

Searching Technique

BINARY SEARCH

UPPER BOUND

LOWER BOUND

PROBLEMS

TRICKS



In the end, it's YOU vs YOU. Be the best today, to defeat your yesterday!!

What is Binary Search

is a searching Technique which is mainly used to find target element in 'sorted Search Space' (start \longleftrightarrow end)

1 2 3 4 5 6 7

7 6 5 4 3 2 1

$\text{mid} = \frac{(s+e)}{2}$

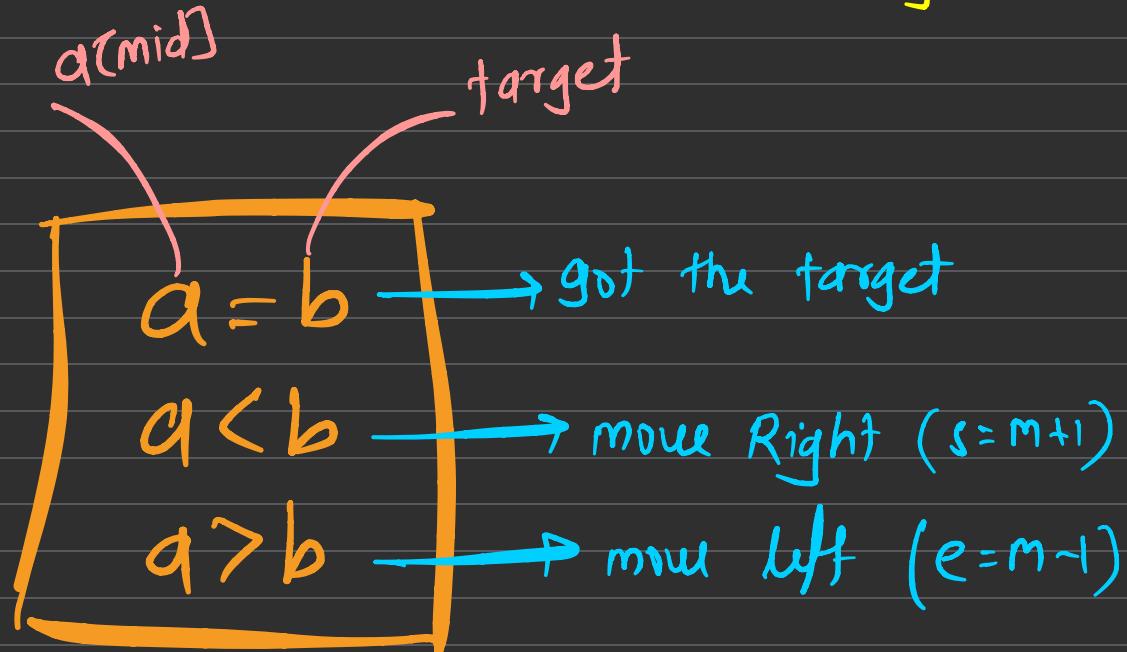
1	2	4	6	8	9	11	14	18
0	1	2	3	4	5	6	7	8

$N = 9$

$\text{target} = 12$

$s = 0$
 $e = 8$
 $\text{mid} = 4$

$s = 7$
 $e = 8$
 $\text{mid} = 7$



```
int binarySearch ( int a[], int target, int N)  
{
```

```
    int start = 0 , end = N-1
```

```
    while ( start <= end )
```

```
    {  
        mid = ( start + end ) / 2 ..
```

