Problem 3431: Minimum Unlocked Indices to Sort Nums

Problem Information

Difficulty: Medium
Acceptance Rate: 60.37%
Paid Only: Yes
Tags: Array, Hash Table

Problem Description

You are given an array `nums` consisting of integers between 1 and 3, and a **binary** array `locked` of the same size.

We consider `nums` **sortable** if it can be sorted using adjacent swaps, where a swap between two indices `i` and `i + 1` is allowed if `nums[i] - nums[i + 1] == 1` and `locked[i] == 0`.

In one operation, you can unlock any index `i` by setting `locked[i]` to 0.

Return the **minimum** number of operations needed to make `nums` **sortable**. If it is not possible to make `nums` sortable, return -1.

Example 1:

Input: nums = [1,2,1,2,3,2], locked = [1,0,1,1,0,1]

Output: 0

Explanation:

We can sort `nums` using the following swaps:

* swap indices 1 with 2 * swap indices 4 with 5

So, there is no need to unlock any index.

Example 2:

Input: nums = [1,2,1,1,3,2,2], locked = [1,0,1,1,0,1,0]

Output: 2

Explanation:

If we unlock indices 2 and 5, we can sort `nums` using the following swaps:

* swap indices 1 with 2 * swap indices 2 with 3 * swap indices 4 with 5 * swap indices 5 with 6

Example 3:

Input: nums = [1,2,1,2,3,2,1], locked = [0,0,0,0,0,0,0]

Output: -1

Explanation:

Even if all indices are unlocked, it can be shown that `nums` is not sortable.

Constraints:

* `1 <= nums.length <= 105` * `1 <= nums[i] <= 3` * `locked.length == nums.length` * `0 <= locked[i] <= 1`

Code Snippets

C++:

```
class Solution {
public:
int minUnlockedIndices(vector<int>& nums, vector<int>& locked) {

}
};
```

Java:

```
class Solution {
public int minUnlockedIndices(int[] nums, int[] locked) {

}
}
```

Python3:

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