Problem 1574: Shortest Subarray to be Removed to Make Array Sorted

Problem Information

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

Problem Description

Given an integer array

arr

, remove a subarray (can be empty) from

arr

such that the remaining elements in

arr

are

non-decreasing

.

Return

the length of the shortest subarray to remove

.

A

subarray

is a contiguous subsequence of the array.

Example 1:

Input:

arr = [1,2,3,10,4,2,3,5]

Output:

3

Explanation:

The shortest subarray we can remove is [10,4,2] of length 3. The remaining elements after that will be [1,2,3,3,5] which are sorted. Another correct solution is to remove the subarray [3,10,4].

Example 2:

Input:

arr = [5,4,3,2,1]

Output:

4

Explanation:

Since the array is strictly decreasing, we can only keep a single element. Therefore we need to remove a subarray of length 4, either [5,4,3,2] or [4,3,2,1].

Example 3:

Input:

arr = [1,2,3]

Output:

0

Explanation:

The array is already non-decreasing. We do not need to remove any elements.

Constraints:

1 <= arr.length <= 10

5

0 <= arr[i] <= 10

9

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

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

}
```

Go:

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

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

/**
```

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (find-length-of-shortest-subarray arr)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Shortest Subarray to be Removed to Make Array Sorted
* Difficulty: Medium
* Tags: array, sort, search, stack
*
* 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 findLengthOfShortestSubarray(vector<int>& arr) {

}
};
```

Java Solution:

```
/**
* Problem: Shortest Subarray to be Removed to Make Array Sorted
* Difficulty: Medium
* Tags: array, sort, search, stack
*
* 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 findLengthOfShortestSubarray(int[] arr) {

}
```

```
}
```

Python3 Solution:

```
"""
Problem: Shortest Subarray to be Removed to Make Array Sorted
Difficulty: Medium
Tags: array, sort, search, stack

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Shortest Subarray to be Removed to Make Array Sorted
* Difficulty: Medium
* Tags: array, sort, search, stack
*
* 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[]} arr
* @return {number}
*/
var findLengthOfShortestSubarray = function(arr) {

};
```

TypeScript Solution:

```
/**
* Problem: Shortest Subarray to be Removed to Make Array Sorted
* Difficulty: Medium
* Tags: array, sort, search, stack
*
* 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 findLengthOfShortestSubarray(arr: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Shortest Subarray to be Removed to Make Array Sorted
* Difficulty: Medium
* Tags: array, sort, search, stack
*
* 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 FindLengthOfShortestSubarray(int[] arr) {

}
}
```

C Solution:

```
/*
* Problem: Shortest Subarray to be Removed to Make Array Sorted
* Difficulty: Medium
* Tags: array, sort, search, stack
*
* 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 findLengthOfShortestSubarray(int* arr, int arrSize) {

}
```

Go Solution:

```
// Problem: Shortest Subarray to be Removed to Make Array Sorted
// Difficulty: Medium
// Tags: array, sort, search, stack
//
// 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 findLengthOfShortestSubarray(arr []int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

```
}
}
```

Rust Solution:

```
// Problem: Shortest Subarray to be Removed to Make Array Sorted
// Difficulty: Medium
// Tags: array, sort, search, stack
//
// 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 find_length_of_shortest_subarray(arr: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (find-length-of-shortest-subarray arr)
(-> (listof exact-integer?) exact-integer?)
)
```