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

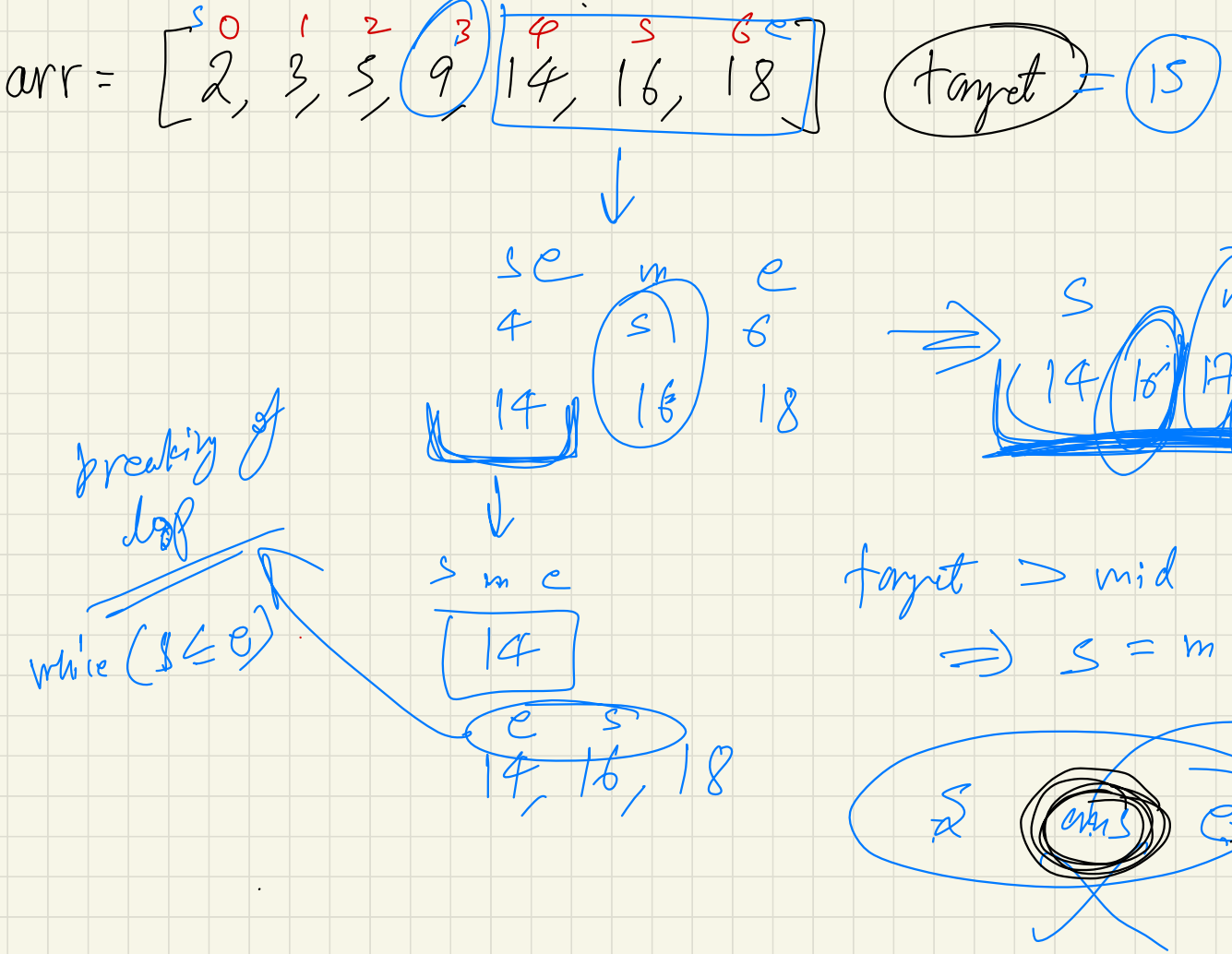
Ceiling = smallest element in array greater or = target.

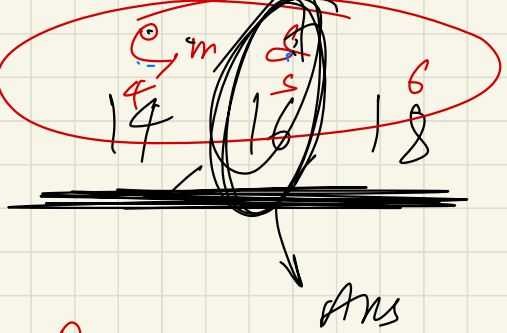
Ceiling (arr, target = 14) = 14

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 6
14
m
s
16
e
18
possible
s m e
14 16 17 18
breaking of loop
while (s <= e)
s m e
14
e s
14, 16, 18
target > mid
⇒ s = m + 1
s
mid
e
arr



let's talk about start & end pointer.

Q: Find a num ≥ target

s e target

⇓

e target s // condition for while loop violated

start ≤ end ⟹ when while loop breaks

start = end + 1

next big no, when no one found = start element. reason?

Q: Find the floor of a number.