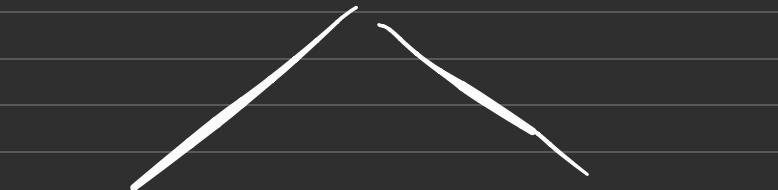
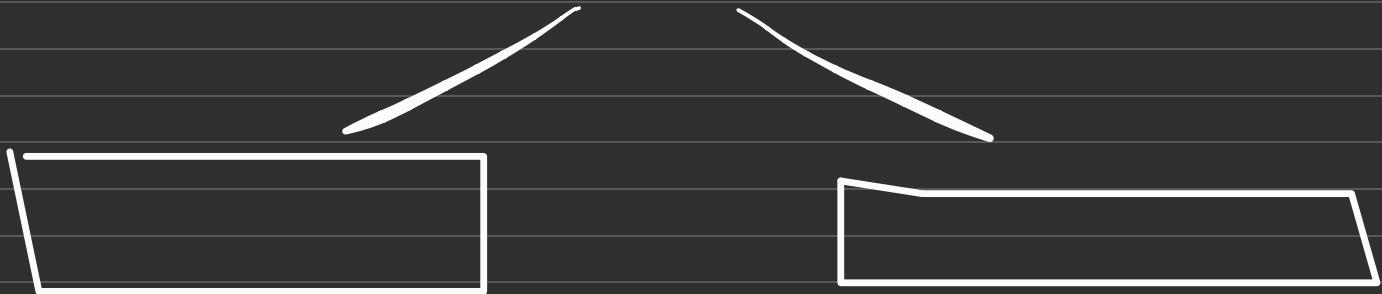
```
        if ( a[mid] == target )  
            return mid
```

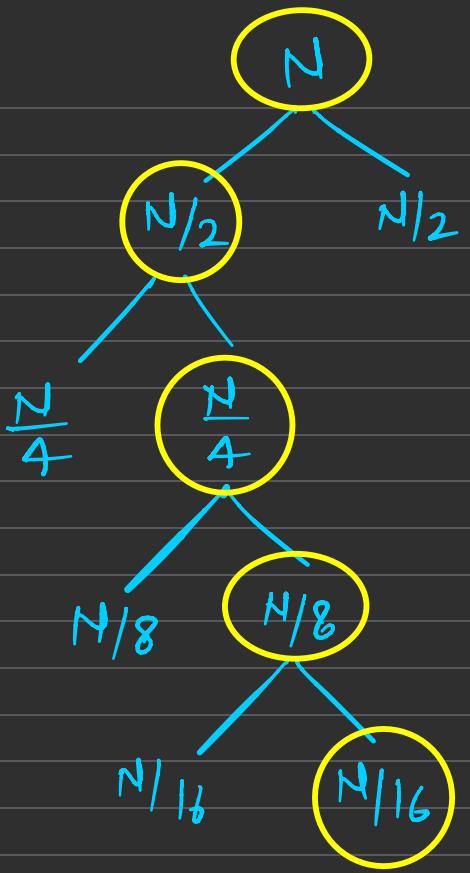
```
        else if ( a[mid] < target )
```

```
            start = mid + 1
```

```
        else  
            end = mid - 1
```

```
    return -1
```





$$\begin{aligned}
 N &= N/2^0 \\
 \frac{N}{2} &= N/2^1 \\
 \frac{N}{4} &= N/2^2 \\
 \frac{N}{8} &= N/2^3 \\
 \frac{N}{16} &= N/2^4
 \end{aligned}$$

