Problem 2826: Sorting Three Groups

Problem Information

Difficulty: Medium
Acceptance Rate: 42.55%
Paid Only: No
Tags: Array, Binary Search, Dynamic Programming

Problem Description

You are given an integer array `nums`. Each element in `nums` is 1, 2 or 3. In each operation, you can remove an element from `nums`. Return the **minimum** number of operations to make `nums` **non-decreasing**.

Example 1:

Input: nums = [2,1,3,2,1]

Output: 3

Explanation:

One of the optimal solutions is to remove `nums[0]`, `nums[2]` and `nums[3]`.

Example 2:

Input: nums = [1,3,2,1,3,3]

Output: 2

Explanation:

One of the optimal solutions is to remove `nums[1]` and `nums[2]`.

Example 3:

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

Output: 0

Explanation:

`nums` is already non-decreasing.

Constraints:

* `1 <= nums.length <= 100` * `1 <= nums[i] <= 3`

Follow-up: Can you come up with an algorithm that runs in `O(n)` time complexity?

Code Snippets

C++:

```
class Solution {
public:
int minimumOperations(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public int minimumOperations(List<Integer> nums) {

}
}
```

Python3:

```
class Solution:
def minimumOperations(self, nums: List[int]) -> int:
```