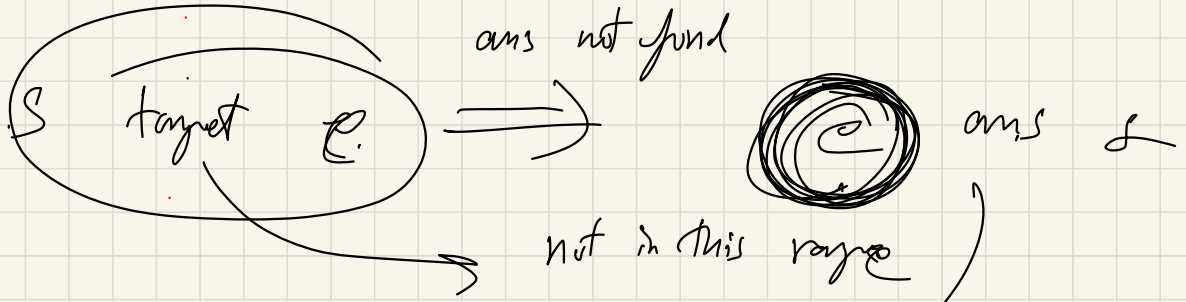
floor = greatest number smaller or = target

arr = [2, 3, 5, 9, 14, 16, 18] (0 1 2 3 4 5 6)

floor is 14

→ Ans = 14

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



When ~~condition~~ it violated,

$s = e + 1$ $\quad s > e$.

Q3 : (1) Exact same approach for veiling of no.

(2) Ignore the target = what we are looking for more

(3) arr = ['c', 'd', 'g', 'j'] target = 'j'

condition violated : start = end + 1 ⟹ length of array = N

return S % N

2 % 4 = 2 // ret stmt

4 % 4 = 0

if S == N :

return 0

Q4 :

arr = [5, 7, 7, 7, 7, 8, 8, 10]

0 1 2 3 4 5 6 7

target = 7

[1, 4]

* Find first occurence of 7.

Ans

1

s m e

5, 7, 7, 7, 7, 8, 8, 10

Ans

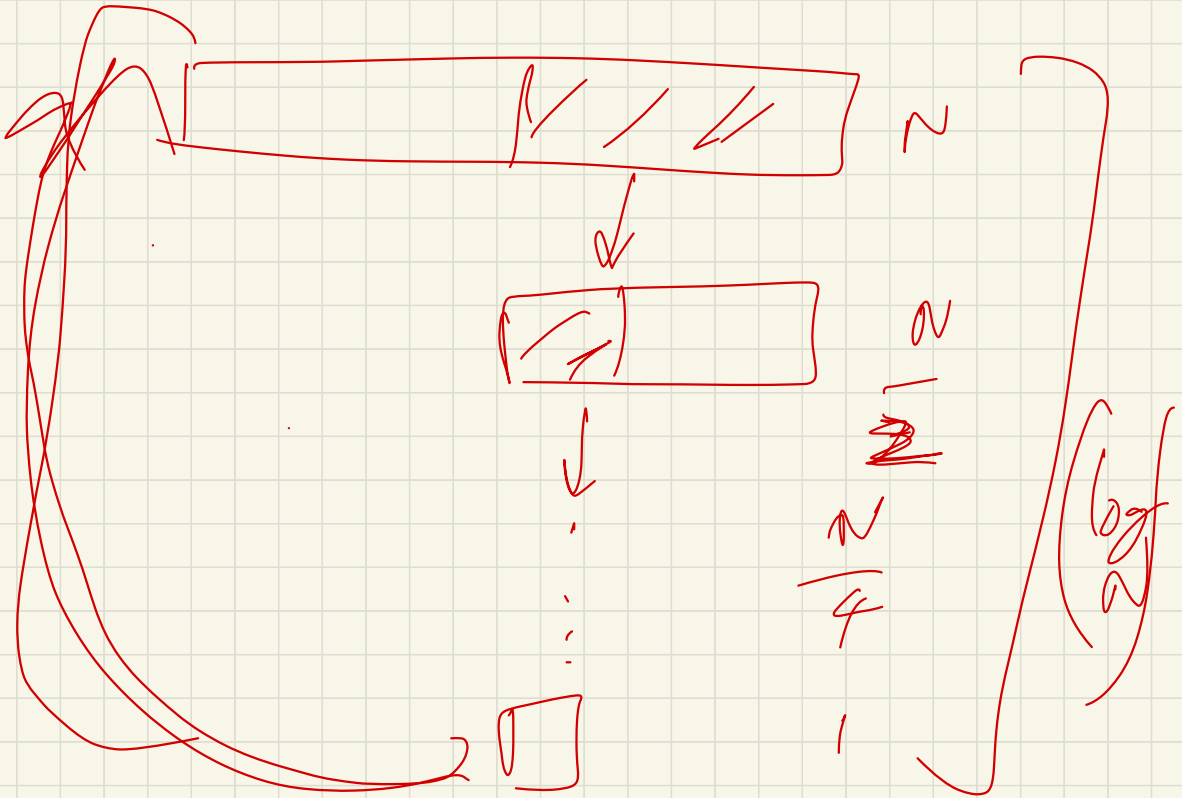
e = m - 1

last occurence :

