

Problem 2436: Minimum Split Into Subarrays With GCD Greater Than One

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

You are given an array

nums

consisting of positive integers.

Split the array into

one or more

disjoint subarrays such that:

Each element of the array belongs to

exactly one

subarray, and

The

GCD

of the elements of each subarray is strictly greater than

.

Return

the minimum number of subarrays that can be obtained after the split

.

Note

that:

The

GCD

of a subarray is the largest positive integer that evenly divides all the elements of the subarray.

A

subarray

is a contiguous part of the array.

Example 1:

Input:

nums = [12,6,3,14,8]

Output:

2

Explanation:

We can split the array into the subarrays: [12,6,3] and [14,8]. - The GCD of 12, 6 and 3 is 3, which is strictly greater than 1. - The GCD of 14 and 8 is 2, which is strictly greater than 1. It

can be shown that splitting the array into one subarray will make the GCD = 1.

Example 2:

Input:

nums = [4,12,6,14]

Output:

1

Explanation:

We can split the array into only one subarray, which is the whole array.

Constraints:

$1 \leq \text{nums.length} \leq 2000$

$2 \leq \text{nums}[i] \leq 10$

9

Code Snippets

C++:

```
class Solution {
public:
    int minimumSplits(vector<int>& nums) {
        }
};
```

Java:

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

```
}
```

```
}
```

Python3:

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

Python:

```
class Solution(object):  
    def minimumSplits(self, nums):  
        """  
        :type nums: List[int]  
        :rtype: int  
        """
```

JavaScript:

```
/**  
 * @param {number[]} nums  
 * @return {number}  
 */  
var minimumSplits = function(nums) {  
  
};
```

TypeScript:

```
function minimumSplits(nums: number[]): number {  
  
};
```

C#:

```
public class Solution {  
    public int MinimumSplits(int[] nums) {  
  
    }  
}
```

C:

```
int minimumSplits(int* nums, int numsSize) {  
  
}
```

Go:

```
func minimumSplits(nums []int) int {  
  
}
```

Kotlin:

```
class Solution {  
    fun minimumSplits(nums: IntArray): Int {  
  
    }  
}
```

Swift:

```
class Solution {  
    func minimumSplits(_ nums: [Int]) -> Int {  
  
    }  
}
```

Rust:

```
impl Solution {  
    pub fn minimum_splits(nums: Vec<i32>) -> i32 {  
  
    }  
}
```

Ruby:

```
# @param {Integer[]} nums  
# @return {Integer}  
def minimum_splits(nums)  
  
end
```

PHP:

```
class Solution {  
  
    /**  
     * @param Integer[] $nums  
     * @return Integer  
     */  
    function minimumSplits($nums) {  
  
    }  
}
```

Dart:

```
class Solution {  
int minimumSplits(List<int> nums) {  
  
}  
}
```

Scala:

```
object Solution {  
def minimumSplits(nums: Array[Int]): Int = {  
  
}  
}
```

Elixir:

```
defmodule Solution do  
@spec minimum_splits(nums :: [integer]) :: integer  
def minimum_splits(nums) do  
  
end  
end
```

Erlang:

```
-spec minimum_splits(Nums :: [integer()]) -> integer().  
minimum_splits(Nums) ->  
.
```

Racket:

```
(define/contract (minimum-splits nums)
  (-> (listof exact-integer?) exact-integer?))
```

Solutions

C++ Solution:

```
/*
 * Problem: Minimum Split Into Subarrays With GCD Greater Than One
 * Difficulty: Medium
 * Tags: array, dp, greedy, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) or O(n * m) for DP table
 */

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

    }
};
```

Java Solution:

```
/**
 * Problem: Minimum Split Into Subarrays With GCD Greater Than One
 * Difficulty: Medium
 * Tags: array, dp, greedy, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) or O(n * m) for DP table
 */

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

```
}
```

```
}
```

Python3 Solution:

```
"""
Problem: Minimum Split Into Subarrays With GCD Greater Than One
Difficulty: Medium
Tags: array, dp, greedy, math
```

```
Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
```

```
"""
```

```
class Solution:
    def minimumSplits(self, nums: List[int]) -> int:
        # TODO: Implement optimized solution
        pass
```

Python Solution:

```
class Solution(object):
    def minimumSplits(self, nums):
        """
        :type nums: List[int]
        :rtype: int
        """
```

JavaScript Solution:

```
/**
 * Problem: Minimum Split Into Subarrays With GCD Greater Than One
 * Difficulty: Medium
 * Tags: array, dp, greedy, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) or O(n * m) for DP table
 */
```

```

/**
 * @param {number[]} nums
 * @return {number}
 */
var minimumSplits = function(nums) {

};

```

TypeScript Solution:

```

/**
 * Problem: Minimum Split Into Subarrays With GCD Greater Than One
 * Difficulty: Medium
 * Tags: array, dp, greedy, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) or O(n * m) for DP table
 */

function minimumSplits(nums: number[]): number {
}

;

```

C# Solution:

```

/*
 * Problem: Minimum Split Into Subarrays With GCD Greater Than One
 * Difficulty: Medium
 * Tags: array, dp, greedy, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) or O(n * m) for DP table
 */

public class Solution {
    public int MinimumSplits(int[] nums) {
    }
}
```

```
}
```

C Solution:

```
/*
 * Problem: Minimum Split Into Subarrays With GCD Greater Than One
 * Difficulty: Medium
 * Tags: array, dp, greedy, math
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) or O(n * m) for DP table
 */

int minimumSplits(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Minimum Split Into Subarrays With GCD Greater Than One
// Difficulty: Medium
// Tags: array, dp, greedy, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func minimumSplits(nums []int) int {

}
```

Kotlin Solution:

```
class Solution {
    fun minimumSplits(nums: IntArray): Int {
        }

    }
}
```

Swift Solution:

```
class Solution {  
func minimumSplits(_ nums: [Int]) -> Int {  
  
}  
}  
}
```

Rust Solution:

```
// Problem: Minimum Split Into Subarrays With GCD Greater Than One  
// Difficulty: Medium  
// Tags: array, dp, greedy, math  
//  
// Approach: Use two pointers or sliding window technique  
// Time Complexity: O(n) or O(n log n)  
// Space Complexity: O(n) or O(n * m) for DP table  
  
impl Solution {  
pub fn minimum_splits(nums: Vec<i32>) -> i32 {  
  
}  
}
```

Ruby Solution:

```
# @param {Integer[]} nums  
# @return {Integer}  
def minimum_splits(nums)  
  
end
```

PHP Solution:

```
class Solution {  
  
/**  
 * @param Integer[] $nums  
 * @return Integer  
 */  
function minimumSplits($nums) {  
  
}  
}
```

Dart Solution:

```
class Solution {  
    int minimumSplits(List<int> nums) {  
  
    }  
}
```

Scala Solution:

```
object Solution {  
    def minimumSplits(nums: Array[Int]): Int = {  
  
    }  
}
```

Elixir Solution:

```
defmodule Solution do  
  @spec minimum_splits(list :: [integer]) :: integer  
  def minimum_splits(list) do  
  
  end  
end
```

Erlang Solution:

```
-spec minimum_splits(list :: [integer()]) -> integer().  
minimum_splits(list) ->  
.
```

Racket Solution:

```
(define/contract (minimum-splits list)  
  (-> (listof exact-integer?) exact-integer?)  
)
```