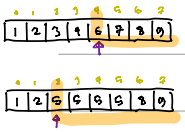
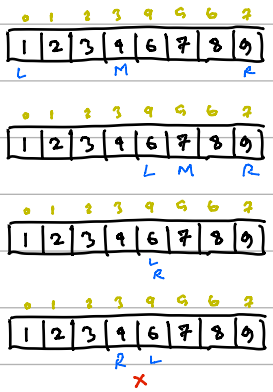
Binary Search

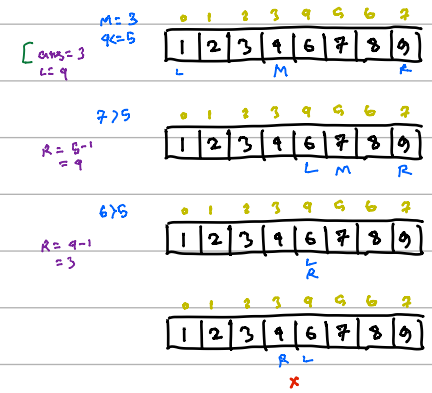
```
while(i<=j)
  mid=(i+j)/2
  if(arr[mid]>=T)
    ans=mid
    j=mid-1
  else i=mid+1
```



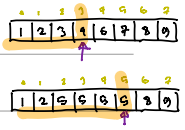
⑥



⑤

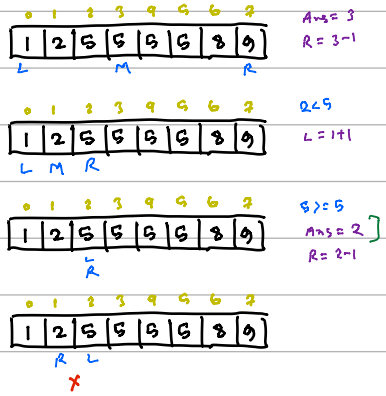



```
while(i<=j)
  mid=(i+j)/2
  if(arr[mid]<=T)
    ans=mid
    i=mid+1
  else j=mid-1
```



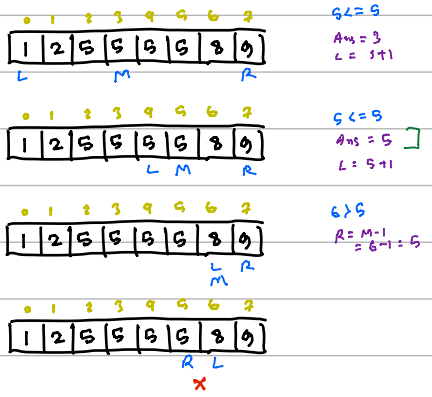
`Arr[M] >=T`
`j=M-1`

⑤

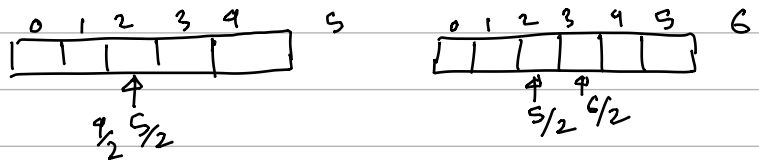


`arr[M] <=T`
`i=M+1`

⑤



0 Median of 2 sorted Array :-



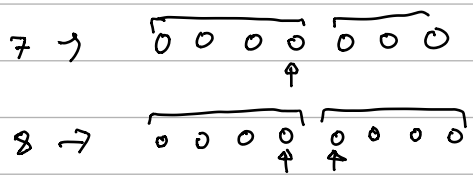
Idea :- nums1 : 3 4 6 len of sorted arr = 8.
nums2 : 5 8 9 11 12 half len = 4, 4

if I take x element from nums1 in the left half of the sorted array (virtual). halflen - x element to be taken from nums2

taking x element from nums1

	1 element		2 element		3 elements	
nums1 :	3	4, 6	3, 4	6	3, 4, 6	{-} ∞
nums2 :	5, 8, 9	11, 12	5, 8	9, 11, 12	5	8, 9, 11, 12
Final arr :	3, 5, 8, 9, 4, 6, 11, 12		3, 4, 5, 8, 6, 9, 11, 12		3, 4, 5, 6, 8, 9, 11, 12	
	✗		✗		✓ correctly sorted.	