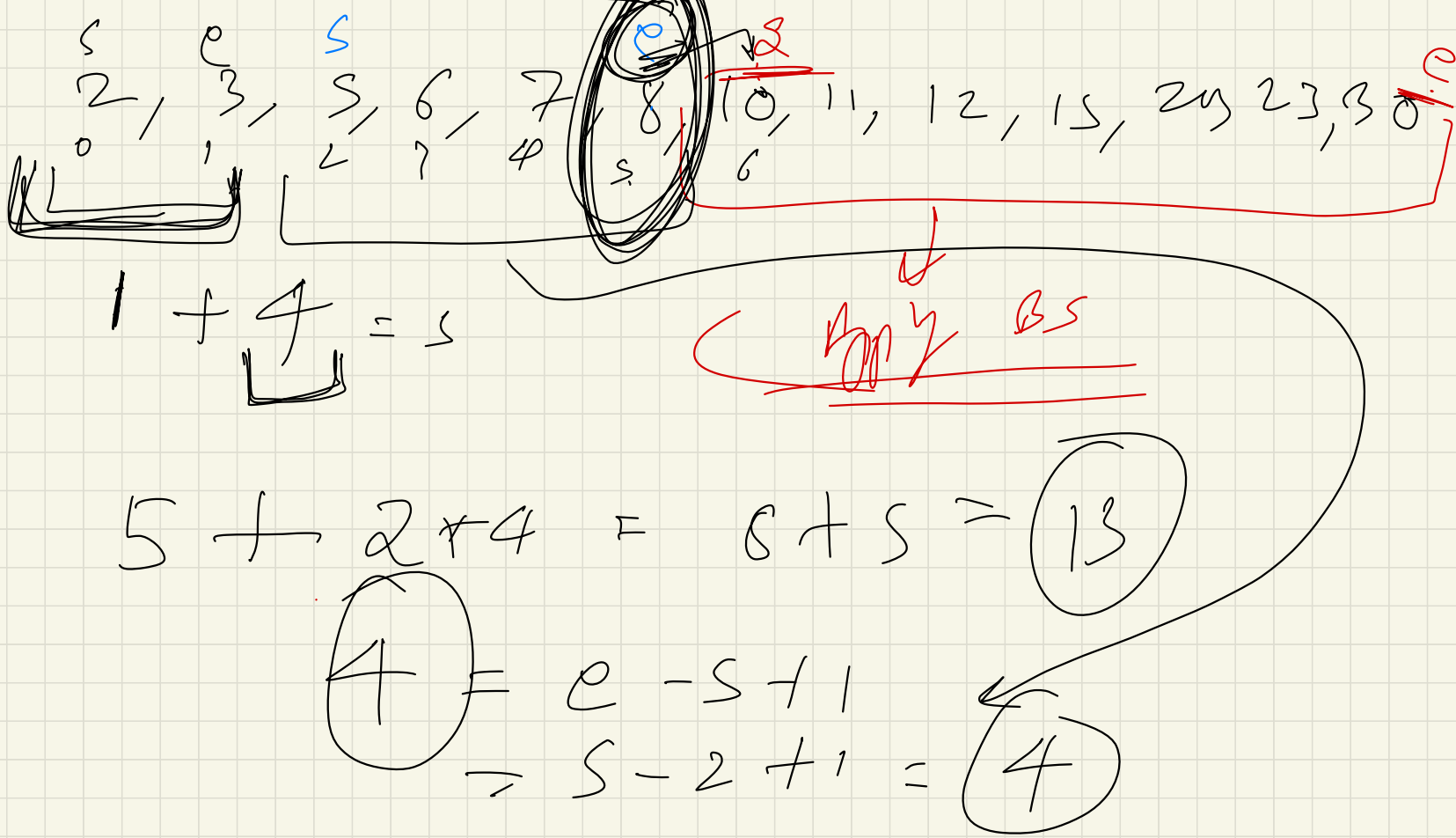
s = m + 1

arr = [2, 3, 5, 6, 7, 8, 10, 11, 12, 15, 20, 23, 30]

