Problem 1819: Number of Different Subsequences GCDs

Problem Information

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

Problem Description

You are given an array

nums

that consists of positive integers.

The

GCD

of a sequence of numbers is defined as the greatest integer that divides

all

the numbers in the sequence evenly.

For example, the GCD of the sequence

[4,6,16]

is

2

.

A

subsequence

of an array is a sequence that can be formed by removing some elements (possibly none) of the array.

For example,

[2,5,10]

is a subsequence of

[1,2,1,

2

,4,1,

5

,

10

]

.

Return

the

number

of

different

GCDs among all

non-empty

subsequences of

nums

.

Example 1:

Subsequence	GCD
[6]	6
[10]	10
[3]	3
[6,10]	2
[6,3]	3
[10,3]	1
[6,10,3]	1

Input:

nums = [6,10,3]

Output:

5

Explanation:

The figure shows all the non-empty subsequences and their GCDs. The different GCDs are 6, 10, 3, 2, and 1.

Example 2:

Input:

nums = [5,15,40,5,6]

Output:

7

Constraints:

1 <= nums.length <= 10

5

1 <= nums[i] <= 2 * 10

5

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (count-different-subsequence-gc-ds nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Different Subsequences GCDs
* Difficulty: Hard
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
};
```

Java Solution:

```
/**
* Problem: Number of Different Subsequences GCDs
* Difficulty: Hard
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

```
}
}
```

Python3 Solution:

```
"""
Problem: Number of Different Subsequences GCDs
Difficulty: Hard
Tags: array, math

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Number of Different Subsequences GCDs
* Difficulty: Hard
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/
```

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

};
```

TypeScript Solution:

```
/**
* Problem: Number of Different Subsequences GCDs
* Difficulty: Hard
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

};
```

C# Solution:

```
/*
* Problem: Number of Different Subsequences GCDs
* Difficulty: Hard
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

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

}
```

```
}
```

C Solution:

```
/*
* Problem: Number of Different Subsequences GCDs
* Difficulty: Hard
* Tags: array, math
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int countDifferentSubsequenceGCDs(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Number of Different Subsequences GCDs
// Difficulty: Hard
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func countDifferentSubsequenceGCDs(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Number of Different Subsequences GCDs
// Difficulty: Hard
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

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

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (count-different-subsequence-gc-ds nums)
(-> (listof exact-integer?) exact-integer?)
)
```