}

$$\frac{N}{2^K} = 1$$

$$N = 2^K$$

$$\log N = \log 2^K$$

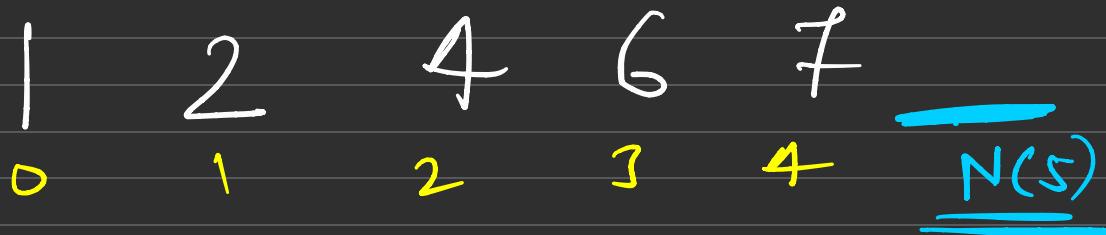
$$\log N = K \log 2$$

$$\frac{\log N}{\log 2} = K$$

$$K = \log_2 N$$

Lower Bound

$N = 5$



$$LB(2) = 1$$

$$LB(3) = 2$$

$$LB(10) = N$$

index

ans = 1

LB

1	2	4	6	8	9	11	14	18
0	1	2	3	4	5	6	7	8

$N = 9$

$\text{target} = 2$

$s = 0$
 $e = 8$
 $\text{mid} = 4$

$s = 0$
 $e = 6$
 $n = 6$

$s = 0$
 $e = 3$
 $\text{mid} = 1$

if ($a(\text{mid}) \geq \text{target}$)
 } $\text{index} = \text{mid}$
 $\text{end} = \text{mid} - 1$
 else
 $\text{start} = \text{mid} + 1$

1	3	4	6
0	1	2	3

\uparrow
 \downarrow
 c

$N=4$

~~$x=10$~~

int index = ?

~~+ $B(10)$~~

$s=0$

$s=3$

$e=3$

$e=3$

$mid=1$

$mid \rightarrow$

$s=2$

$e=3$

$mid=2$

$s=4$
 $e=2$

6
10

index = N

start = 0 end = N - 1

index = N

while (start ≤ end)

} mid = (start + end) / 2

if (a(mid) ≥ target)

} index = mid

end = mid - 1

{ else

} start = mid + 1

return index

Upper Bound

$N = 5$



$$\begin{aligned} UB(2) &= 2 \\ UB(3) &= 2 \\ UB(10) &= N(5) \end{aligned}$$

\left. \begin{array}{l} \\ \\ \end{array} \right\} \text{index}

if ($a[mid] > \text{target}$)
} $\text{index} = \text{mid}$
 $\text{end} = \text{mid} - 1$

else
 $\text{start} = \text{mid} + 1$

35. Search Insert Position

Solved 

Easy

Topics

Companies

Re-do

Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

You must write an algorithm with $O(\log n)$ runtime complexity.

Example 1:

```
| Input: nums = [1,3,5,6], target = 5  
| Output: 2
```

Example 2:

```
| Input: nums = [1,3,5,6], target = 2  
| Output: 1
```

Example 3:

```
| Input: nums = [1,3,5,6], target = 7  
| Output: 4
```

~~$x = 0$~~

(3)						
1	2	4	5	7	8	
0	1	2	3	4	5	

~~$x = 7$~~

for ($i = 0 : i < N : i++$)

} if ($q(i) == \text{target}$)
 return i

~~$x = 3$~~

(2) index

else if ($q(i) > \text{target}$)
 return i

$O(N)$



return M

Rich People Value Time



Poor People Value Money



Floor in a Sorted Array



Difficulty: Easy

Accuracy: 33.75%

Submissions: 475K+

Points: 2

Average

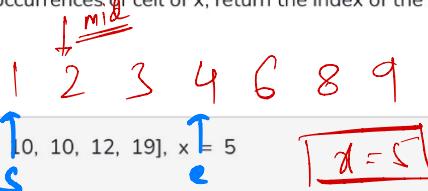
Time: 30m

Given a sorted array $\text{arr}[]$ and an integer x , find the index (0-based) of the largest element in $\text{arr}[]$ that is less than or equal to x . This element is called the **floor** of x . If such an element does not exist, return -1.

$$\text{a}(j) \leq x$$

Note: In case of multiple occurrences of floor of x , return the index of the last occurrence.

Examples



Input: $\text{arr}[] = [1, 2, 8, 10, 10, 12, 19]$, $x = 5$

Output: 1

Explanation: Largest number less than or equal to 5 is 2, whose index is 1.

Input: $\text{arr}[] = [1, 2, 8, 10, 10, 12, 19]$, $x = 11$

Output: 4

Explanation: Largest Number less than or equal to 11 is 10, whose indices are 3 and 4. The index of last occurrence is 4.