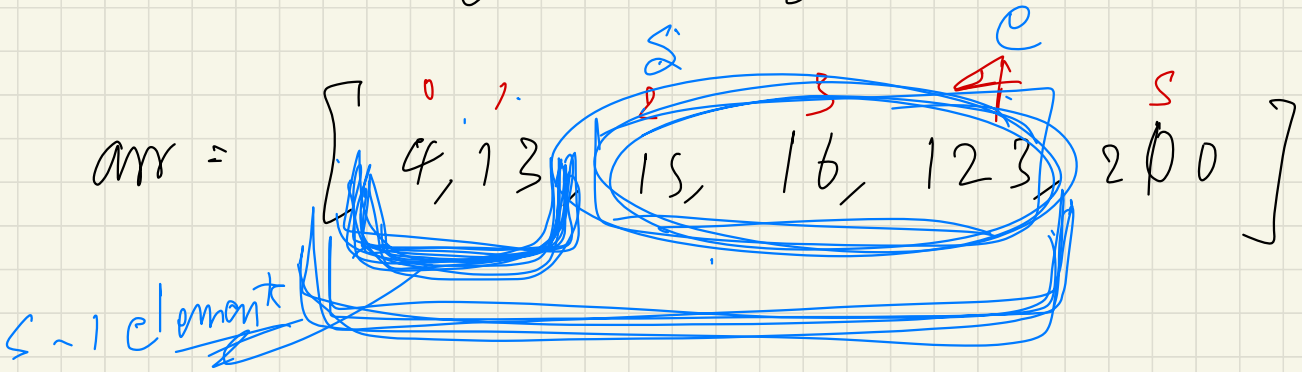
target = 15



log N


s e s e
2, 3, 5, 6, 7, 8, 10, 11, 12, 15, 24 23, 30
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:

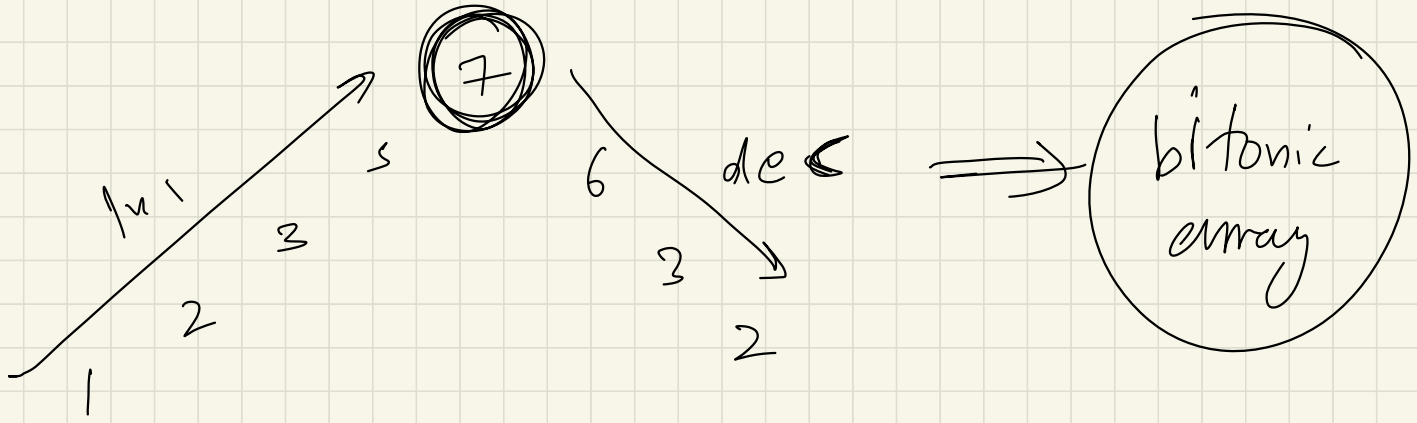


$$e - (s-1) = (e - s + 1)$$

$4 - 2 + 1 = 3$

Ans

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



Q: Find peak in mountain array.

① → Ans

0 0

arr = [0, 2, 1, 0]

② → Ans index = 1

1

0 0

arr = [1, 2, 3, 5, 6, 4, 3, 2]

