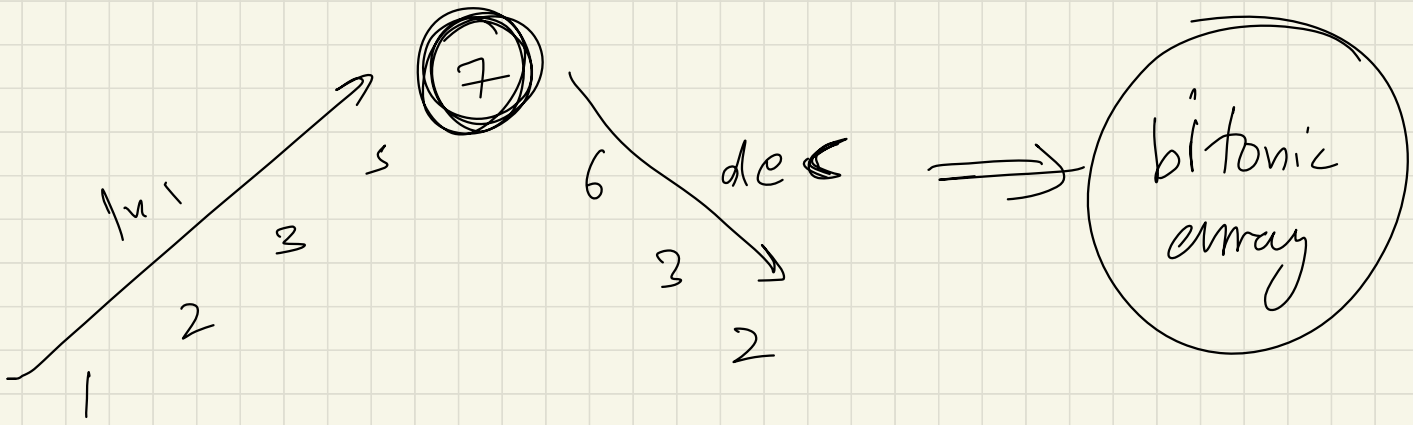


$$e - (s-1) = (e - s + 1)$$

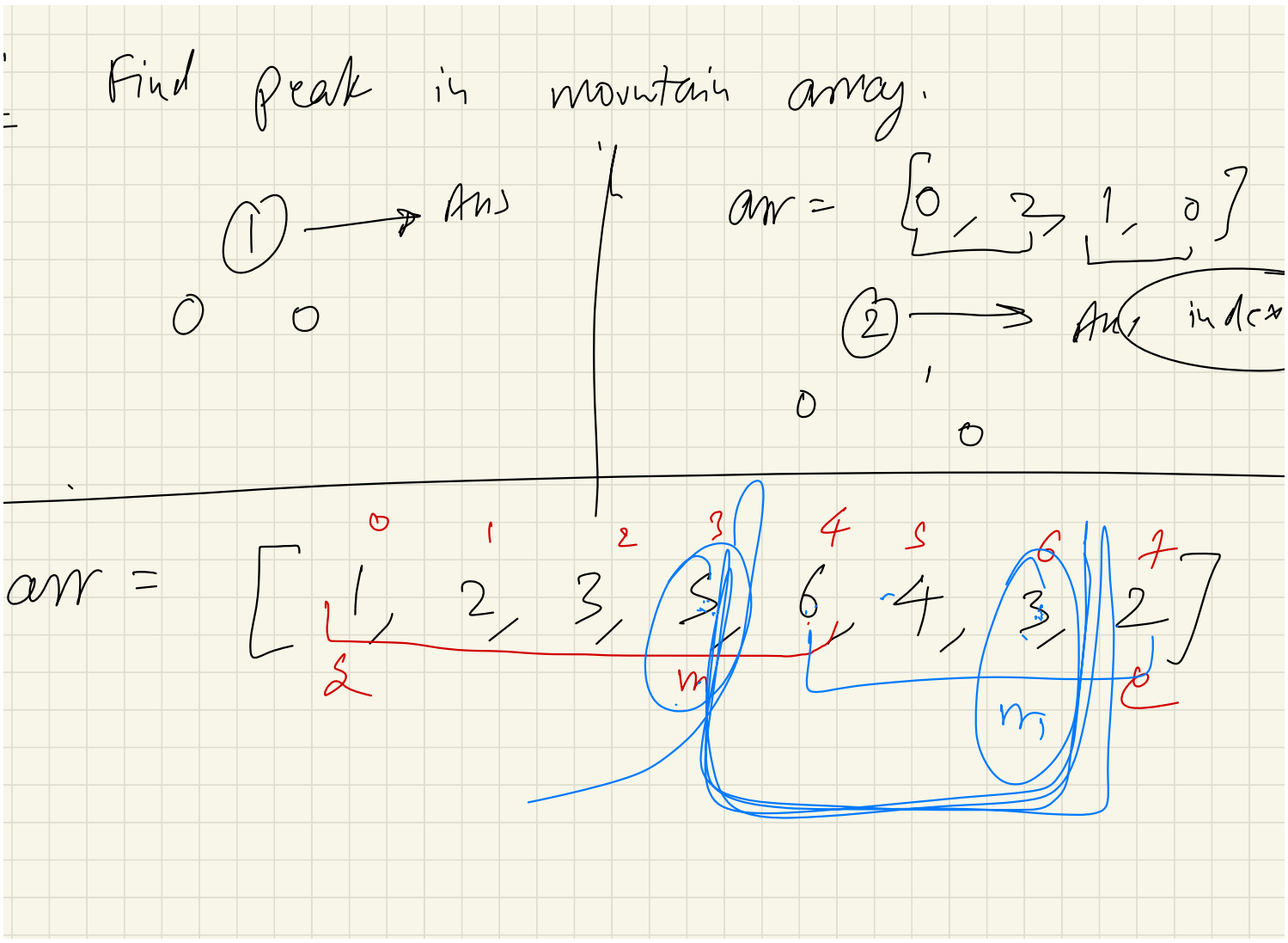
$4 - 2 + 1 = 3$

Ans

arr = [1, 2, 3, 5, 7, 6, 3, 2]



Find peak in mountain array.