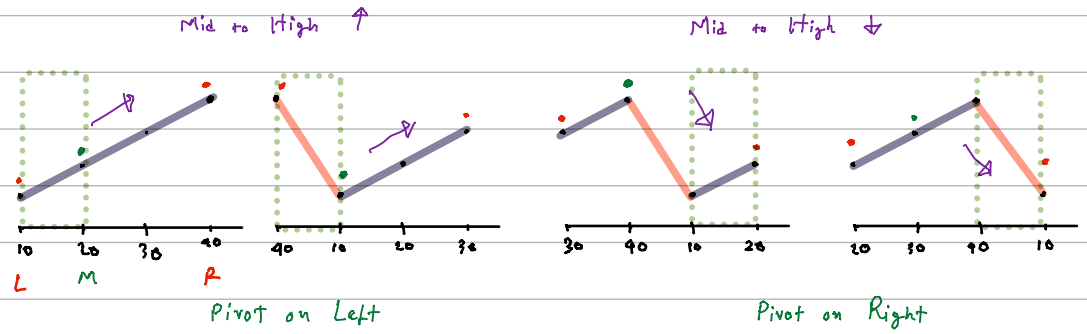
x elements from nums1
HL - x elements from nums2

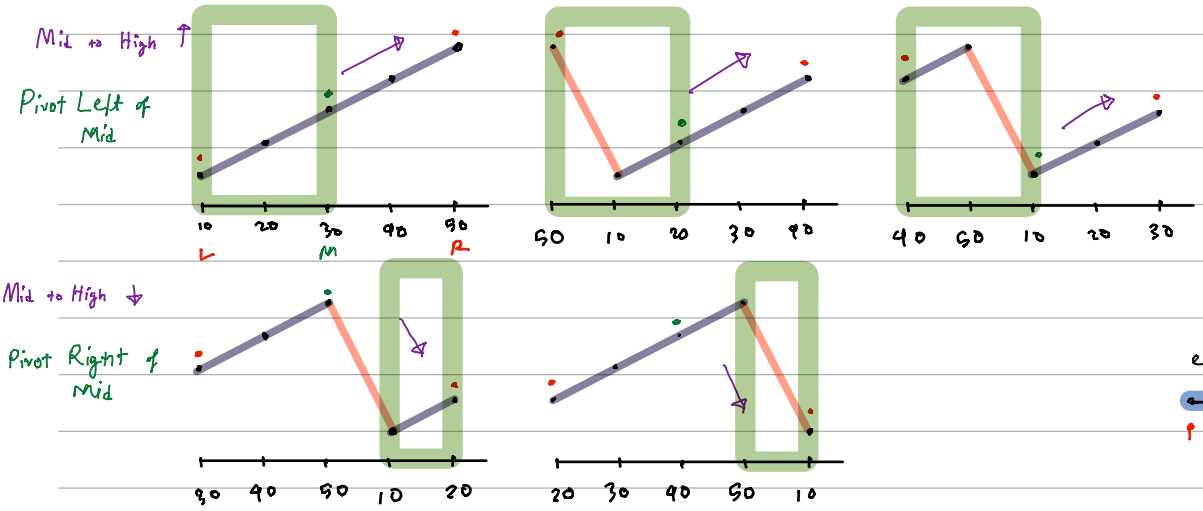


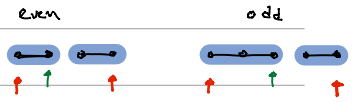
ans2 = max (nums1[x-1], nums2[HL-x-1])

ans2 = min (nums1[x], nums2[HL-x])

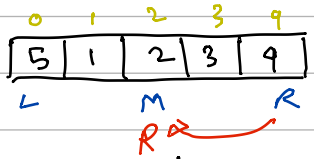
Rotated Sorted Array :-







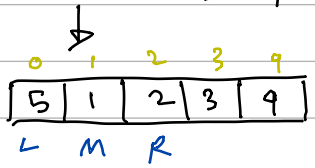
Dry run,



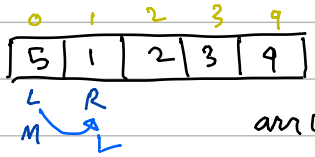
arr[mid] < arr[R]

R = Mid * because mid is a possible candidate

2 is a possible candidate.