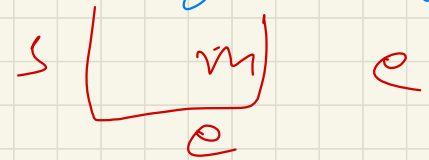
0 1 2 3 4 5 6 7

s m e m

① if $e[mid] > e[mid+1] \Longrightarrow$ You are in the dec part of array

$\Longrightarrow e = mid$

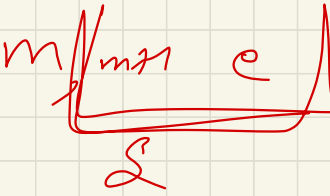
// check in lh



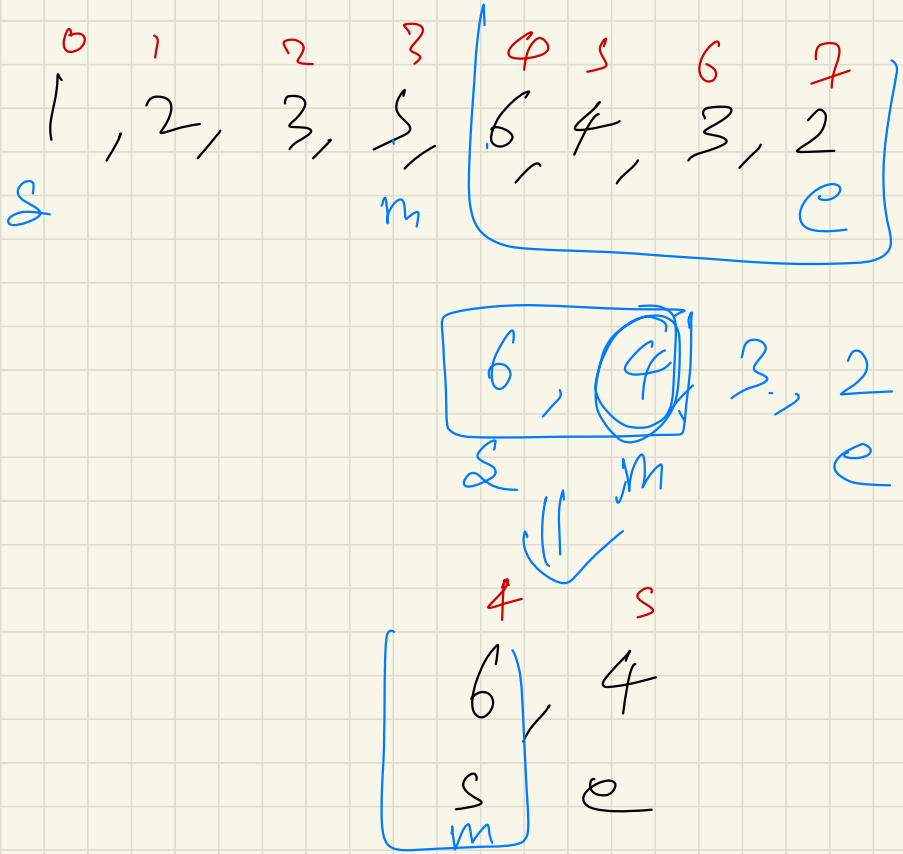
② $e[mid] < e[mid+1] \Longrightarrow$ You are in asc part of arr.

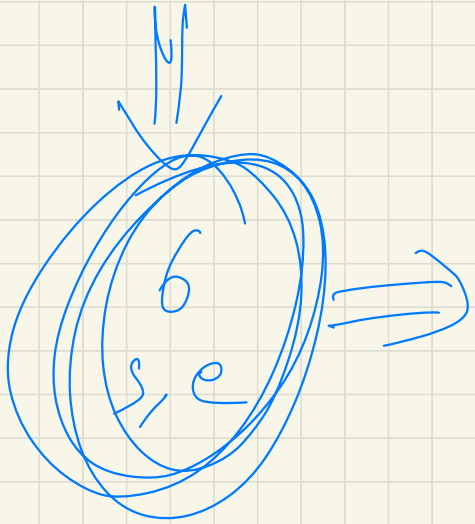
$\Longrightarrow s = mid+1$

$\Longrightarrow$



③ When will loop break?





In the end, s & e both will get to the largest no.

why? ① & ②!

Ans

Q:

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

0 1 2 3 4 5 6

target = 3

Ans:

(1) Find peak element ⇒ 4 index

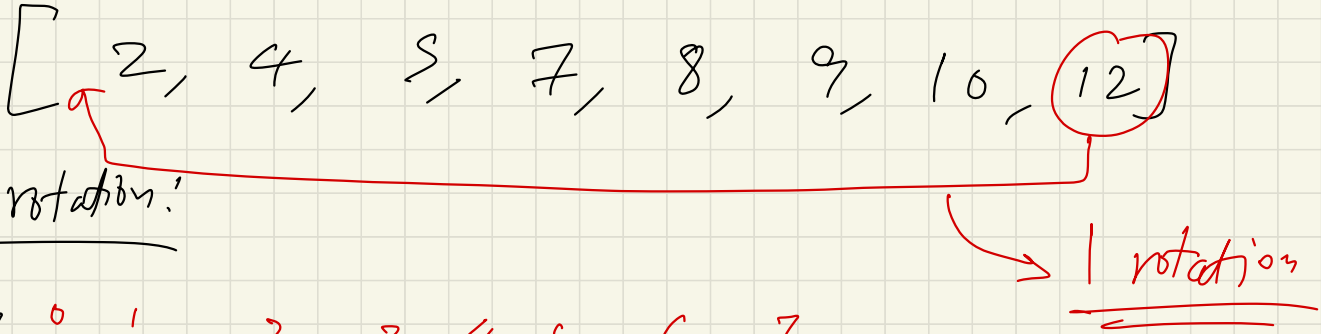
(2) Binary Search in asc array ⇒ (0, 4)

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

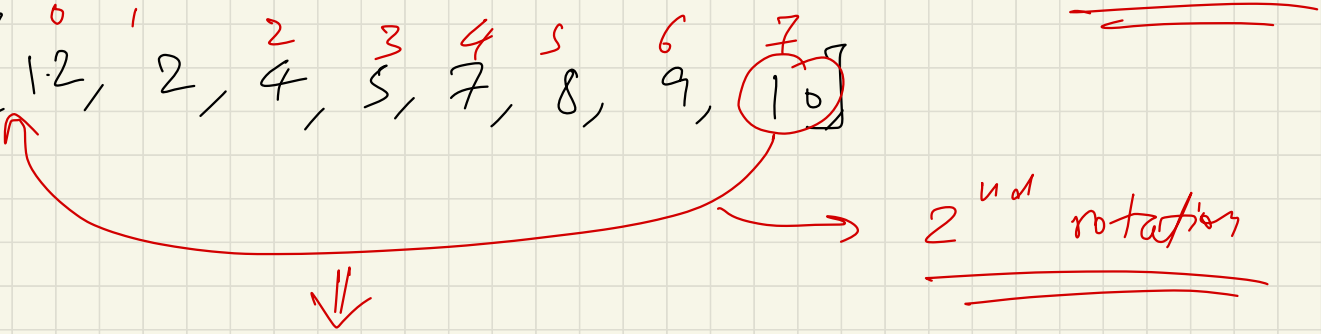
Q: Rotated Binary Search:

