Problem 2110: Number of Smooth Descent Periods of a Stock

Problem Information

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

Problem Description

You are given an integer array

prices

representing the daily price history of a stock, where

prices[i]

is the stock price on the

i

th

day.

A

smooth descent period

of a stock consists of

one or more contiguous

days such that the price on each day is

lower

than the price on the

preceding day

by

exactly

1

. The first day of the period is exempted from this rule.

Return

the number of

smooth descent periods

.

Example 1:

Input:

prices = [3,2,1,4]

Output:

7

Explanation:

There are 7 smooth descent periods: [3], [2], [1], [4], [3,2], [2,1], and [3,2,1] Note that a period with one day is a smooth descent period by the definition.

Example 2:

Input:

prices = [8,6,7,7]

Output:

4

Explanation:

There are 4 smooth descent periods: [8], [6], [7], and [7] Note that [8,6] is not a smooth descent period as 8 - 6 ≠ 1.

Example 3:

Input:

prices = [1]

Output:

1

Explanation:

There is 1 smooth descent period: [1]

Constraints:

1 <= prices.length <= 10

5

1 <= prices[i] <= 10

5

Code Snippets

C++:

```
class Solution {
public:
long long getDescentPeriods(vector<int>& prices) {

}
};
```

Java:

```
class Solution {
public long getDescentPeriods(int[] prices) {

}
}
```

Python3:

```
class Solution:
def getDescentPeriods(self, prices: List[int]) -> int:
```

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function getDescentPeriods(prices: number[]): number {

};
```

C#:

```
public class Solution {
public long GetDescentPeriods(int[] prices) {

}
}
```

C:

```
long long getDescentPeriods(int* prices, int pricesSize) {

}
```

Go:

```
func getDescentPeriods(prices []int) int64 {

}
```

Kotlin:

```
class Solution {
fun getDescentPeriods(prices: IntArray): Long {

}
}
```

Swift:

```
class Solution {
func getDescentPeriods(_ prices: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn get_descent_periods(prices: Vec<i32>) -> i64 {

}
}
```

Ruby:

```
# @param {Integer[]} prices
# @return {Integer}
def get_descent_periods(prices)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $prices
* @return Integer
*/
function getDescentPeriods($prices) {

}
}
```

Dart:

```
class Solution {
int getDescentPeriods(List<int> prices) {

}
}
```

Scala:

```
object Solution {
def getDescentPeriods(prices: Array[Int]): Long = {

}
}
```

Elixir:

```
defmodule Solution do
@spec get_descent_periods(prices :: [integer]) :: integer
def get_descent_periods(prices) do

end
end
```

Erlang:

```
-spec get_descent_periods(Prices :: [integer()]) -> integer().
get_descent_periods(Prices) ->
.
```

Racket:

```
(define/contract (get-descent-periods prices)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Number of Smooth Descent Periods of a Stock
* Difficulty: Medium
* Tags: array, dp, math
*
* 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:
long long getDescentPeriods(vector<int>& prices) {

}
};
```

Java Solution:

```
/**
* Problem: Number of Smooth Descent Periods of a Stock
* Difficulty: Medium
* Tags: array, dp, math
*
* 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 long getDescentPeriods(int[] prices) {

}
}
```

Python3 Solution:

```
"""
Problem: Number of Smooth Descent Periods of a Stock
Difficulty: Medium
Tags: array, dp, math

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

Python Solution:

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

```
"""
```

JavaScript Solution:

```
/**
* Problem: Number of Smooth Descent Periods of a Stock
* Difficulty: Medium
* Tags: array, dp, math
*
* 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[]} prices
* @return {number}
*/
var getDescentPeriods = function(prices) {

};
```

TypeScript Solution:

```
/**
* Problem: Number of Smooth Descent Periods of a Stock
* Difficulty: Medium
* Tags: array, dp, math
*
* 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 getDescentPeriods(prices: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Number of Smooth Descent Periods of a Stock
* Difficulty: Medium
* Tags: array, dp, math
*
* 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 long GetDescentPeriods(int[] prices) {

}
}
```

C Solution:

```
/*
* Problem: Number of Smooth Descent Periods of a Stock
* Difficulty: Medium
* Tags: array, dp, math
*
* 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
*/

long long getDescentPeriods(int* prices, int pricesSize) {

}
```

Go Solution:

```
// Problem: Number of Smooth Descent Periods of a Stock
// Difficulty: Medium
// Tags: array, dp, math
//
// 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 getDescentPeriods(prices []int) int64 {

}
```

Kotlin Solution:

```
class Solution {
fun getDescentPeriods(prices: IntArray): Long {

}
}
```

Swift Solution:

```
class Solution {
func getDescentPeriods(_ prices: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Number of Smooth Descent Periods of a Stock
// Difficulty: Medium
// Tags: array, dp, math
//
// 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 get_descent_periods(prices: Vec<i32>) -> i64 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} prices
# @return {Integer}
def get_descent_periods(prices)
```

```
end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $prices
* @return Integer
*/
function getDescentPeriods($prices) {

}
}
```

Dart Solution:

```
class Solution {
int getDescentPeriods(List<int> prices) {

}
}
```

Scala Solution:

```
object Solution {
def getDescentPeriods(prices: Array[Int]): Long = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec get_descent_periods(prices :: [integer]) :: integer
def get_descent_periods(prices) do

end
end
```

Erlang Solution:

```
-spec get_descent_periods(Prices :: [integer()]) -> integer().
get_descent_periods(Prices) ->
.
```

Racket Solution:

```
(define/contract (get-descent-periods prices)
(-> (listof exact-integer?) exact-integer?)
)
```