Problem 594: Longest Harmonious Subsequence

Problem Information

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

Problem Description

We define a harmonious array as an array where the difference between its maximum value and its minimum value is

exactly

1

.

Given an integer array

nums

, return the length of its longest harmonious

subsequence

among all its possible subsequences.

Example 1:

Input:

nums = [1,3,2,2,5,2,3,7]

Output:

5

Explanation:

The longest harmonious subsequence is

[3,2,2,2,3]

.

Example 2:

Input:

nums = [1,2,3,4]

Output:

2

Explanation:

The longest harmonious subsequences are

[1,2]

,

[2,3]

, and

[3,4]

, all of which have a length of 2.

Example 3:

Input:

nums = [1,1,1,1]

Output:

0

Explanation:

No harmonic subsequence exists.

Constraints:

1 <= nums.length <= 2 * 10

4

-10

9

<= nums[i] <= 10

9

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Longest Harmonious Subsequence
* Difficulty: Easy
* Tags: array, hash, sort
*
* 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 findLHS(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Longest Harmonious Subsequence
* Difficulty: Easy
* Tags: array, hash, sort
*
* 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 findLHS(int[] nums) {
```

```
}
}
```

Python3 Solution:

```
"""
Problem: Longest Harmonious Subsequence
Difficulty: Easy
Tags: array, hash, sort

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Longest Harmonious Subsequence
* Difficulty: Easy
* Tags: array, hash, sort
*
* 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 findLHS = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Longest Harmonious Subsequence
* Difficulty: Easy
* Tags: array, hash, sort
*
* 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 findLHS(nums: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Longest Harmonious Subsequence
* Difficulty: Easy
* Tags: array, hash, sort
*
* 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 FindLHS(int[] nums) {

}
```

```
}
```

C Solution:

```
/*
* Problem: Longest Harmonious Subsequence
* Difficulty: Easy
* Tags: array, hash, sort
*
* 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 findLHS(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Longest Harmonious Subsequence
// Difficulty: Easy
// Tags: array, hash, sort
//
// 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 findLHS(nums []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Longest Harmonious Subsequence
// Difficulty: Easy
// Tags: array, hash, sort
//
// 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_lhs(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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