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

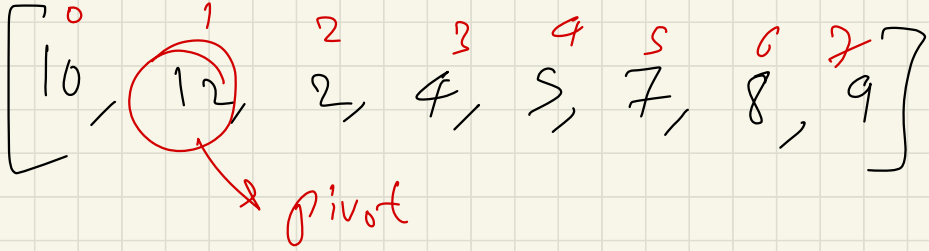
After 1 rotation:



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

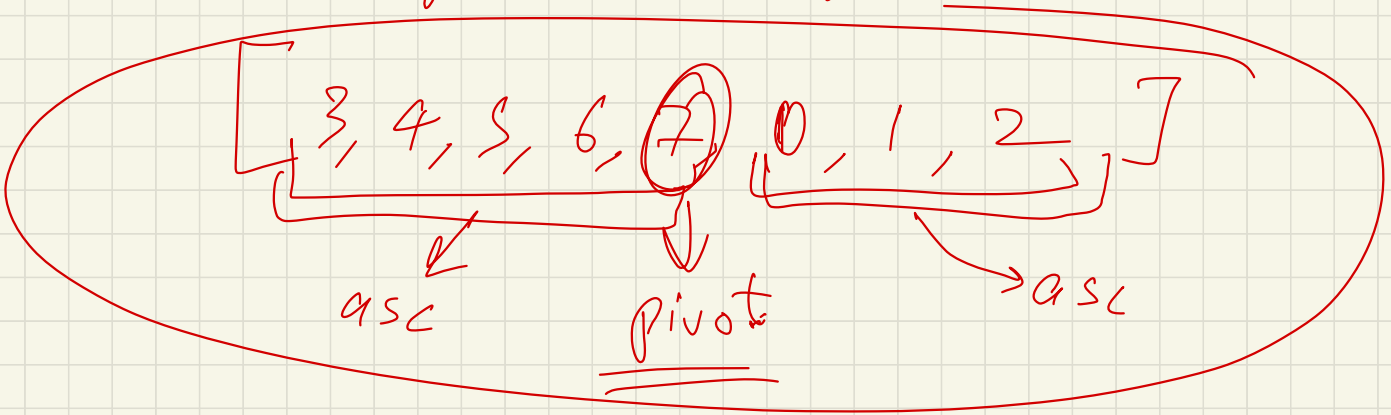


= [10, 12, 2, 4, 5, 7, 8, 9]



① ⇒ Find the pivot in the array.

pivot ⇒ you where your next nos are asc.

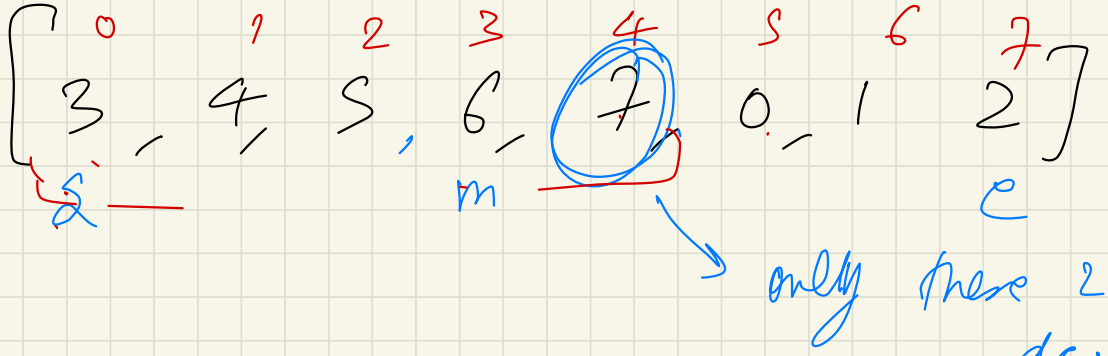


✯ Find pivot

✯ Search in first half ⇒ Simple BS.
(0, pivot)

⭐ otherwise, search in second half : (pt 1, end)

Q: Find pivot :