Input: $\text{arr}[] = [1, 2, 8, 10, 10, 12, 19]$, $x = 0$

Output: -1

Explanation: No element less than or equal to 0 is found. So, output is -1.

index = -1

$x = 0$

m	1	2	3	4
0	1	2	3	

s = 0 e = 3

m = 1

s = 0 e = 0

m = 0

(s = 0, e = -1)

int upperBound (arr[], N, target)

}

start = 0 , end = N - 1

index = -1

while (s ≤ e)

?

mid = (s + e) / 2

if (arr[m] ≤ x)

? index = mid

? s = m + 1

else

? e = m - 1

return index

{

34. Find First and Last Position of Element in Sorted Array

Solved 

Medium

Topics

Companies

Re-do

Given an array of integers `nums` sorted in non-decreasing order, find the starting and ending position of a given `target` value.

If `target` is not found in the array, return `[-1, -1]`.

You must write an algorithm with $O(\log n)$ runtime complexity.

Example 1:

Input: `nums = [5,7,7,8,8,10]`, `target = 8`

Output: `[3,4]`

Example 2:

Input: `nums = [5,7,7,8,8,10]`, `target = 6`

Output: `[-1,-1]`

Example 3:

Input: `nums = []`, `target = 0`

Output: `[-1,-1]`

$\text{start} = -1$, $\text{end} = -1$

gives index
of first
occ .

{

```
for (i=0; i<N; i++)
    if (a(i) == x)
        start = i
        break
```

gives index
of last
occ .

{

```
for (i=N-1; i>=0; i--)
    if (a(i) == x)
        end = i
        break
```

(2, 5)
/ /

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

N=8

k=3

start = -1, end = -1

start = -1, 2

end = -1, 2

2

4

S

for (i=0 ; i < N ; i++) — O (N)

?

if (a(i) == k)

?

if (start == -1)

start = i

≤ end = i

Y

index = 7
2

	1	2	3	3	3	3	4	6	-
	0	1	2	3	4	5	6	7	

sc
m

N=8
 $\chi = 3$

s=0

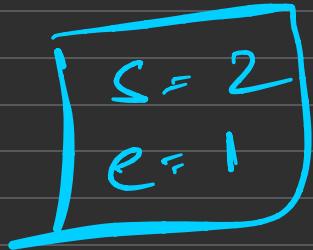
e=7

mid=3

s=0

e=2

mid=1



s = 2

e = 2

int binarySearch (int a[], int target, int N)
{

 int start = 0 , end = N-1

 while (start ≤ end)

 mid = (start + end) / 2

 if (a[mid] == target)

 ans = mid

 e = mid - 1

 else

 s = mid + 1

}

return ans

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

N=8
 $\chi = 3$

int binarySearch (int a[], int target, int N)
{

 int start = 0 , end = N-1
 while (start ≤ end)
 {
 mid = (start + end) / 2

 if (a[mid] == target)

 ans = mid

 s = mid + 1
 }

 else
 e = mid - 1

 return ans

540. Single Element in a Sorted Array

Solved 

Medium

Topics

Companies

Re-do

You are given a sorted array consisting of only integers where every element appears exactly twice, except for one element which appears exactly once.

Return *the single element that appears only once*.

Your solution must run in $O(\log n)$ time and $O(1)$ space.

Example 1:

Input: `nums = [1,1,2,3,3,4,4,8,8]`

Output: 2

Example 2:

Input: `nums = [3,3,7,7,10,11,11]`

Output: 10

Peak element



Difficulty: Medium

Accuracy: 38.86%

Submissions: 618K+

Points: 4

Average Time: 30m

You are given an array **arr[]** where no two adjacent elements are same, find the index of a peak element. An element is considered to be a **peak** if it is greater than its adjacent elements (if they exist).

If there are multiple peak elements, Return index of any one of them. The output will be "true" if the index returned by your function is correct; otherwise, it will be "false".

Note: Consider the element **before** the **first** element and the element **after** the **last** element to be **negative infinity**.

Examples :

$\Theta(\log N)$ $\Theta(N)$

Input: arr = [1, 2, 4, 5, 7, 8, 3]

Output: true

Explanation: arr[5] = 8 is a peak element because arr[4] < arr[5] > arr[6].