1) if a[mid] > a[mid+1] ⟹ You are in the dec p of arra

⟹ e = mid

// check in lh

s [m] e

e

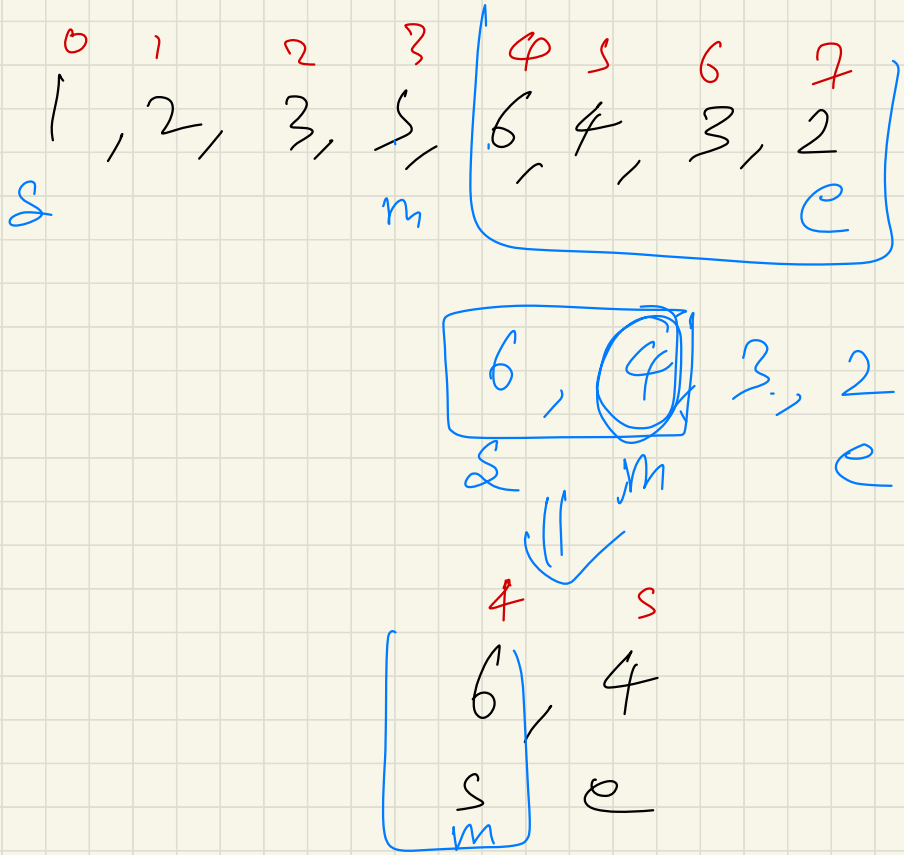
2) a[mid] < a[mid+1] ⟹ You are in asc part of

