Problem 2369: Check if There is a Valid Partition For The Array

Problem Information

Difficulty: Medium
Acceptance Rate: 52.13%
Paid Only: No
Tags: Array, Dynamic Programming

Problem Description

You are given a **0-indexed** integer array `nums`. You have to partition the array into one or more **contiguous** subarrays.

We call a partition of the array **valid** if each of the obtained subarrays satisfies **one** of the following conditions:

1. The subarray consists of **exactly** `2,` equal elements. For example, the subarray `[2,2]` is good. 2. The subarray consists of **exactly** `3,` equal elements. For example, the subarray `[4,4,4]` is good. 3. The subarray consists of **exactly** `3` consecutive increasing elements, that is, the difference between adjacent elements is `1`. For example, the subarray `[3,4,5]` is good, but the subarray `[1,3,5]` is not.

Return `true` _if the array has**at least** one valid partition_. Otherwise, return `false`.

Example 1:

Input: nums = [4,4,4,5,6] **Output:** true **Explanation:** The array can be partitioned into the subarrays [4,4] and [4,5,6]. This partition is valid, so we return true.

Example 2:

Input: nums = [1,1,1,2] **Output:** false **Explanation:** There is no valid partition for this array.

Constraints:

* `2 <= nums.length <= 105` * `1 <= nums[i] <= 106`

Code Snippets

C++:

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

}
};
```

Java:

```
class Solution {
public boolean validPartition(int[] nums) {

}
}
```

Python3:

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