$s = 0$

$e = 6$

$\text{mid} = 3$

$s = 4$

$e = 6$

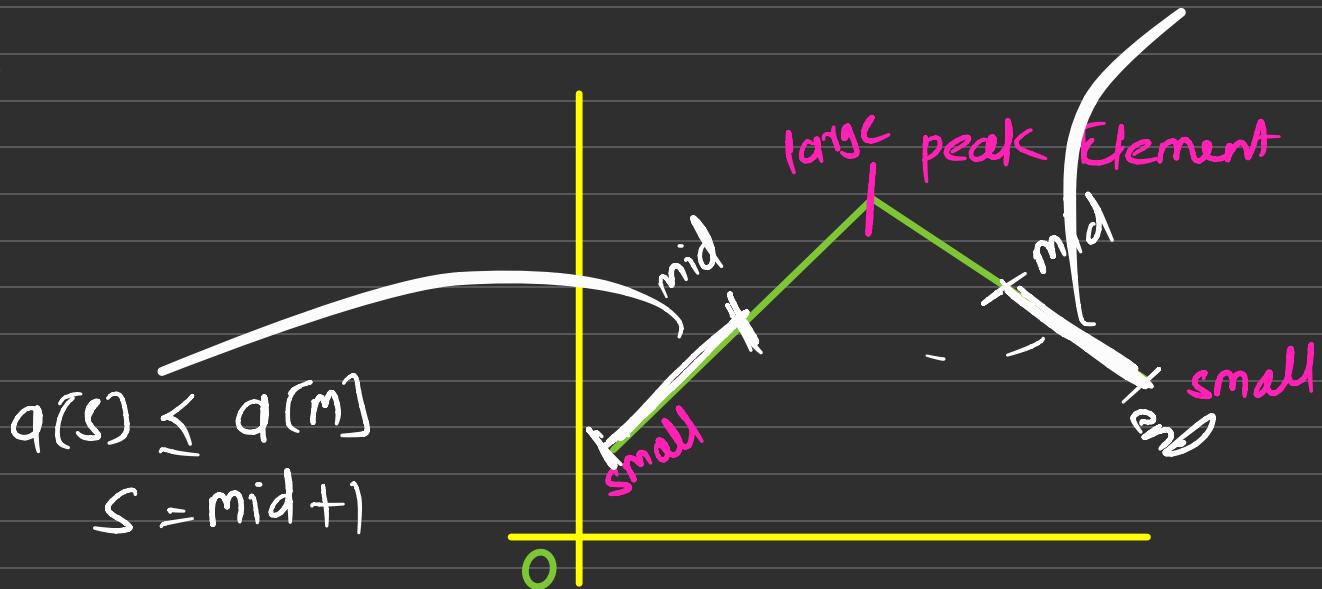
$\text{mid} = 5$

1	2	4	5	7	8	3
---	---	---	---	---	---	---

0 1 2 3 4 5 6

↑
mid

$a(m) \geq a(\text{end})$



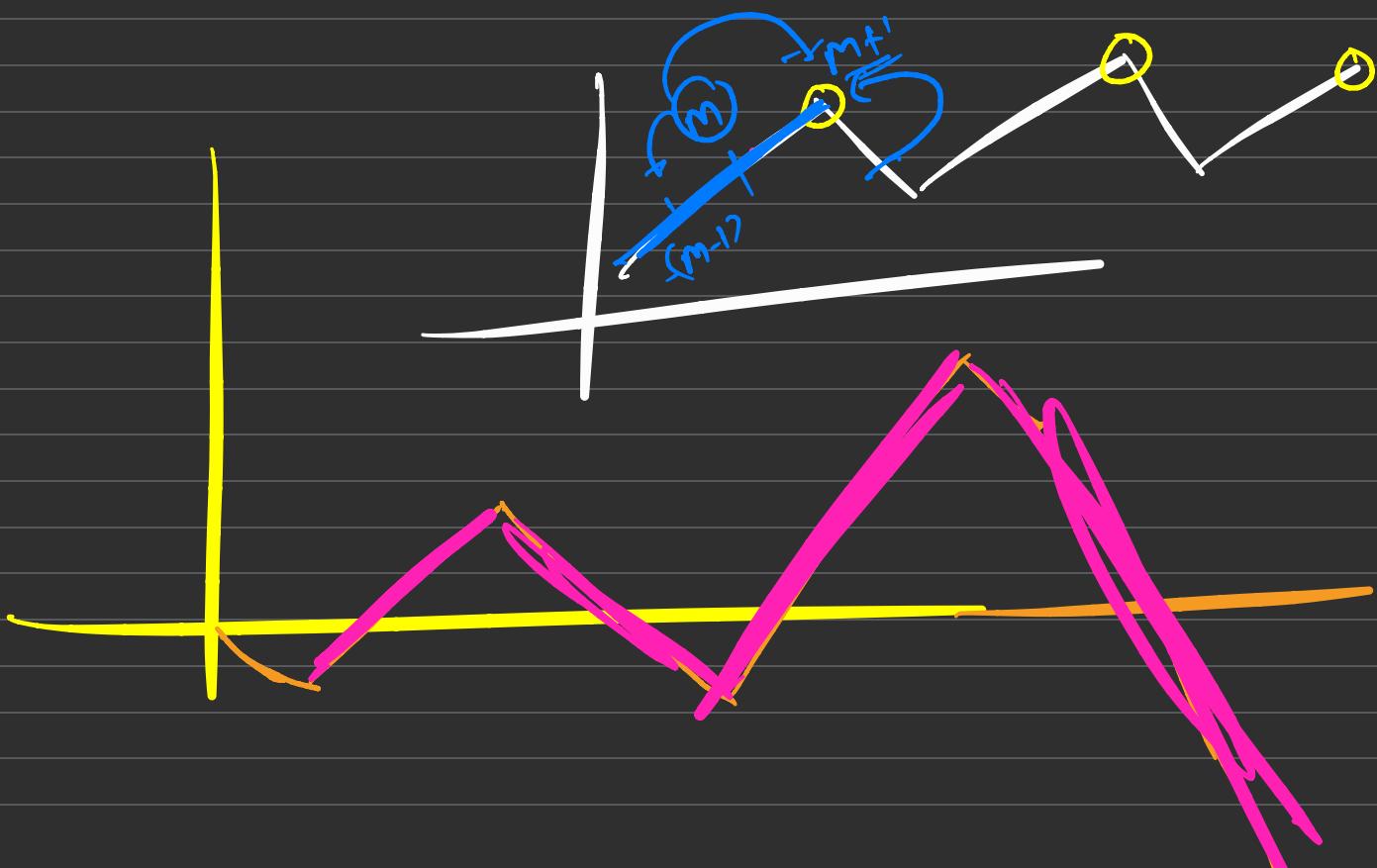
```

    { if (n == 1) return 0
      if (a(0) > a(1)) return 0
      if (a(n-1) > a(n-2)) return n-1
      int s=1, e=n-2
    }
  
```

```

while (s <= e)
{
  mid = (s+e)/2
  if (a(m) > a(mid-1) && a(m) > a(m+1))
    return mid
  if (a(m) > a(m-1))
    s = m + 1
  else
    e = m - 1
}
  
```

-2 18 -16 25 24 5 -1 -2 -15



$O(n)$

for($i=1 : i < n-1 : i + \epsilon$)

{ if ($a(i) > a(i-1) \& \&$
 $a(i) > a(i+1)$)

return i

}

137. Single Number II

Solved

Medium

Topics

Companies

Re-do

