

2780. Minimum Index of a Valid Split

Medium

Topics

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:

- $0 \leq i < n - 1$
- `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 1:

Input: `nums = [1,2,2,2]`

Output: 2

Explanation: We can split the array at index 2 to obtain arrays `[1,2,2]` and `[2]`.

In array `[1,2,2]`, element 2 is dominant since it occurs twice in the array and $2 * 2 > 3$.

In array `[2]`, element 2 is dominant since it occurs once in the array and $1 * 2 > 1$.

Both `[1,2,2]` and `[2]` have the same dominant element as `nums`, so this is a valid split.

It can be shown that index 2 is the minimum index of a valid split.

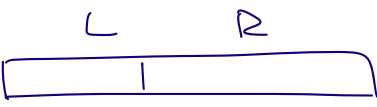
```
1 class Solution:
2     def minimumIndex(self, nums: List[int]) -> int:
3         def findDom(arr: List[int]):
4             n = len(arr)
5             freq = Counter(arr)
6             sorted_freq = sorted(freq.items(), key=lambda x: x[1], reverse=True)
7             if sorted_freq[0][1] > (n // 2):
8                 return sorted_freq[0][0]
9
10            return -1
11
12        for i in range(1, len(nums)):
13            a, b = findDom(nums[:i-1]), findDom(nums[i:])
14            if a == b:
15                print(a, i)
16                return i
17
18        return -1
```

1st idea :

① Index out of range

② inefficient

2nd idea :

nums: 

if majority of L (ML) is equal to MR, then this char must be majority of nums. and the majority can exist at most 1.

So - find the char first, and then scan through.

```
1 class Solution:
2     def minimumIndex(self, nums: List[int]) -> int:
3         freq = Counter(nums)
4         sorted_freq = sorted(freq.items(), key=lambda x: x[1], reverse=True)
5         c, f = sorted_freq[0]
6         if f <= len(nums) // 2:
7             return -1
8
9         count = 0
10        n = len(nums)
11        for i in range(n):
12            if c == nums[i]:
13                count += 1
14
15            if count * 2 > (i + 1) and (f - count) * 2 > (n - i - 1):
16                return i
17
18        return -1
19
20
```