Problem 1764: Form Array by Concatenating Subarrays of Another Array

Problem Information

Difficulty: Medium
Acceptance Rate: 54.39%
Paid Only: No
Tags: Array, Two Pointers, Greedy, String Matching

Problem Description

You are given a 2D integer array `groups` of length `n`. You are also given an integer array `nums`.

You are asked if you can choose `n` **disjoint** subarrays from the array `nums` such that the `ith` subarray is equal to `groups[i]` (**0-indexed**), and if `i > 0`, the `(i-1)th` subarray appears **before** the `ith` subarray in `nums` (i.e. the subarrays must be in the same order as `groups`).

Return `true` _if you can do this task, and_ `false` _otherwise_.

Note that the subarrays are **disjoint** if and only if there is no index `k` such that `nums[k]` belongs to more than one subarray. A subarray is a contiguous sequence of elements within an array.

Example 1:

Input: groups = [[1,-1,-1],[3,-2,0]], nums = [1,-1,0,1,-1,-1,3,-2,0] **Output:** true **Explanation:** You can choose the 0th subarray as [1,-1,0,_**1,-1,-1**_ ,3,-2,0] and the 1st one as [1,-1,0,1,-1,-1,_**3,-2,0**_]. These subarrays are disjoint as they share no common nums[k] element.

Example 2:

Input: groups = [[10,-2],[1,2,3,4]], nums = [1,2,3,4,10,-2] **Output:** false **Explanation:** Note that choosing the subarrays [_**1,2,3,4**_ ,10,-2] and [1,2,3,4,_**10,-2**_] is incorrect

because they are not in the same order as in groups. [10,-2] must come before [1,2,3,4].

Example 3:

Input: groups = [[1,2,3],[3,4]], nums = [7,7,1,2,3,4,7,7] **Output:** false **Explanation:** Note that choosing the subarrays [7,7,_**1,2,3**_ ,4,7,7] and [7,7,1,2,_**3,4**_ ,7,7] is invalid because they are not disjoint. They share a common elements nums[4] (0-indexed).

Constraints:

* `groups.length == n` * `1 <= n <= 103` * `1 <= groups[i].length, sum(groups[i].length) <= 103` * `1 <= nums.length <= 103` * `-107 <= groups[i][j], nums[k] <= 107`

Code Snippets

C++:

```
class Solution {
public:
bool canChoose(vector<vector<int>>& groups, vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public boolean canChoose(int[][] groups, int[] nums) {

}
}
```

Python3:

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