Given an integer array `nums` where every element appears **three times** except for one, which appears **exactly once**. *Find the single element and return it.*

You must implement a solution with a linear runtime complexity and use only constant extra space.

Example 1:

Input: `nums = [2,2,3,2]`

Output: 3

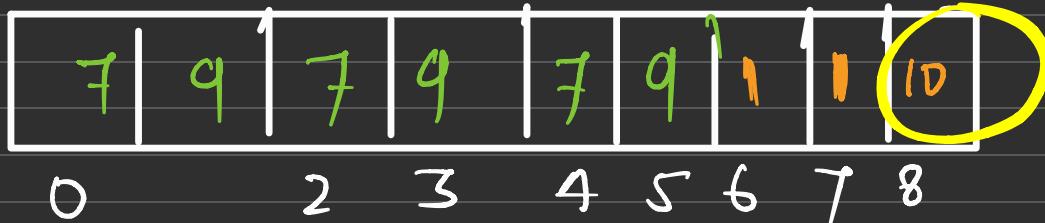
Example 2:

Input: `nums = [0,1,0,1,0,1,99]`

Output: 99

$N=10$

11



B17



Array Leaders



Difficulty: **Easy**

Accuracy: **29.94%**

Submissions: **976K+**

Points: **2**

Average Time: **15m**

You are given an array **arr** of positive integers. Your task is to find all the leaders in the array. An element is considered a leader if it is greater than or equal to all elements to its right. The rightmost element is always a leader.

