

Problem 1027: Longest Arithmetic Subsequence

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

Given an array

nums

of integers, return

the length of the longest arithmetic subsequence in

nums

.

Note

that:

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.

A sequence

seq

is arithmetic if

$\text{seq}[i + 1] - \text{seq}[i]$

are all the same value (for

$0 \leq i < \text{seq.length} - 1$

).

Example 1:

Input:

nums = [3,6,9,12]

Output:

4

Explanation:

The whole array is an arithmetic sequence with steps of length = 3.

Example 2:

Input:

nums = [9,4,7,2,10]

Output:

3

Explanation:

The longest arithmetic subsequence is [4,7,10].

Example 3:

Input:

```
nums = [20,1,15,3,10,5,8]
```

Output:

```
4
```

Explanation:

The longest arithmetic subsequence is [20,15,10,5].

Constraints:

```
2 <= nums.length <= 1000
```

```
0 <= nums[i] <= 500
```

Code Snippets

C++:

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

Java:

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

Python3:

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

Python:

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

JavaScript:

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

TypeScript:

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

C#:

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

C:

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

Go:

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

Kotlin:

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

Swift:

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

Rust:

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

Ruby:

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

PHP:

```
class Solution {  
    /**
```

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (longest-arith-seq-length nums)
  (-> (listof exact-integer?) exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
 * Problem: Longest Arithmetic Subsequence
 * Difficulty: Medium
 * Tags: array, dp, hash, 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 longestArithSeqLength(vector<int>& nums) {

    }
};
```

Java Solution:

```
/**
 * Problem: Longest Arithmetic Subsequence
 * Difficulty: Medium
 * Tags: array, dp, hash, 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 longestArithSeqLength(int[] nums) {

    }
}
```

```
}
```

Python3 Solution:

```
"""
Problem: Longest Arithmetic Subsequence
Difficulty: Medium
Tags: array, dp, hash, 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 longestArithSeqLength(self, nums: List[int]) -> int:
        # TODO: Implement optimized solution
        pass
```

Python Solution:

```
class Solution(object):

    def longestArithSeqLength(self, nums):
        """
:type nums: List[int]
:rtype: int
"""



```

JavaScript Solution:

```
/**
 * Problem: Longest Arithmetic Subsequence
 * Difficulty: Medium
 * Tags: array, dp, hash, 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 longestArithSeqLength = function(nums) {

};

```

TypeScript Solution:

```

/**
 * Problem: Longest Arithmetic Subsequence
 * Difficulty: Medium
 * Tags: array, dp, hash, 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 longestArithSeqLength(nums: number[]): number {
}


```

C# Solution:

```

/*
 * Problem: Longest Arithmetic Subsequence
 * Difficulty: Medium
 * Tags: array, dp, hash, 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 LongestArithSeqLength(int[] nums) {
        }

    }
}
```

C Solution:

```
/*
 * Problem: Longest Arithmetic Subsequence
 * Difficulty: Medium
 * Tags: array, dp, hash, 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 longestArithSeqLength(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Longest Arithmetic Subsequence
// Difficulty: Medium
// Tags: array, dp, hash, 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 longestArithSeqLength(nums []int) int {

}
```

Kotlin Solution:

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

    }
}
```

Swift Solution:

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

```
}
```

```
}
```

Rust Solution:

```
// Problem: Longest Arithmetic Subsequence
// Difficulty: Medium
// Tags: array, dp, hash, 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_arith_seq_length(nums: Vec<i32>) -> i32 {
        }

    }
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

    }
}
```

Dart Solution:

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

Scala Solution:

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

Elixir Solution:

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

Erlang Solution:

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

Racket Solution:

```
(define/contract (longest-arith-seq-length nums)  
  (-> (listof exact-integer?) exact-integer?)  
)
```