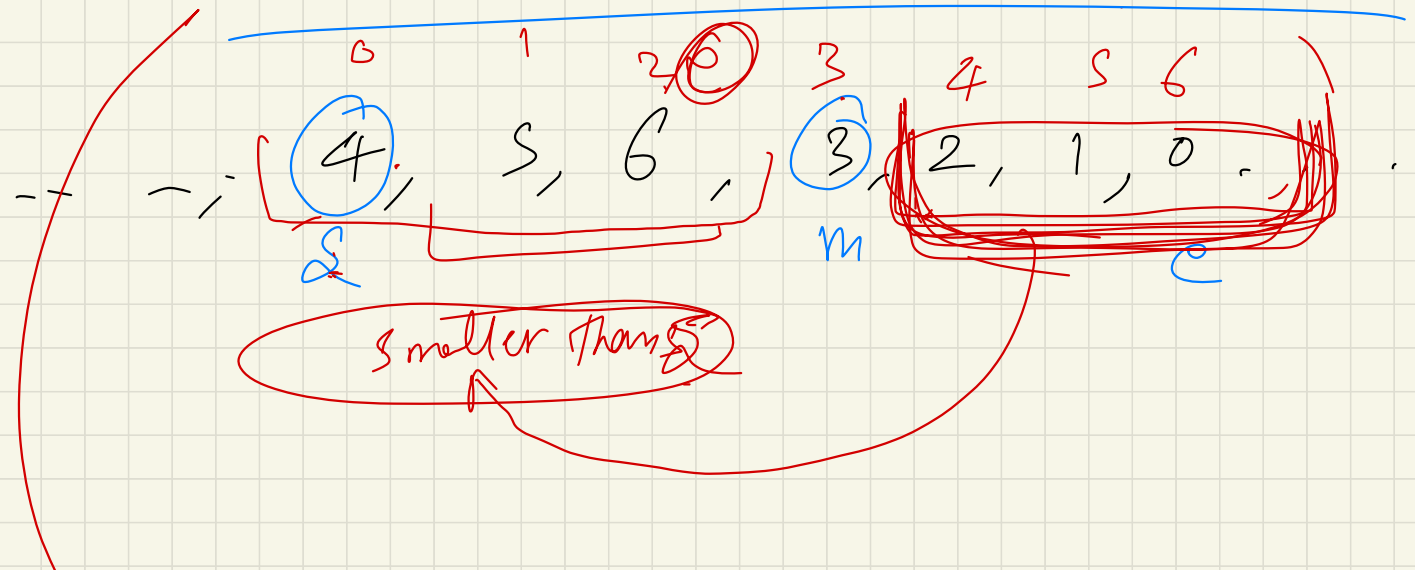


Ans when? → when you find that mid > mid+1 dec. element, i.e. pivot. // one !

Case 2 ✓ if mid element < (mid-1) element

i.e also my ans ⟹ ans = m-1

Case 3 start element ≥ mid element

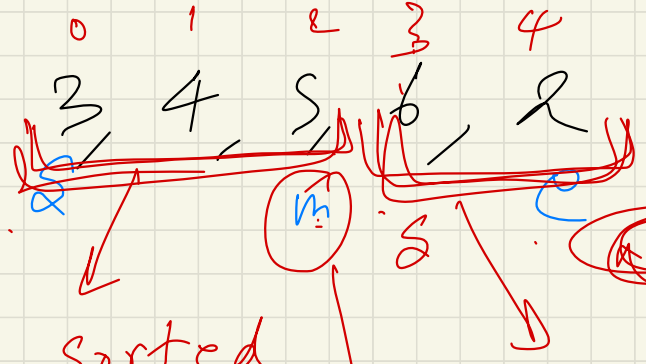


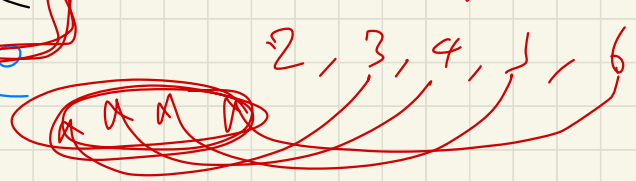
→ In this case, all elements from mid, will be < start.

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

e = mid - 1

Case 4: s element < mid element





Sorted

s = m + 1

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

Hence proved, that bigger no. lie ahead.

Hence, ignore mid & put s = mid + 1

Do BS using pivot.

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

s p e

Case 1 : pivot element = target // Bus

Case 2 : target > start element // search for 6

search space = (s, p-1)

why ? coz all nos. after pivot are < s

Case 3 : target < start element

i.e. we know that all elements from s, pivot are the going to be

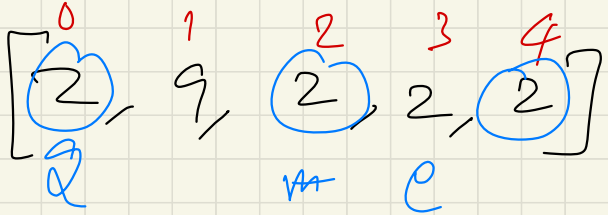
bigger than target. // target = 1

Search space = (pivot + 1 till end)