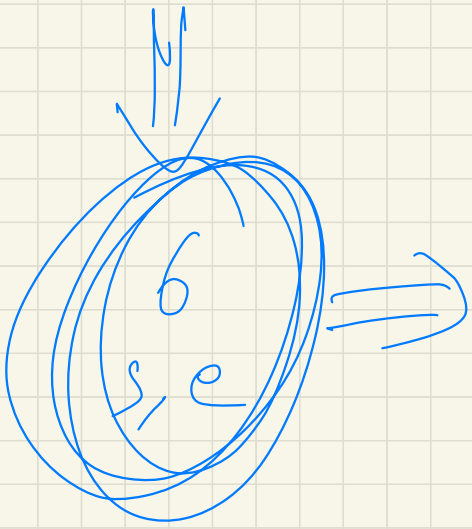
⟹ s = mid+1

⟹ s m max

e

3) When will loop break ?





In the end,
sk e both
will get to
the large
no.

Ans

why? ① &

:

arr = [1, 2, 3, 4, 5, 3, 1]

0 1 2 3 4 5 6

target = 3

ns :

(1) Find peak element ⇒ 4 index

(2) Binary Search in asc array ⇒ (0, 4)

(3) If not found, binary search in dec array ⇒ [4, 6]

Rotated Binary Search:

arr = [2, 4, 5, 7, 8, 9, 10, 12]

After 1 rotation:

= [12, 2, 4, 5, 7, 8, 9, 10]

1 rotat

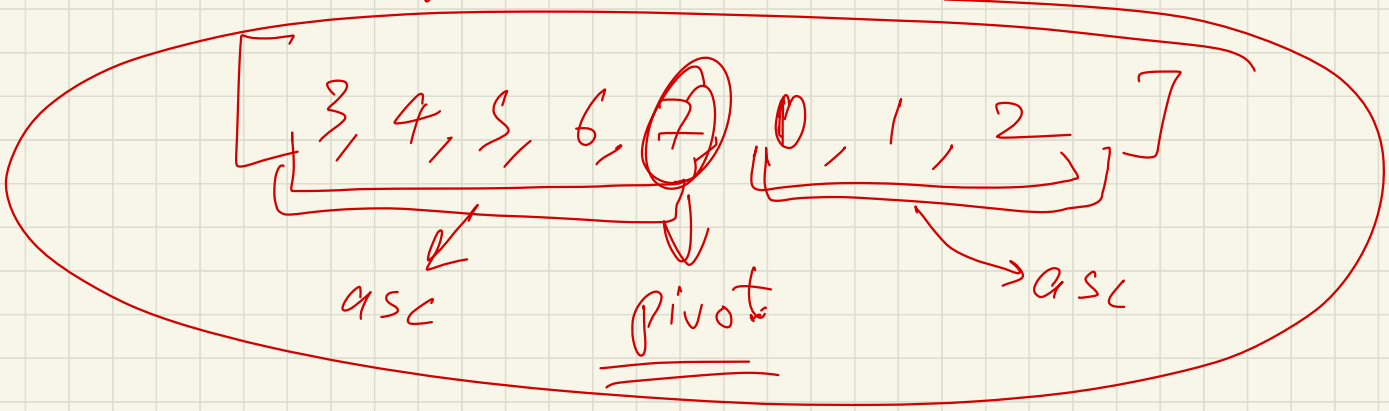
2nd rotati

= [10, 12, 2, 4, 5, 7, 8, 9]

pivot

⇒ Find the pivot in the array.

