⇒ Variation of Binary Search

→ **BSLB**

target = 3

ans = 2

	0	1	2	3	4	5	6	7	8	9
arr =	1	2	3	3	3	3	3	4	5	5

i, j, mid

Note:- if current target element is equal to previous element that means, we have to move left for ans.

BSLB

```
int i=0, j=n-1;
while ( i <=j ){
        mid = (i+j)/2;
        if ( arr[mid] == target){
                if ( mid-1 >=0 && arr [mid] == arr [mid-1] ) {
                        j = mid-1;
                } else {
                        return mid;
                }
        } else if( arr [mid] < target){
                i = mid+1 ;
        } else if ( arr [mid] > target){
                j = mid-1;
        }
}
```

→ BSUB

target = ~~3~~ 5

arr =

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

i j

mid

BSUB

```
int i=0, j=n-1;
while ( i <=j ){
        mid = (i+j)/2;
        if (arr[mid] == target){
                if (arr[mid] == arr[mid+1] ) {
                        i = mid+1;
                } else {
                        return mid;
                }
        } else if( arr[mid] < target){
                i = mid+1;
        } else if ( arr[mid] > target){
                j = mid-1;
        }
}
```

Find Last Occurrence (BSUB)

```
public static int BSUB(int[] arr, int n, int target) {
    int i = 0;
    int j = n - 1;
    while ( i <= j ) {
        int mid = (i + j) / 2;
        if ( arr[mid] == target ) {
            if ( mid + 1 < n && arr[mid] == arr[mid + 1] ) {
                i = mid + 1;
            } else {
                return mid;
            }
        } else if ( arr[mid] > target ) {
            j = mid - 1;
        } else if ( arr[mid] < target ) {
            i = mid + 1;
        }
    }
    return -1;
}
```

Ques given array with repeated elements. given target.

Imp find no. of target elements in log(n)

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

$\underbrace{2, 2, 2, 2, 2, 2}_{x \quad\quad y}$

target = 2

ans) x = find first index of target → BSLB

y = find last index of target → BSUB

$$\text{ans} = y - x + 1$$

(leetcode 34)

Find The Index of Rotation

↳ sorted
↳ rotated

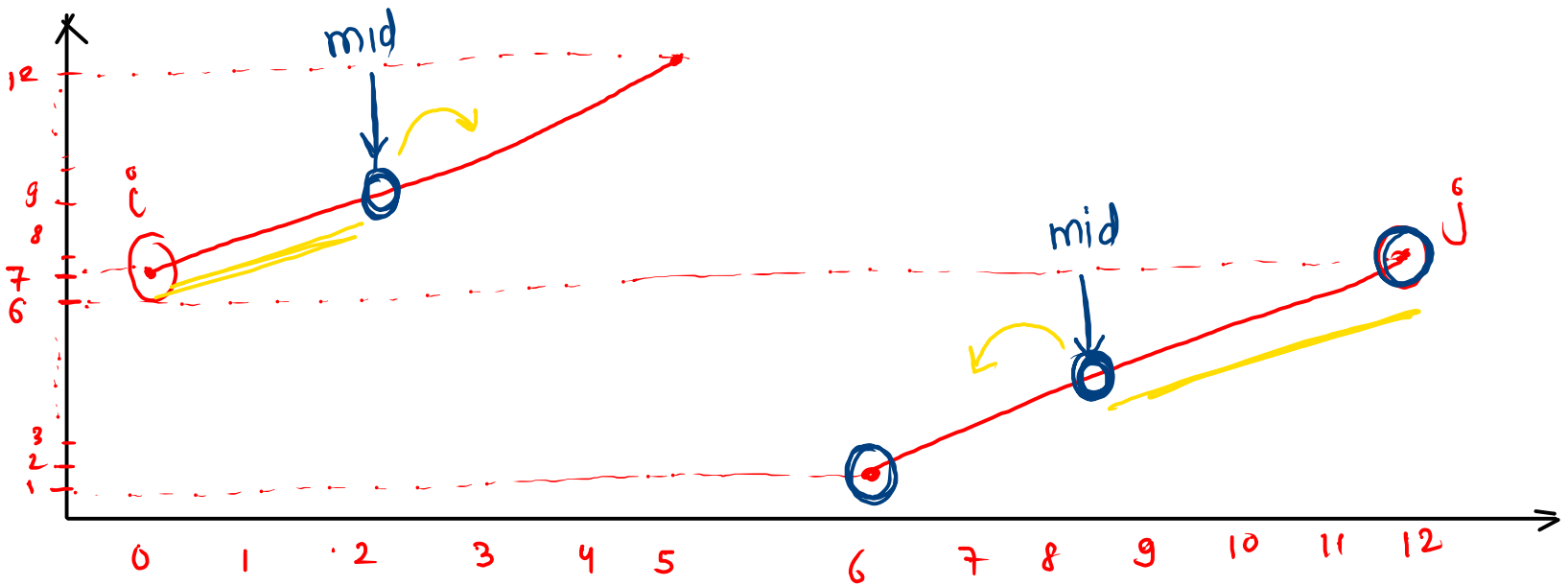
(T.C = log(n))

ans = 5

arr = [7 , 8 , 9 , 10 , 11 , 12 , 1 , 2 , 3 , 4 , 5 , 6]

7	8	9	10	11	12	1	2	3	4	5	6
0	1	2	3	4	5	6	7	8	9	10	11

i, mid, j



Code

```
int i=0, j=n-1;
while( i<=j) {
    mid = (i+j)/2;
    if (arr[mid] <= arr[mid-1]
        &&
        arr[mid] <= arr[mid+1]
        return mid-1;
    } else if (arr[mid] <= arr[j]) {
        j = mid-1;
    } else if (arr[mid] >= arr[i]) {
        i = mid+1;
    }
}
```

arr = [7 , 8 , 9 , 10 , 11 , 12 , 1 , 2 , 3 , 4 , 5 , 6]

0 1 2 3 4 5 6 7 8 9 10 11

mid

mid

mid = 11

$\Rightarrow (mid+1)\%\, n$

$\Rightarrow (12)\%\, 12$

$\Rightarrow 0$

Note:-

Clockwise rotation :- $(x \% n)$

anti-clockwise rotation :- $(x + n) \% n$

code

```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[]arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }
    System.out.println(findIndexOfRotation(arr, n));
}
public static int findIndexOfRotation(int[] arr, int n) {
    int i = 0;
    int j = n - 1;
    while ( i <= j ) {
        int mid = (i + j) / 2;
        int prev = (mid - 1 + n) % n;
        int next = (mid + 1) % n;
        if ( arr[mid] <= arr[prev] && arr[mid] <= arr[next] ) {
            return mid - 1;
        } else if ( arr[mid] <= arr[j] ) {
            j = mid - 1;
        } else if ( arr[mid] >= arr[i] ) {
            i = mid + 1;
        }
    }
    return -1;
}
```