1 is possible candidate



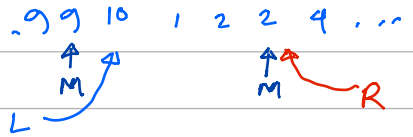
arr[mid] > arr[R]

mid can't be an answer

L = Mid + 1

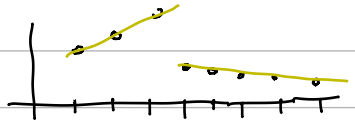
Rotated arr. with duplicates :- 3 3 | 3 4 9 9 1 1 2 2 3 | 3 3 *

Remove — Remove these parts



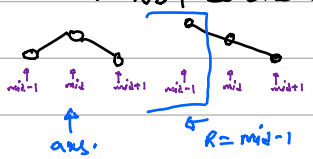

```
if (arr[mid] > arr[R])
    L = mid + 1
else  R = mid.
```

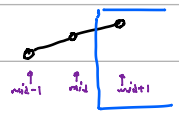
0 Bitonic Sorted / Peak in Mountain:



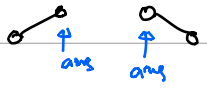
all possibility

✓ Not corner :





✓ corner :

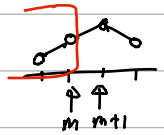


Directly answerable. ✓

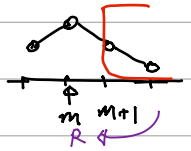
another smaller version :

if arr[mid] < arr[mid+1]
i = mid + 1
else j = mid

return i or j ;



0 → m can be discarded



m+1 → n-1 can be discarded.

Find single Element in Sorted array:

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

Devide & Conquer

1 1 2 3 3 4 4 5 5

odd possible — even ✗

Find peak : unsorted array

[2 3 1 2 6 4 3 2]

0 1 2 3 4 5 6 7

2 3 1 2 6 4 3 2

may be possible but Right side guaranteed 1 possible candidate. — Not ans — possible candidate 6

if Array was [2 3 1 2 6 7 8 9]

then 9 would have been the answer.

Kth missing +ve number :-

#⭐, #Trick

①, 2, 3, 4, ⑤, ⑥, 7, ⑧, ⑨, 10, 11 ...

nums; [2 | 3 | 4 | 7 | 11] return 5th missing no.

o/p: 9

⇒

1 2 3 4 5 → number to be present.

2 3 4 7 11 missing till = 4-3 = 1

```
if (arr[mid] - (mid+1) < k)
    i = mid+1
else j = mid-1
```

Till this point 1 number is missing we need 5th missing no.

So go Right.

1 2 3 4 5

2 3 4 7 11

L R M

7-4 = 3

0 1 2 3 4

2 3 4 7 11

L=4 7

L R M

11-5 = 6

* → Here missing no. < nums[i] = 11

so; we have 4 no. to the left & need the 5th missing no.

so 4+5 = 9 is the number.

* since at 11 loop broke & 4 present no. at back, the 5th missing must be 4+5.

0 Monotonic boolean.

0 Maximum running time of computers :-

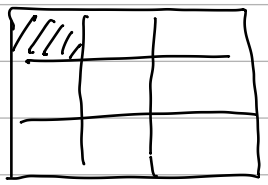
n computer batteries : [10, 10, 3, 5]

n = 3

10 10 5 & 3 8 min max possible

check T x 3 = Time needed Sum += min(batteries[i], T)

0 Is possible to cross :-



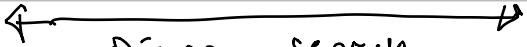
[[0,1], [1,2], ...

at i+1 th only grid [arr[i][0]]
[arr[i][1]]
is flooded.

[[0,1], [1,2] ... [ ]]

No box in grid filled

all box in grid is filled.



Binary search is Possible on DFS or BFS.