Binary Search Playlist

Video - 27

Leetcode - 1608

Easy

Facebook
Instagram
→ codestorywithMIK

Twitter → cswithMIK

→ codestorywithMIK

"Good Practice Problem"

1608. Special Array With X Elements Greater Than or Equal X

Easy | Topics | Companies | Hint

You are given an array `nums` of non-negative integers. `nums` is considered **special** if there exists a number `x` such that there are **exactly** `x` numbers in `nums` that are **greater than or equal to** `x`.

Notice that `x` **does not** have to be an element in `nums`.

Return `x` *if the array is* ***special****, otherwise, return* `-1`. It can be proven that if `nums` is special, the value for `x` is **unique**.

Example :- nums = {3, 5}

x = 2

output = 2

nums = {0,0}

Output = -1

nums = {0, 4, 3, 0, 4}

Output = 3

Brute Force

nums = {0, 4, 3, 0, 4} , n = 5

$0 <= x <= n$: i

for(x = 0; x <= n; x++) { → n

count = findgreaterEqual(nums, x); //O(n)

if (count == x) return x;

return -1;

T.C = O(n²)
S.C = O(1).

Approach-1

Sort + BSearch

nums = {0, 4, 3, 0, 4}

Sort
O(nlogn)

nums = {0, 0, 3, 4, 4}

n = 5

>= x → Lower Bound

~~for (x = 0; x <= n; x++)~~ → O(n)

int idx = lower_bound(begin(nums), end(nums), x) - begin(nums);

O(logn)

count = n - idx;

if (count == x) return x;

T.C = O(nlogn)
S.C = O(1)

return -1;

Approach-2

idx

(sorted) nums = { 0, 0, 3, 4, 4 }, n = 5

x_l = 3

x_r = 3

mid_x = 3 + (3-3)/2 = 3

```
count = n - idx ; // 5-2 = 3
if (count > mid-x) {
    x_l = mid-x + 1;
} else if (count < mid-x) {
    x_r = mid-x - 1;
} else {
    return mid-x;
}
}
```

~~for~~ → B.Sear.

T.C = O(log(n) * log(n)) + n log(n).

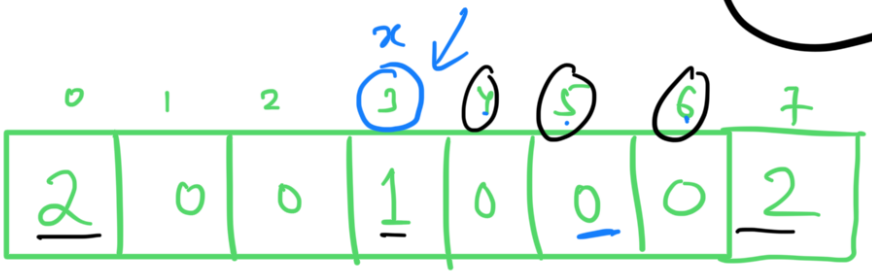
S.C = O(1).

Approach-3

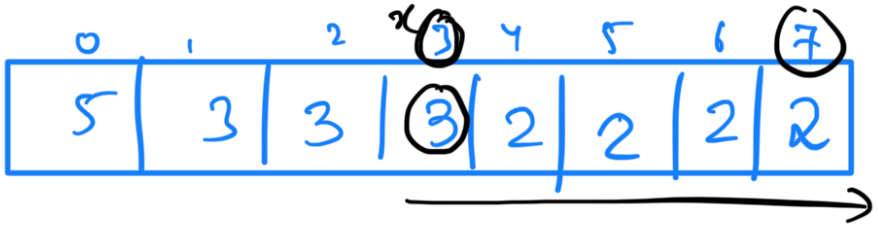
0 <= x <= n

nums = {0, 7, 3, 0, 7}

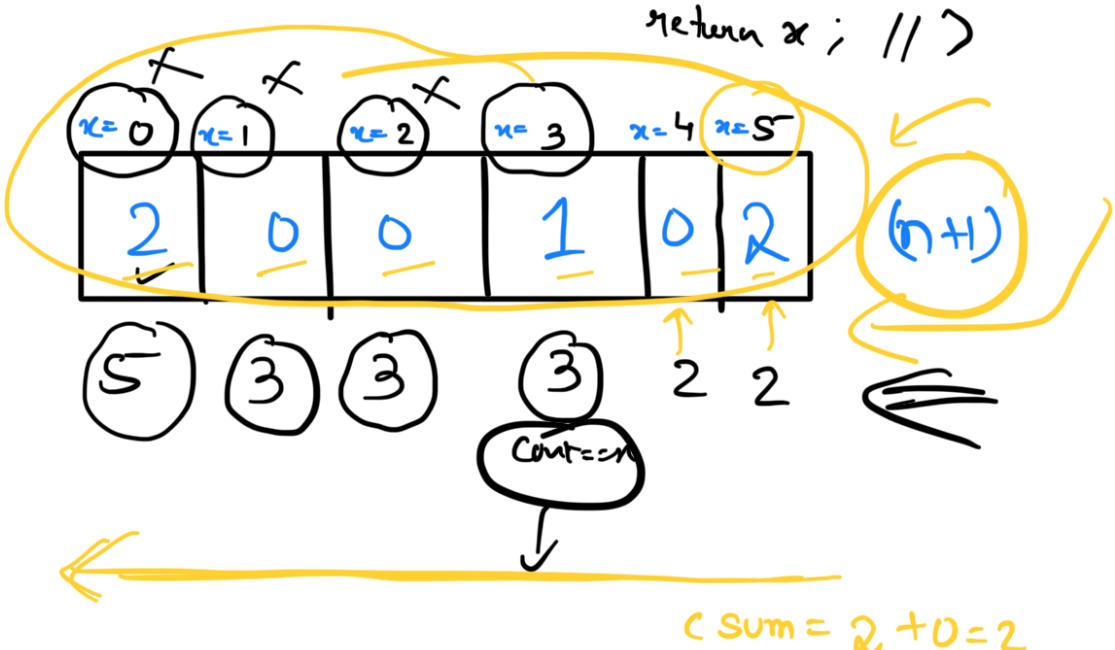
n = 5



0 <= x <= n



return x; // 3



0 <= x <= n

T.C = O(n).

S.C = O(n)