

Problem 873: Length of Longest Fibonacci Subsequence

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

A sequence

x

1

, x

2

, ..., x

n

is

Fibonacci-like

if:

$n \geq 3$

x

i

+ x

i+1

== x

i+2

for all

$i + 2 \leq n$

Given a

strictly increasing

array

arr

of positive integers forming a sequence, return

the

length

of the longest Fibonacci-like subsequence of

arr

. If one does not exist, return

0

.

A

subsequence

is derived from another sequence

arr

by deleting any number of elements (including none) from

arr

, without changing the order of the remaining elements. For example,

[3, 5, 8]

is a subsequence of

[3, 4, 5, 6, 7, 8]

.

Example 1:

Input:

arr = [1,2,3,4,5,6,7,8]

Output:

5

Explanation:

The longest subsequence that is fibonacci-like: [1,2,3,5,8].

Example 2:

Input:

arr = [1,3,7,11,12,14,18]

Output:

3

Explanation

:

The longest subsequence that is fibonacci-like: [1,11,12], [3,11,14] or [7,11,18].

Constraints:

$3 \leq \text{arr.length} \leq 1000$

$1 \leq \text{arr}[i] < \text{arr}[i + 1] \leq 10$

9

Code Snippets

C++:

```
class Solution {
public:
    int lenLongestFibSubseq(vector<int>& arr) {

    }
};
```

Java:

```
class Solution {
    public int lenLongestFibSubseq(int[] arr) {

    }
}
```

Python3:

```
class Solution:
    def lenLongestFibSubseq(self, arr: List[int]) -> int:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function lenLongestFibSubseq(arr: number[]): number {

};
```

C#:

```
public class Solution {
    public int LenLongestFibSubseq(int[] arr) {

    }
}
```

C:

```
int lenLongestFibSubseq(int* arr, int arrSize) {

}
```

Go:

```
func lenLongestFibSubseq(arr []int) int {

}
```

Kotlin:

```
class Solution {
    fun lenLongestFibSubseq(arr: IntArray): Int {

    }
}
```

Swift:

```
class Solution {
    func lenLongestFibSubseq(_ arr: [Int]) -> Int {

    }
}
```

Rust:

```
impl Solution {
    pub fn len_longest_fib_subseq(arr: Vec<i32>) -> i32 {

    }
}
```

Ruby:

```
# @param {Integer[]} arr
# @return {Integer}
def len_longest_fib_subseq(arr)

end
```

PHP:

```
class Solution {

    /**
     * @param Integer[] $arr
     * @return Integer
     */
}
```

```

*/
function lenLongestFibSubseq($arr) {

}

}

```

Dart:

```

class Solution {
  int lenLongestFibSubseq(List<int> arr) {

  }

}

```

Scala:

```

object Solution {
  def lenLongestFibSubseq(arr: Array[Int]): Int = {

  }

}

```

Elixir:

```

defmodule Solution do
  @spec len_longest_fib_subseq(arr :: [integer]) :: integer
  def len_longest_fib_subseq(arr) do

  end

end

```

Erlang:

```

-spec len_longest_fib_subseq(Arr :: [integer()]) -> integer().
len_longest_fib_subseq(Arr) ->

.

```

Racket:

```

(define/contract (len-longest-fib-subseq arr)
  (-> (listof exact-integer?) exact-integer?)
  )

```

Solutions

C++ Solution:

```
/*
 * Problem: Length of Longest Fibonacci Subsequence
 * Difficulty: Medium
 * Tags: array, dp, hash
 *
 * 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 lenLongestFibSubseq(vector<int>& arr) {

    }
};
```

Java Solution:

```
/**
 * Problem: Length of Longest Fibonacci Subsequence
 * Difficulty: Medium
 * Tags: array, dp, hash
 *
 * 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 lenLongestFibSubseq(int[] arr) {

    }
}
```

Python3 Solution:

```

"""
Problem: Length of Longest Fibonacci Subsequence
Difficulty: Medium
Tags: array, dp, hash

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 lenLongestFibSubseq(self, arr: List[int]) -> int:
        # TODO: Implement optimized solution
        pass

```

Python Solution:

```

class Solution(object):
    def lenLongestFibSubseq(self, arr):
        """
        :type arr: List[int]
        :rtype: int
        """

```

JavaScript Solution:

```

/**
 * Problem: Length of Longest Fibonacci Subsequence
 * Difficulty: Medium
 * Tags: array, dp, hash
 *
 * 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[]} arr
 * @return {number}
 */
var lenLongestFibSubseq = function(arr) {

```

```
};
```

TypeScript Solution:

```
/**
 * Problem: Length of Longest Fibonacci Subsequence
 * Difficulty: Medium
 * Tags: array, dp, hash
 *
 * 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 lenLongestFibSubseq(arr: number[]): number {

};
```

C# Solution:

```
/*
 * Problem: Length of Longest Fibonacci Subsequence
 * Difficulty: Medium
 * Tags: array, dp, hash
 *
 * 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 LenLongestFibSubseq(int[] arr) {

    }
}
```

C Solution:

```
/*
 * Problem: Length of Longest Fibonacci Subsequence
 * Difficulty: Medium
```

```

* Tags: array, dp, hash
*
* 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 lenLongestFibSubseq(int* arr, int arrSize) {

}

```

Go Solution:

```

// Problem: Length of Longest Fibonacci Subsequence
// Difficulty: Medium
// Tags: array, dp, hash
//
// 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 lenLongestFibSubseq(arr []int) int {

}

```

Kotlin Solution:

```

class Solution {
    fun lenLongestFibSubseq(arr: IntArray): Int {

    }
}

```

Swift Solution:

```

class Solution {
    func lenLongestFibSubseq(_ arr: [Int]) -> Int {

    }
}

```

Rust Solution:

```
// Problem: Length of Longest Fibonacci Subsequence
// Difficulty: Medium
// Tags: array, dp, hash
//
// 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 len_longest_fib_subseq(arr: Vec<i32>) -> i32 {

    }
}
```

Ruby Solution:

```
# @param {Integer[]} arr
# @return {Integer}
def len_longest_fib_subseq(arr)

end
```

PHP Solution:

```
class Solution {

    /**
     * @param Integer[] $arr
     * @return Integer
     */
    function lenLongestFibSubseq($arr) {

    }
}
```

Dart Solution:

```
class Solution {
    int lenLongestFibSubseq(List<int> arr) {
```

```
}  
}
```

Scala Solution:

```
object Solution {  
  def lenLongestFibSubseq(arr: Array[Int]): Int = {  
  
  }  
}
```

Elixir Solution:

```
defmodule Solution do  
  @spec len_longest_fib_subseq(arr :: [integer]) :: integer  
  def len_longest_fib_subseq(arr) do  
  
  end  
end
```

Erlang Solution:

```
-spec len_longest_fib_subseq(Arr :: [integer()]) -> integer().  
len_longest_fib_subseq(Arr) ->  
.
```

Racket Solution:

```
(define/contract (len-longest-fib-subseq arr)  
  (-> (listof exact-integer?) exact-integer?)  
)
```