Problem 2464: Minimum Subarrays in a Valid Split

Problem Information

Difficulty: Medium
Acceptance Rate: 64.86%
Paid Only: Yes
Tags: Array, Math, Dynamic Programming, Number Theory

Problem Description

You are given an integer array `nums`.

Splitting of an integer array `nums` into **subarrays** is **valid** if:

* the _greatest common divisor_ of the first and last elements of each subarray is **greater** than `1`, and * each element of `nums` belongs to exactly one subarray.

Return _the**minimum** number of subarrays in a **valid** subarray splitting of_ `nums`. If a valid subarray splitting is not possible, return `-1`.

Note that:

* The **greatest common divisor** of two numbers is the largest positive integer that evenly divides both numbers. * A **subarray** is a contiguous non-empty part of an array.

Example 1:

Input: nums = [2,6,3,4,3] **Output:** 2 **Explanation:** We can create a valid split in the following way: [2,6] | [3,4,3]. - The starting element of the 1st subarray is 2 and the ending is 6. Their greatest common divisor is 2, which is greater than 1. - The starting element of the 2nd subarray is 3 and the ending is 3. Their greatest common divisor is 3, which is greater than 1. It can be proved that 2 is the minimum number of subarrays that we can obtain in a valid split.

Example 2:

Input: nums = [3,5] **Output:** 2 **Explanation:** We can create a valid split in the following way: [3] | [5]. - The starting element of the 1st subarray is 3 and the ending is 3. Their greatest common divisor is 3, which is greater than 1. - The starting element of the 2nd subarray is 5 and the ending is 5. Their greatest common divisor is 5, which is greater than 1. It can be proved that 2 is the minimum number of subarrays that we can obtain in a valid split.

Example 3:

Input: nums = [1,2,1] **Output:** -1 **Explanation:** It is impossible to create valid split.

Constraints:

* `1 <= nums.length <= 1000` * `1 <= nums[i] <= 105`

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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