Problem 3740: Minimum Distance Between Three Equal Elements I

Problem Information

Difficulty: Easy
Acceptance Rate: 59.98%
Paid Only: No
Tags: Array, Hash Table

Problem Description

You are given an integer array `nums`.

A tuple `(i, j, k)` of 3 **distinct** indices is **good** if `nums[i] == nums[j] == nums[k]`.

The **distance** of a **good** tuple is `abs(i - j) + abs(j - k) + abs(k - i)`, where `abs(x)` denotes the **absolute value** of `x`.

Return an integer denoting the **minimum** possible **distance** of a **good** tuple. If no **good** tuples exist, return `-1`.

Example 1:

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

Output: 6

Explanation:

The minimum distance is achieved by the good tuple `(0, 2, 3)`.

`(0, 2, 3)` is a good tuple because `nums[0] == nums[2] == nums[3] == 1`. Its distance is `abs(0 - 2) + abs(2 - 3) + abs(3 - 0) = 2 + 1 + 3 = 6`.

Example 2:

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

Output: 8

Explanation:

The minimum distance is achieved by the good tuple `(2, 4, 6)`.

`(2, 4, 6)` is a good tuple because `nums[2] == nums[4] == nums[6] == 2`. Its distance is `abs(2 - 4) + abs(4 - 6) + abs(6 - 2) = 2 + 2 + 4 = 8`.

Example 3:

Input: nums = [1]

Output: -1

Explanation:

There are no good tuples. Therefore, the answer is -1.

Constraints:

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

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public int minimumDistance(int[] nums) {

}
}
```

Python3:

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