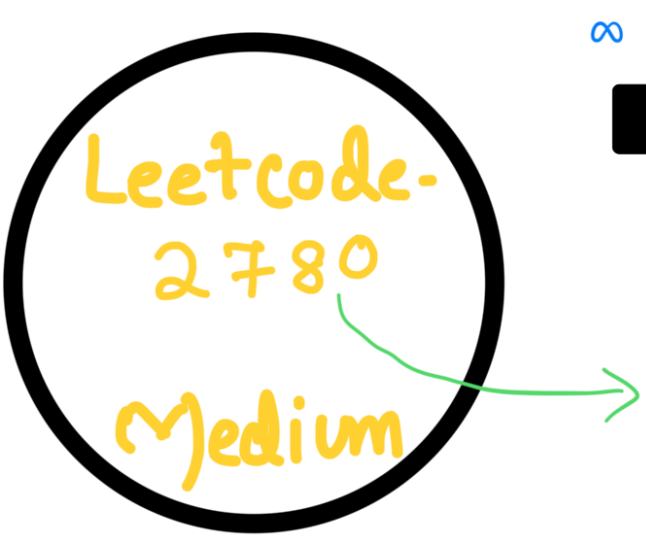
Hash Map/Set

video - 17



codestorywithmik

CSwithMIK

codestorywithMIK

+ Tech ...

Motivation :-

The difference between

ORDINARY & EXTRAORDINARY

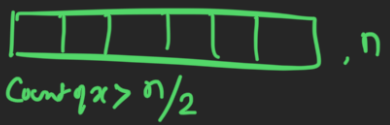
is

that little "EXTRA".

It's the extra effort, extra hours, and extra persistence that turn dream into reality.

MIK...

2780. Minimum Index of a Valid Split



Medium | Topics | Companies | Hint

An element `x` of an integer array `arr` of length `m` is **dominant** if **more than half** the elements of `arr` have a value of `x`.

You are given a **0-indexed** integer array `nums` of length `n` with one **dominant** element.

You can split `nums` at an index `i` into two arrays `nums[0, ..., i]` and `nums[i + 1, ..., n - 1]`, but the split is only **valid** if:

- `0 <= i < n - 1`
- `nums[0, ..., i]`, and `nums[i + 1, ..., n - 1]` have the same dominant element.

Here, `nums[i, ..., j]` denotes the subarray of `nums` starting at index `i` and ending at index `j`, both ends being inclusive. Particularly, if `j < i` then `nums[i, ..., j]` denotes an empty subarray.

Return *the **minimum** index of a **valid split***. If no valid split exists, return `-1`.

Example: arr = [1, 2, 2, 2] (indices 0 1 2 3)

Output = 2

n1 = 3

{1, 2, 2} → 2

Count of 2 - (2) n1/2

2 > 3/2

{2} → 2

n2 = 1

2

count of 2 = 1

1 > 1/2

1 > 0

Thought Process

Brute Force ...

arr = { 1, 2, 2, 2 }
(0, 1, 2, 3)
i

nums[i] = 2

Ultimately freq of elements

map1

1 : 1

2 : 2

n1 = 2

map2

1 : 0

2 : 1

n2 = 4 - 2 - 1 = 1

2 > 2/2

1 > 1/2

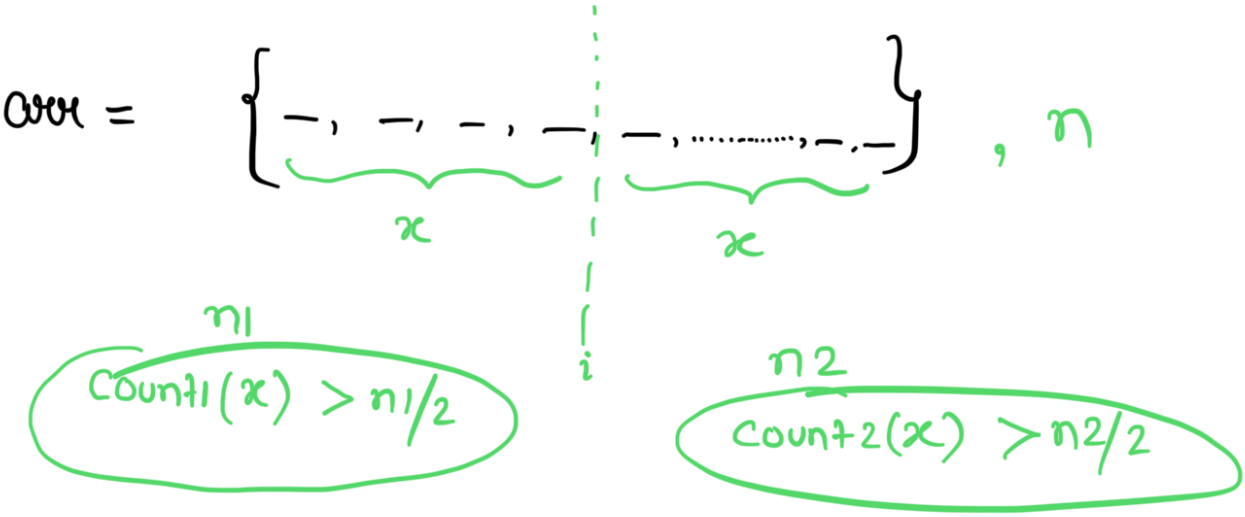
if (map1[nums[i]] > n1/2 && map2[nums[i]] > n2/2) {

return i ;

}

return -1;

Approach - 2

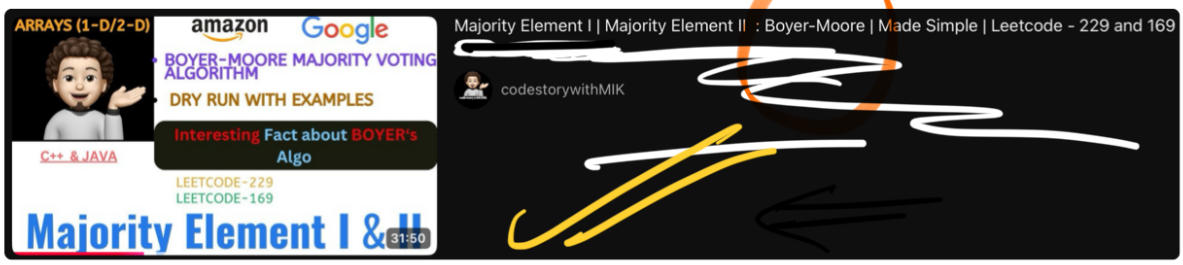


count1(x) + count2(x) > n1/2 + n2/2

Total Count of x > (n1+n2)/2

Total Count of x > n/2

Majority Element.



① Find Majority Element.

② This will be dominant in left & right subarrays of a valid split.

arr: { 1, 2, 2, | 2 } → maj = 2, countMaj = 3

n1 = i, n2

Count = 2;

remai = 3 − 2 = 1

2 > 3/2 1 > 1/2

if (count > n1/2 && remCount > n2/2) {

return i; // 2

}

{ 1, 2, 2, 2 } , n = 4

count = 2

Boore Moyre Algo.

freq(2) > n/2

3 > 4/2 .