---

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

⇓ rotate twice



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

Rotated 4 times from original one

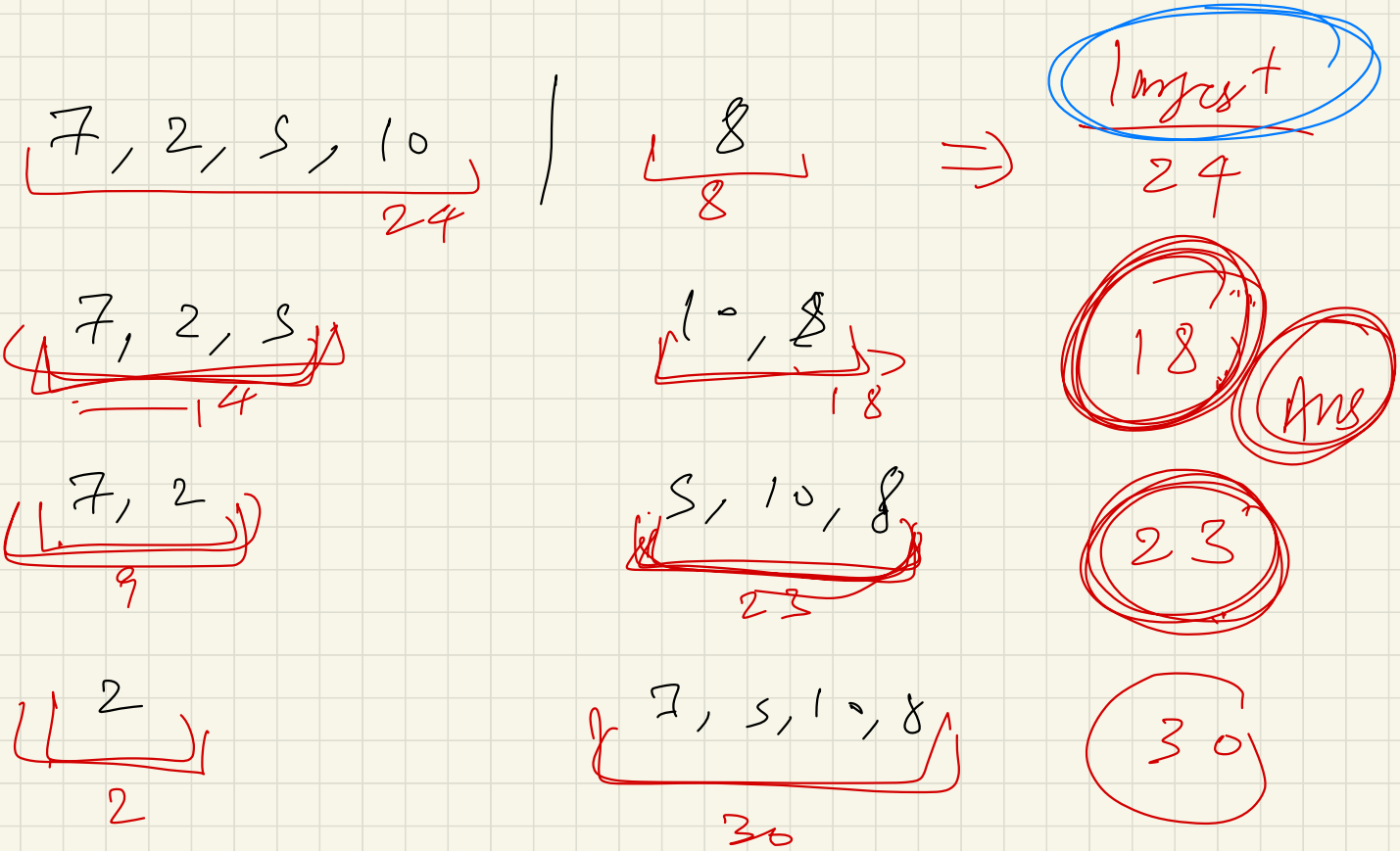
0 1 2 3 4 5 6 7

4 5 6 7 0 1 2 3

pivot = 3

ans = 4

arr = [7, 2, 5, 10, 8] , m = 2



✱① min no. of partitions that we can make = 1

✱② what is the max no. of partitions / m that can be = N

arr = [3, 4, 1, 2] ⟹ [3] [4] [1] [2]

what will be the ans in case 10 :

[7, 2, 5, 10, 8]

Sum of entire array.

= 32

In Case (2) :

Ans for this = max element in array = 4

max value of ans of question = case 1

min value of ans of question = case 2

---

minAns = max value in array

maxAns = sum of all values in array.

[10 , 32]

Here it divided that we use 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 1 if (piece ≤ m) ⇒ end = mid

s = 10 , end = 21

mid = 15

7, 2, 5, 8, 10

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

piece = 3

check 2 ⇒ start = mid + 1 if piece > m

$s = 16$, $e = 21$

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, 10

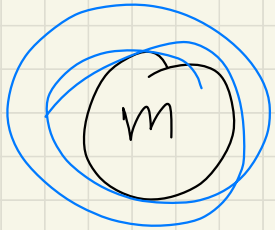
$[7, 2, 5]$, $[8]$, $[17]$ pieces = 3

$$8 = m+1 = 18$$

$$S = 18, \quad e = 18$$

$m = 18$

Ans



// The ans exists definitely, hence by the above 2 steps we will reach it