Problem 2584: Split the Array to Make Coprime Products

Problem Information

Difficulty: Hard
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

You are given a

0-indexed

integer array

nums

of length

n

.

A

split

at an index

i

where

0 <= i <= n - 2

is called

valid

if the product of the first

i + 1

elements and the product of the remaining elements are coprime.

For example, if

nums = [2, 3, 3]

, then a split at the index

i = 0

is valid because

2

and

9

are coprime, while a split at the index

i = 1

is not valid because

6

and

3

are not coprime. A split at the index

i = 2

is not valid because

i == n - 1

.

Return

the smallest index

i

at which the array can be split validly or

-1

if there is no such split

.

Two values

val1

and

val2

are coprime if

gcd(val1, val2) == 1

where

gcd(val1, val2)

is the greatest common divisor of

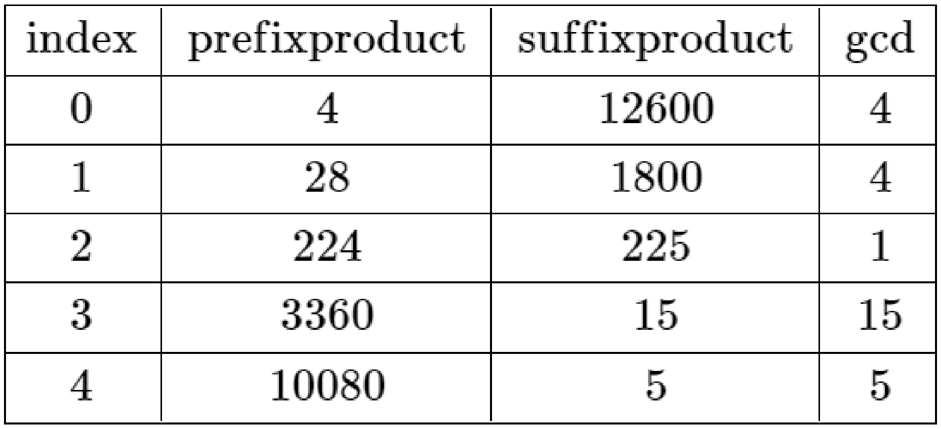
val1

and

val2

.

Example 1:



index	prefixproduct	suffixproduct	gcd
0	4	12600	4
1	28	1800	4
2	224	225	1
3	3360	15	15
4	10080	5	5

Input:

nums = [4,7,8,15,3,5]

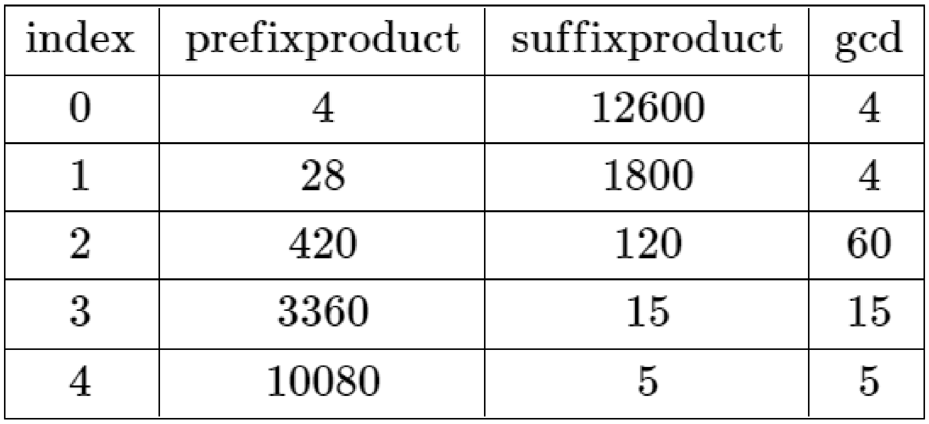
Output:

2

Explanation:

The table above shows the values of the product of the first i + 1 elements, the remaining elements, and their gcd at each index i. The only valid split is at index 2.

Example 2:



index	prefixproduct	suffixproduct	gcd
0	4	12600	4
1	28	1800	4
2	420	120	60
3	3360	15	15
4	10080	5	5

Input:

nums = [4,7,15,8,3,5]

Output:

-1

Explanation:

The table above shows the values of the product of the first i + 1 elements, the remaining elements, and their gcd at each index i. There is no valid split.

Constraints:

n == nums.length

1 <= n <= 10

4

1 <= nums[i] <= 10

6

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function findValidSplit(nums: number[]): number {

};
```

C#:

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

}
}
```

C:

```
int findValidSplit(int* nums, int numsSize) {

}
```

Go:

```
func findValidSplit(nums []int) int {

}
```

Kotlin:

```
class Solution {
fun findValidSplit(nums: IntArray): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
*/
function findValidSplit($nums) {

}
}
```

Dart:

```
class Solution {
int findValidSplit(List<int> nums) {

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (find-valid-split nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Split the Array to Make Coprime Products
* Difficulty: Hard
* Tags: array, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Split the Array to Make Coprime Products
* Difficulty: Hard
* Tags: array, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

class Solution {
public int findValidSplit(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Split the Array to Make Coprime Products
Difficulty: Hard
Tags: array, math, hash

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

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

Python Solution:

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

```
"""
```

JavaScript Solution:

```
/**
* Problem: Split the Array to Make Coprime Products
* Difficulty: Hard
* Tags: array, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

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

};
```

TypeScript Solution:

```
/**
* Problem: Split the Array to Make Coprime Products
* Difficulty: Hard
* Tags: array, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

function findValidSplit(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Split the Array to Make Coprime Products
* Difficulty: Hard
* Tags: array, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int FindValidSplit(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Split the Array to Make Coprime Products
* Difficulty: Hard
* Tags: array, math, hash
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/

int findValidSplit(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Split the Array to Make Coprime Products
// Difficulty: Hard
// Tags: array, math, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map
```

```
func findValidSplit(nums []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun findValidSplit(nums: IntArray): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Split the Array to Make Coprime Products
// Difficulty: Hard
// Tags: array, math, hash
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
pub fn find_valid_split(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

```
end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
*/
function findValidSplit($nums) {

}
}
```

Dart Solution:

```
class Solution {
int findValidSplit(List<int> nums) {

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (find-valid-split nums)
  (-> (listof exact-integer?) exact-integer?)
  )
```