Problem 1438: Longest Continuous Subarray With Absolute Diff Less Than or Equal to Limit

Problem Information

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

Problem Description

Given an array of integers

nums

and an integer

limit

, return the size of the longest

non-empty

subarray such that the absolute difference between any two elements of this subarray is less than or equal to

limit

.

Example 1:

Input:

nums = [8,2,4,7], limit = 4

Output:

2

Explanation:

All subarrays are: [8] with maximum absolute diff |8-8| = 0 <= 4. [8,2] with maximum absolute diff |8-2| = 6 > 4. [8,2,4] with maximum absolute diff |8-2| = 6 > 4. [8,2,4,7] with maximum absolute diff |8-2| = 6 > 4. [2] with maximum absolute diff |2-2| = 0 <= 4. [2,4] with maximum absolute diff |2-4| = 2 <= 4. [2,4,7] with maximum absolute diff |2-7| = 5 > 4. [4] with maximum absolute diff |4-4| = 0 <= 4. [4,7] with maximum absolute diff |4-7| = 3 <= 4. [7] with maximum absolute diff |7-7| = 0 <= 4. Therefore, the size of the longest subarray is 2.

Example 2:

Input:

nums = [10,1,2,4,7,2], limit = 5

Output:

4

Explanation:

The subarray [2,4,7,2] is the longest since the maximum absolute diff is |2-7| = 5 <= 5.

Example 3:

Input:

nums = [4,2,2,2,4,4,2,2], limit = 0

Output:

3

Constraints:

1 <= nums.length <= 10

5

1 <= nums[i] <= 10

9

0 <= limit <= 10

9

Code Snippets

C++:

```
class Solution {
public:
int longestSubarray(vector<int>& nums, int limit) {

}
};
```

Java:

```
class Solution {
public int longestSubarray(int[] nums, int limit) {

}
}
```

Python3:

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

Python:

```
class Solution(object):
def longestSubarray(self, nums, limit):
```

```
"""
:type nums: List[int]
:type limit: int
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

```
function longestSubarray(nums: number[], limit: number): number {

};
```

C#:

```
public class Solution {
public int LongestSubarray(int[] nums, int limit) {

}
}
```

C:

```
int longestSubarray(int* nums, int numsSize, int limit) {

}
```

Go:

```
func longestSubarray(nums []int, limit int) int {

}
```

Kotlin:

```
class Solution {
fun longestSubarray(nums: IntArray, limit: Int): Int {

}
}
```

Swift:

```
class Solution {
func longestSubarray(_ nums: [Int], _ limit: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn longest_subarray(nums: Vec<i32>, limit: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @param {Integer} limit
# @return {Integer}
def longest_subarray(nums, limit)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $limit
* @return Integer
*/
function longestSubarray($nums, $limit) {
}
}
```

Dart:

```
class Solution {
int longestSubarray(List<int> nums, int limit) {

}
}
```

Scala:

```
object Solution {
def longestSubarray(nums: Array[Int], limit: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec longest_subarray(nums :: [integer], limit :: integer) :: integer
def longest_subarray(nums, limit) do

end
end
```

Erlang:

```
-spec longest_subarray(Nums :: [integer()], Limit :: integer()) -> integer().
longest_subarray(Nums, Limit) ->
.
```

Racket:

```
(define/contract (longest-subarray nums limit)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Longest Continuous Subarray With Absolute Diff Less Than or Equal
to Limit
* Difficulty: Medium
* Tags: array, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int longestSubarray(vector<int>& nums, int limit) {

}
};
```

Java Solution:

```
/**
* Problem: Longest Continuous Subarray With Absolute Diff Less Than or Equal
to Limit
* Difficulty: Medium
* Tags: array, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int longestSubarray(int[] nums, int limit) {

}
}
```

Python3 Solution:

```
"""
Problem: Longest Continuous Subarray With Absolute Diff Less Than or Equal to
Limit
Difficulty: Medium
Tags: array, queue, heap

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def longestSubarray(self, nums: List[int], limit: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Longest Continuous Subarray With Absolute Diff Less Than or Equal
to Limit
* Difficulty: Medium
* Tags: array, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number[]} nums
* @param {number} limit
* @return {number}
*/
var longestSubarray = function(nums, limit) {

};
```

TypeScript Solution:

```
/**
* Problem: Longest Continuous Subarray With Absolute Diff Less Than or Equal
to Limit
* Difficulty: Medium
* Tags: array, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function longestSubarray(nums: number[], limit: number): number {

};
```

C# Solution:

```
/*
* Problem: Longest Continuous Subarray With Absolute Diff Less Than or Equal
to Limit
* Difficulty: Medium
* Tags: array, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int LongestSubarray(int[] nums, int limit) {

}
}
```

C Solution:

```
/*
* Problem: Longest Continuous Subarray With Absolute Diff Less Than or Equal
to Limit
* Difficulty: Medium
* Tags: array, queue, heap
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

int longestSubarray(int* nums, int numsSize, int limit) {

}
```

Go Solution:

```
// Problem: Longest Continuous Subarray With Absolute Diff Less Than or Equal
to Limit
// Difficulty: Medium
// Tags: array, queue, heap
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func longestSubarray(nums []int, limit int) int {

}
```

Kotlin Solution:

```
class Solution {
fun longestSubarray(nums: IntArray, limit: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func longestSubarray(_ nums: [Int], _ limit: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Longest Continuous Subarray With Absolute Diff Less Than or Equal
to Limit
// Difficulty: Medium
// Tags: array, queue, heap
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn longest_subarray(nums: Vec<i32>, limit: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @param {Integer} limit
# @return {Integer}
def longest_subarray(nums, limit)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @param Integer $limit
* @return Integer
*/
function longestSubarray($nums, $limit) {



}
}
```

Dart Solution:

```
class Solution {
int longestSubarray(List<int> nums, int limit) {

}
}
```

Scala Solution:

```
object Solution {
def longestSubarray(nums: Array[Int], limit: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec longest_subarray(nums :: [integer], limit :: integer) :: integer
def longest_subarray(nums, limit) do

end
end
```

Erlang Solution:

```
-spec longest_subarray(Nums :: [integer()], Limit :: integer()) -> integer().
longest_subarray(Nums, Limit) ->
.
```

Racket Solution:

```
(define/contract (longest-subarray nums limit)
(-> (listof exact-integer?) exact-integer? exact-integer?)
)
```