pivot ⇒ you where your next nos are

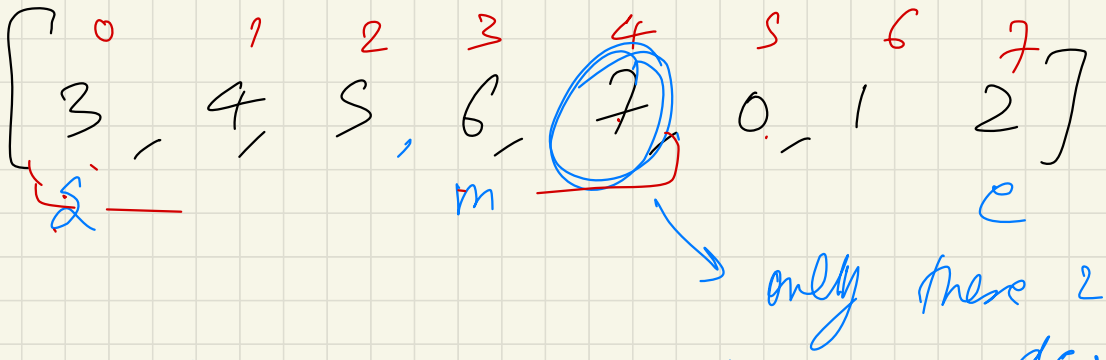


* Find pivot

* Search in first half ⇒ Simple B.S. (0, pivot)

otherwise, search in second half : (pt+1, e

: Find pivot :

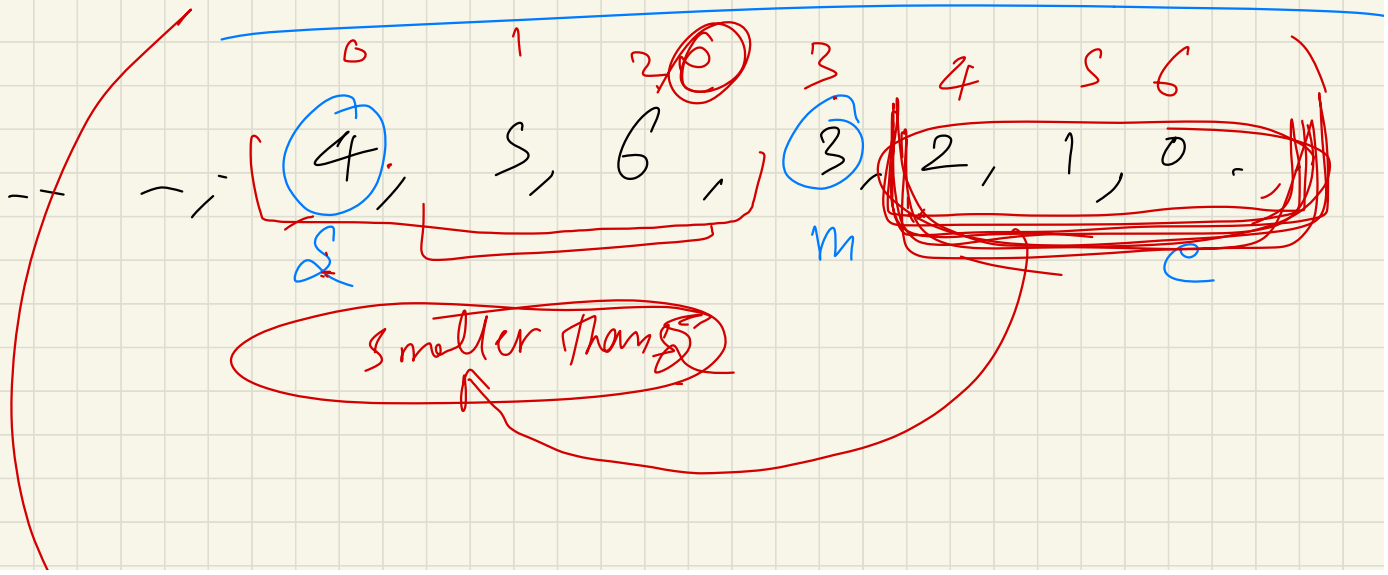


when? → when you find that mid > mid+1 element, i.e. pivot. // Done!

