Problem 7.

Search element in mountain array.

↳ find peak element.
↳ binary search in ascending part of the array.
↳ if not found, binary search in descending part of the array.

Problem 8.

Rotated Binary Search. (unique elements)

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

After 1 rotation

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

After 2nd rotation

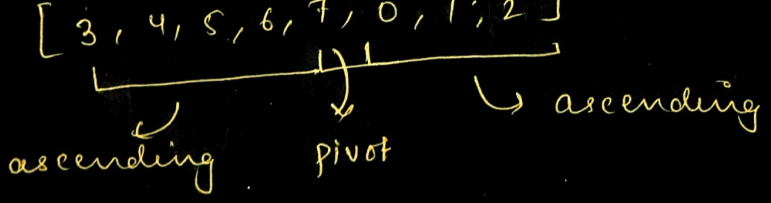
= [10, 12, 2, 4, 5, 6, 7, 8, 9]

Approach - ①

1) Find the pivot in array.

Here - 12

Pivot → from where your next no.s are ascending.



2) search in first half → simple BS (0, pivot)

3) search in second half → (pivot + 1, e)

Differently approaching (Pepcoding)

0	1	2	3	4	5	6	7	8
4	5	6	7	8	9	10	1	2

s ↓ m e

target : 1

1) one part of array (s - m) / (m - e)

↓

shall be sorted.

here s - m is sorted.

but 1 wont exist in the range.

2) s = m + 1.

9	10	1	2
5	6	7	8

s m e.

↓

sorted part but → 1 wont exist in the range.

3) s = m + 1

1	2
s	e
7	8

↓

mid ↳ value found.

So, the main goal is to keep searching in only the ~~sot~~ sorted parts of the array.

Algorithm

1) search for a given part of the array which is sorted

start — mid
OR
mid — end.

2) once the sorted part is found, check if the target can exist, if it can then

i) if the part is start to mid

$e = m - 1$

ii) if part is mid to end

$s = m + 1$

if it can't exist then

i) $s = m + 1$.

ii) $e = m - 1$.

3) return m if arr[m] == target.
else keep repeating till

Sie.

Problem 9.

↳ same as Problem 8 (duplicate elements)

arr [3 1 2 3 3 3 3]
s m e

for an array like the above one, we can't identify the sorted half.

* we will try to trim down this condition

arr[B] = arr[m] = arr[E] → this is the problem.

* if this exists just keep cutting down till we end up getting a sorted half.

for this just

```
increment start
decrement end
continue;
```

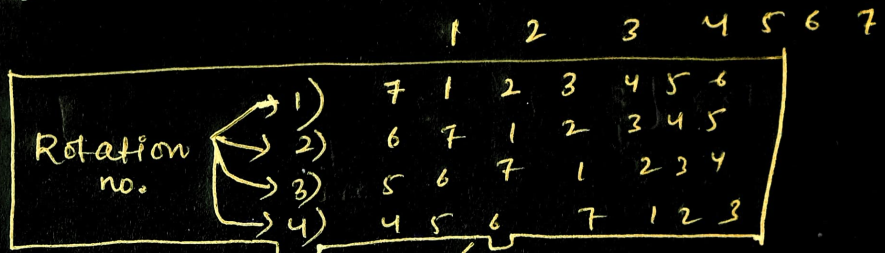
as soon as a sorted half is found the previous logic will hold.

Average time complexity : $O(\log_2 N)$

Worst case : $O(N/2) \rightarrow O(N)$

Problem 10

Count the no. of times array has been rotated.



4 ~~5 6 7 8~~

0	1	2	3	4	5	6
4	5	6	(7)	1	2	3

pos. of PIVOT + 1 = total no. of rotations.

OR.

pos. of smallest element.

Problem 11

Split array largest sum.

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

↓

no. of sub-arrays to divides into

7 | 2, 10, 5, 8 → 25

7, 2 | 10, 5, 8 → 23.

7, 2, 10 | 5, 8 → 19

7, 2, 10, 5 | 8 → 24

Ans = 19.

In each possibility, find the maximum sum of elements.

And then in all the sums, return the minimum value.

* minimum no. of partitions = 1. → case 1
* maximum no. of partition = N. → case 2

in case 1, just sum of elements is the answer.

in case 2, maximum value in the array is the answer because there's just one possibility.

7, 2, 5, 10, 8.

s = 10
e = 32

mid = 21

↳ Try to see if you can split the array with 21 as maximum sum.

7, 2, 5, 8, 10

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

Pieces
2.

if (pieces ≤ m)
↳ end = mid.

s = 10
e = 21

mid = 15.

[7, 2, 5], [8][0]

pieces
3.

if pieces > m)
start = mid + 1.

s = 16
e = 21

mid = 18.

Note:
answer definitely exists

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

s = 16
e = 18

mid = 17.

s = m + 1 → s = 18
e = 18 > answer found