Problem 2501: Longest Square Streak in an Array

Problem Information

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

Problem Description

You are given an integer array

nums

. A subsequence of

nums

is called a

square streak

if:

The length of the subsequence is at least

2

, and

after

sorting the subsequence, each element (except the first element) is the

square

of the previous number.

Return

the length of the

longest square streak

in

nums

, or return

-1

if there is no

square streak

.

A

subsequence

is an array that can be derived from another array by deleting some or no elements without changing the order of the remaining elements.

Example 1:

Input:

nums = [4,3,6,16,8,2]

Output:

3

Explanation:

Choose the subsequence [4,16,2]. After sorting it, it becomes [2,4,16]. - 4 = 2 * 2. - 16 = 4 * 4. Therefore, [4,16,2] is a square streak. It can be shown that every subsequence of length 4 is not a square streak.

Example 2:

Input:

nums = [2,3,5,6,7]

Output:

-1

Explanation:

There is no square streak in nums so return -1.

Constraints:

2 <= nums.length <= 10

5

2 <= nums[i] <= 10

5

Code Snippets

C++:

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

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (longest-square-streak nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Longest Square Streak in an Array
* Difficulty: Medium
* Tags: array, dp, hash, sort, search
*
* 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 longestSquareStreak(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Longest Square Streak in an Array
* Difficulty: Medium
* Tags: array, dp, hash, sort, search
*
* 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 longestSquareStreak(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Longest Square Streak in an Array
Difficulty: Medium
Tags: array, dp, hash, sort, search

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 longestSquareStreak(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Longest Square Streak in an Array
* Difficulty: Medium
* Tags: array, dp, hash, sort, search
*
* 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 longestSquareStreak = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Longest Square Streak in an Array
* Difficulty: Medium
* Tags: array, dp, hash, sort, search
*
* 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 longestSquareStreak(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Longest Square Streak in an Array
* Difficulty: Medium
* Tags: array, dp, hash, sort, search
*
* 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 LongestSquareStreak(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Longest Square Streak in an Array
* Difficulty: Medium
* Tags: array, dp, hash, sort, search
*
* 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 longestSquareStreak(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Longest Square Streak in an Array
// Difficulty: Medium
// Tags: array, dp, hash, sort, search
//
// 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 longestSquareStreak(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Longest Square Streak in an Array
// Difficulty: Medium
// Tags: array, dp, hash, sort, search
//
// 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 longest_square_streak(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {
```

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (longest-square-streak nums)
(-> (listof exact-integer?) exact-integer?)
)
```