Binary Search

1. define a region (region of search)
2. try to eliminate half of the region
3. Continue the search in the other half.

$$\left\{ \begin{array}{l} TC: O(\log_2 N) \\ SC: O(1) \end{array} \right.$$

N → length of the region.

Q Search minimum value in a rotated sorted array.

Or Search pivot index in a Rotated sorted array.

0	1	2	3	4	5	6
20	40	46	70	1	2	7

arr[] = {20, 40, 46, 70, 1, 2, 7}

pivot

rotated Sorted Array

Brute force

↳ linear search

↳ store min value

TC: O(N)
SC: O(1)

$$\begin{array}{cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ arr[\,] = \{ & 20, & 40, & 46, & 70, & 1, & 2, & 7\} \\ & \uparrow & & & \uparrow & & & \uparrow \\ & si & & & \underline{mid} & & & ei \end{array}$$

if <u>Case</u> $arr[mid-1] > arr[mid]$

$mid \longrightarrow pivot$ }

OR $\underline{arr[mid+1] < arr[mid]}$

$mid+1$

↳ pivot

else if <u>Case</u> $arr[si] <= arr[mid]$ {leftside is sorted}

$si = mid+1;$

else <u>Case</u>   {right side is sorted}

$ei = mid-1;$

Search in a rotated Sorted Array

```
            0    1    2    3    4    5   6   7
int[] arr = { 20, 40, 45, 46, 70, 1, 2, 7 }        target = 2
              ↑              ↑              ↑
              si             mid            ei
```

Case : arr[mid] == target

Case : is leftside Sorted

↳ Range [20, 46]

present → ei = mid-1;

Not → si = mid+1;

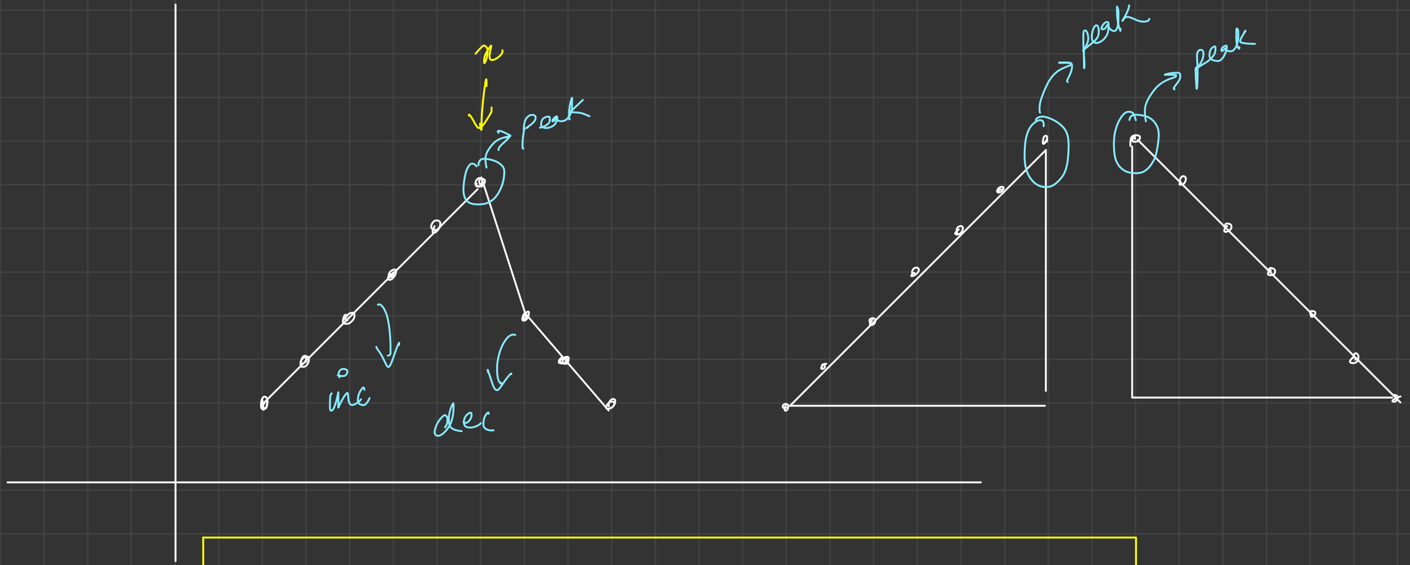
Case : Range = [arr[mid], arr[ei]]

present → si = mid+1;

Not → ei = mid-1;

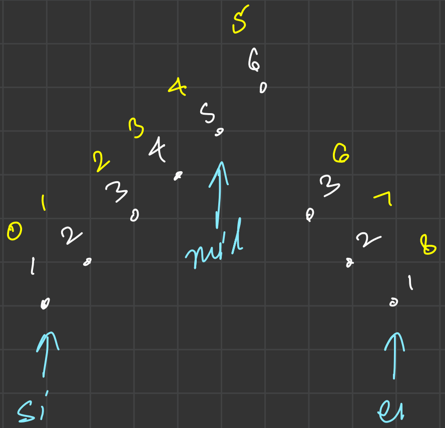
Peak index in a Mountain Array

$arr[\,] = \{1, 2, 3, 4, 5, 6, 3, 2, 1\}$



$$\boxed{arr[x-1] < arr[x] \;\&\&\; arr[x+1] < arr[x]}$$

↳ peak at index x



Case — verify as Peak

→ return mid

Case — arr[mid-1] < arr[mid]

→ leftside is inc.

(move right)

→ si = mid+1

Case — (move left)

→ ei = mid-1