2/ if mid element < (mid-1) element

i.e also my ans ⟹ ans = m-1

case 3 .. start element ≥ mid element

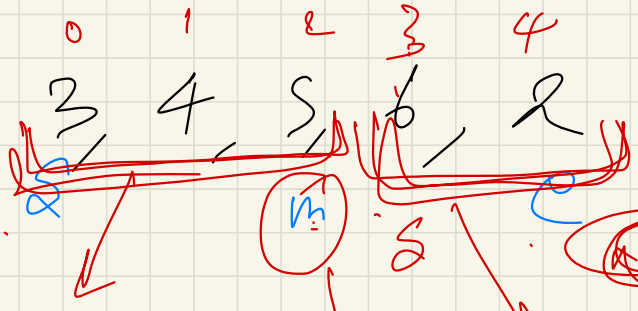


→ In this case, all elements from m will be < start.

Hence, we can ignore all these elements since we are looking for peak i.e largest element.

e = mid - 1

ase 4 :- & element < mid elem



sorted

in p in

2, 3, 4, 5, 6

s = m + 1

If this way pivot, it would've been returned in case 2

Hence proved, that bigger no. lie ahe

Hence, ignore mid & put s = mid

To BS using pivot.

arr = [4, 5, 6, 7, 0, 1, 2]

s p e

se 1 : pivot element = target // Bus

se 2 : target > start element // Sear
for

search space = (s, p-1)

why ? coz all nos. after pivot
are < s

se 3 : target < start element

i.e. we know that all elements
from s, pivot are not going to

bigger than target. // target = 1

Search space = (pivot + 1 till e

---

arr = [2, 2, 2, 2, 9]

⇓ rotate twice

[2, 9, 2, 2, 2]

0 1 2 3 4

s m e

arr = [4, 5, 6, 7, 0, 1, 2]