Examples:

Input: arr = [16, 17, 4, 3, 5, 2]

Output: [17, 5, 2]

Explanation: Note that there is nothing greater on the right side of 17, 5 and, 2.

Input: arr = [10, 4, 2, 4, 1]

Output: [10, 4, 4, 1]

Explanation: Note that both of the 4s are in output, as to be a leader an equal element is also allowed on the right. side

Input: arr = [5, 10, 20, 40]

Output: [40]

Explanation: When an array is sorted in increasing order, only the rightmost element is leader.

16	17	4	3	5	2
0	1	2	3	4	5

~~$i = 1$~~

~~$j = 2$~~

~~Flag = 0~~

vector

ArrayList

2	5	17	

$O(n^2)$

```

for (j = N-2 : j ≥ 0 : j--)
    int flag = 0
    for (j = i+1 : j < N : j++)
        if (a(i) < a(j))
            flag = 1
            break
    if (flag == 0)
        ans. pb(a(i))
    }
}

```

16	17	4	3	5	.	2
0	1	2	3	4	5	

ans.add($a(n-1)$)

$\max = a(n-1)$

for ($i=n-2 : i \geq 0 : i--$)

i if ($a(i) > \max$)

\uparrow ans.add($a(i)$)

}

$\maxE = \max(\maxE, a(i))$

}

Given a **sorted** array, **arr[]** and a number **target**, you need to find the number of occurrences of **target** in **arr[]**.

Examples :

Input: arr[] = [1, 1, 2, 2, 2, 2, 3], target = 2

Output: 4

Explanation: target = 2 occurs 4 times in the given array so the output is 4.

Input: arr[] = [1, 1, 2, 2, 2, 2, 3], target = 4

Output: 0

Explanation: target = 4 is not present in the given array so the output is 0.

Input: arr[] = [8, 9, 10, 12, 12, 12], target = 12

Output: 3

Explanation: target = 12 occurs 3 times in the given array so the output is 3.

- DB
- OS
- OOP
- CPP/Java
- DSA
- Apti
- Project

OSI
Python
CN
AT
CA { SW
HI/WI
Marathi

Aptitude (must do)

- percentage
- Ratio proportion
- SI, CI
- Ages
- Time & work
- Profit loss
- MA
- Probability
- PnL
- Blood Reln
- Syllogism
- SDT
- DI

$N = 9$

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

$mid \% 2 == 0$

$$q(m) == q(m+1)$$

right

$mid \% 2 == 0$

$$q(m) == q(m+1) \quad mid \% 2 == 1$$

right

$mid \% 2 == 1$

$$q(m-1) == q(m)$$

right

mid

left

e

(1,1)	(2,2)	(3,3)	(4)	(5,5)
0	1	2	4	5

██████████

$N = 9$

1	1	2	3	3	4	4	5	5
---	---	---	---	---	---	---	---	---

0 1 2 3 4 5 6 7 8

$s \uparrow$
 s

mid

$t \uparrow$
 t

$s = 0$
 $e = 3$
 $mid = 1$

$(1, 1)$
0 1
2

(2)
2

$(3, 3)$
3 4

$(4, 4)$
5 6

$(5, 5)$
7 8

if ($N == 1$) return $a(0)$
if ($a(0) != a(1)$) return $a(0)$
if ($a(n-1) != a(n-2)$) return $a(n-1)$

$$s = 1, e = n-2$$

while ($s \leq e$)

$$\uparrow m = \frac{s+e}{2}$$

if ($a(m) != a(m+1)$ & & $a(m) != a(m-1)$)
return $a(m)$

if ($mid \% 2 == 0$ && ($a(m) == a(n+1)$ ||
 $mid \% 2 == 1$ && ($a(m-1) == a(n)$))
 $\Rightarrow s = m+1$)

else

$$e = m-1$$