Problem 1218: Longest Arithmetic Subsequence of Given Difference

Problem Information

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

Problem Description

Given an integer array

arr

and an integer

difference

, return the length of the longest subsequence in

arr

which is an arithmetic sequence such that the difference between adjacent elements in the subsequence equals

difference

.

A

subsequence

is a sequence that can be derived from

arr

by deleting some or no elements without changing the order of the remaining elements.

Example 1:

Input:

arr = [1,2,3,4], difference = 1

Output:

4

Explanation:

The longest arithmetic subsequence is [1,2,3,4].

Example 2:

Input:

arr = [1,3,5,7], difference = 1

Output:

1

Explanation:

The longest arithmetic subsequence is any single element.

Example 3:

Input:

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

Output:

4

Explanation:

The longest arithmetic subsequence is [7,5,3,1].

Constraints:

1 <= arr.length <= 10

5

-10

4

<= arr[i], difference <= 10

4

Code Snippets

C++:

```
class Solution {
public:
int longestSubsequence(vector<int>& arr, int difference) {

}
};
```

Java:

```
class Solution {
public int longestSubsequence(int[] arr, int difference) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function longestSubsequence(arr: number[], difference: number): number {

};
```

C#:

```
public class Solution {
public int LongestSubsequence(int[] arr, int difference) {

}
}
```

C:

```
int longestSubsequence(int* arr, int arrSize, int difference) {

}
```

Go:

```
func longestSubsequence(arr []int, difference int) int {

}
```

Kotlin:

```
class Solution {
fun longestSubsequence(arr: IntArray, difference: Int): Int {

}
}
```

Swift:

```
class Solution {
func longestSubsequence(_ arr: [Int], _ difference: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn longest_subsequence(arr: Vec<i32>, difference: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} arr
# @param {Integer} difference
# @return {Integer}
def longest_subsequence(arr, difference)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $arr
* @param Integer $difference
* @return Integer
*/
function longestSubsequence($arr, $difference) {

}
}
```

Dart:

```
class Solution {
int longestSubsequence(List<int> arr, int difference) {

}
}
```

Scala:

```
object Solution {
def longestSubsequence(arr: Array[Int], difference: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec longest_subsequence(arr :: [integer], difference :: integer) :: integer
def longest_subsequence(arr, difference) do

end
end
```

Erlang:

```
-spec longest_subsequence(Arr :: [integer()], Difference :: integer()) ->
integer().
```

```
longest_subsequence(Arr, Difference) ->
.
```

Racket:

```
(define/contract (longest-subsequence arr difference)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Longest Arithmetic Subsequence of Given Difference
* 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 longestSubsequence(vector<int>& arr, int difference) {

}
};
```

Java Solution:

```
/**
* Problem: Longest Arithmetic Subsequence of Given Difference
* 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 longestSubsequence(int[] arr, int difference) {

}
}
```

Python3 Solution:

```
"""
Problem: Longest Arithmetic Subsequence of Given Difference
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 longestSubsequence(self, arr: List[int], difference: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Longest Arithmetic Subsequence of Given Difference
* 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
* @param {number} difference
* @return {number}
*/
var longestSubsequence = function(arr, difference) {

};
```

TypeScript Solution:

```
/**
* Problem: Longest Arithmetic Subsequence of Given Difference
* 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 longestSubsequence(arr: number[], difference: number): number {

};
```

C# Solution:

```
/*
* Problem: Longest Arithmetic Subsequence of Given Difference
* 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 LongestSubsequence(int[] arr, int difference) {

}
}
```

C Solution:

```
/*
* Problem: Longest Arithmetic Subsequence of Given Difference
* 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 longestSubsequence(int* arr, int arrSize, int difference) {

}
```

Go Solution:

```
// Problem: Longest Arithmetic Subsequence of Given Difference
// 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 longestSubsequence(arr []int, difference int) int {

}
```

Kotlin Solution:

```
class Solution {
fun longestSubsequence(arr: IntArray, difference: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func longestSubsequence(_ arr: [Int], _ difference: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Longest Arithmetic Subsequence of Given Difference
// 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 longest_subsequence(arr: Vec<i32>, difference: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} arr
# @param {Integer} difference
# @return {Integer}
def longest_subsequence(arr, difference)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $arr
* @param Integer $difference
* @return Integer
*/
function longestSubsequence($arr, $difference) {

}
}
```

Dart Solution:

```
class Solution {
int longestSubsequence(List<int> arr, int difference) {

}
}
```

Scala Solution:

```
object Solution {
def longestSubsequence(arr: Array[Int], difference: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec longest_subsequence(arr :: [integer], difference :: integer) :: integer
def longest_subsequence(arr, difference) do

end
end
```

Erlang Solution:

```
-spec longest_subsequence(Arr :: [integer()], Difference :: integer()) ->
integer().
longest_subsequence(Arr, Difference) ->
```

```
.
```

Racket Solution:

```
(define/contract (longest-subsequence arr difference)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```