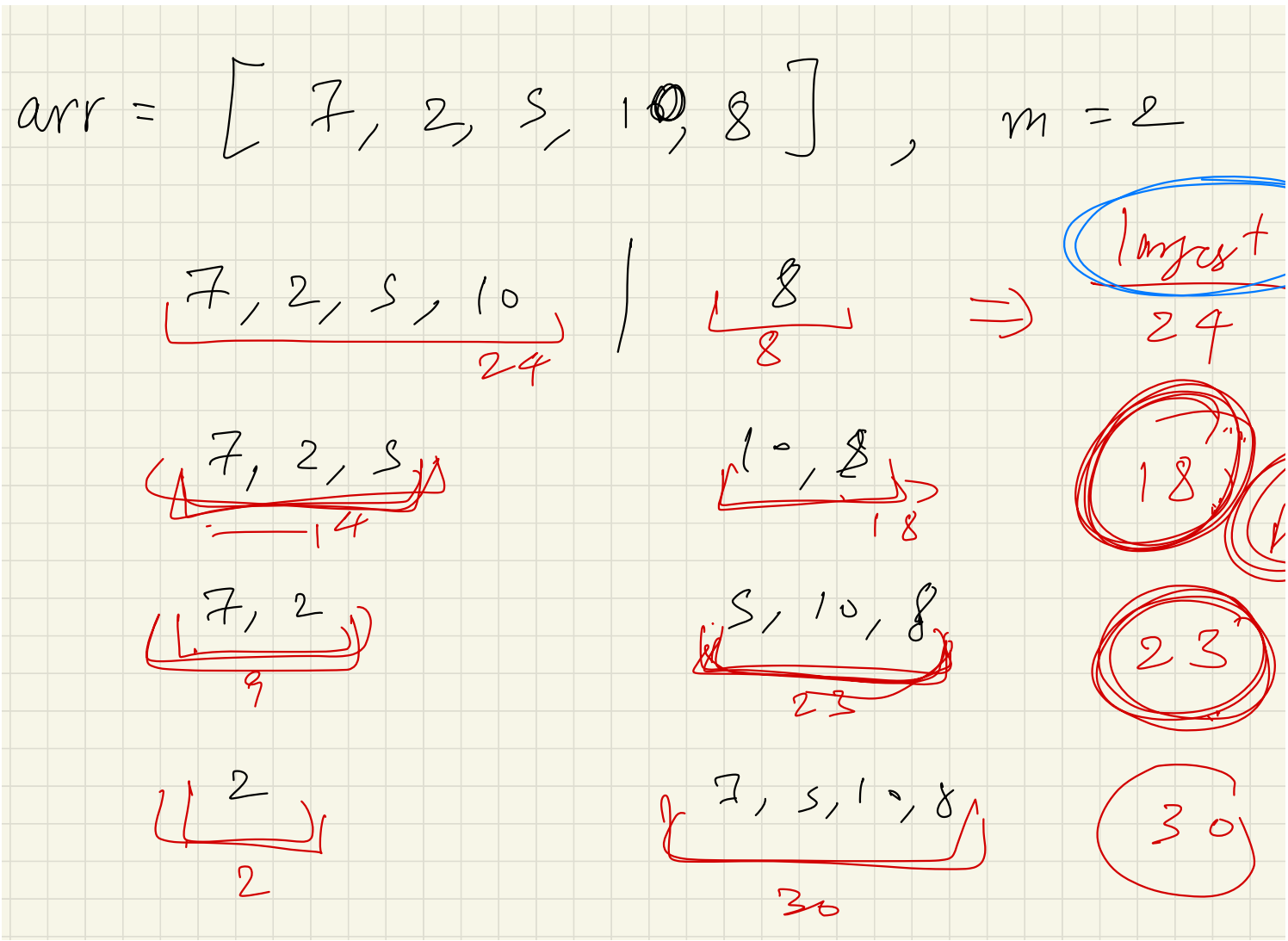
Rotated 4 times from original

0 1 2 3 4 5 6 7

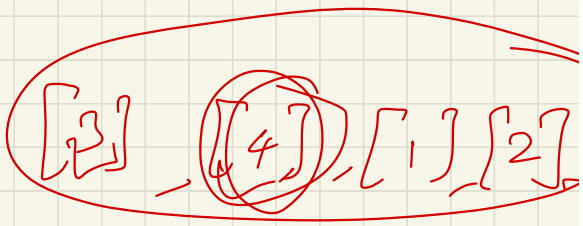
4, 5, 6, 7 | 0, 1, 2, 3

pivot = 3

Ans = 0


arr = [7, 2, 5, 10, 8] , m = 2
7, 2, 5, 10
24
8
8
largest
24
7, 2, 5
14
10, 8
18
18
7, 2
9
5, 10, 8
23
23
2
2
7, 5, 10, 8
30
30

=> (1) min no. of partitions that we can make
(2) what is the max no. of partitions / m that can be = N

arr = [3, 4, 1, 2] ⟹ 

next will be the ans in case (1) :

[7, 2, 5, 10, 8]

Sum of entire array = 32

Case (2) :

Ans for this = max element in array = 4

ax value of ans of question = case 1

in value of ans of question = case 2

---

minAns = max value in array

maxAns = Sum of all values in array.

[10, 32]

Here it di
that we
BS.

start = 10

end = 32

$$mid = \frac{st + e}{2} = \frac{42}{2} = 21$$

Try to see if you can split the array with 21 as the max sum:

7, 2, 5, 8, 10

[7, 2, 5], [8, 10]

pieces

2

check if (pieces ≤ m) ⇒ end = mi

s = 10 , end = 21

mid = 15

7, 2, 5, 8, 10

[7, 2, 5] , [8] , [10] pieces

if pieces > m

⇒ start = mid + 1

$s = 16$, $e = 21$

$7, 2, 5, 8, 10$

$m = 18$

$[7, 2, 5]$, $[8, 10]$ pieces = 2

$\Rightarrow$ $s = 11$ $e = 18$

$m = 17$

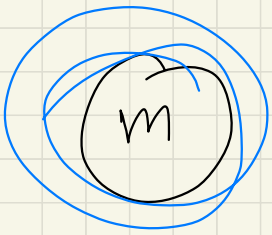
$7, 2, 5, 8$

$[7, 2, 5]$, $[8]$, $[17]$ pieces =

$8 = m+1 = 18$

$S = 18, \quad e = 18$

$m = 18$ Ans



// The arr exists definitely, hence by the above 2 checks we will reach :)