Problem 1996: The Number of Weak Characters in the Game

Problem Information

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

Problem Description

You are playing a game that contains multiple characters, and each of the characters has

two

main properties:

attack

and

defense

. You are given a 2D integer array

properties

where

properties[i] = [attack

i

, defense

i

]

represents the properties of the

i

th

character in the game.

A character is said to be

weak

if any other character has

both

attack and defense levels

strictly greater

than this character's attack and defense levels. More formally, a character

i

is said to be

weak

if there exists another character

j

where

attack

j

> attack

i

and

defense

j

> defense

i

.

Return

the number of

weak

characters

.

Example 1:

Input:

properties = [[5,5],[6,3],[3,6]]

Output:

0

Explanation:

No character has strictly greater attack and defense than the other.

Example 2:

Input:

properties = [[2,2],[3,3]]

Output:

1

Explanation:

The first character is weak because the second character has a strictly greater attack and defense.

Example 3:

Input:

properties = [[1,5],[10,4],[4,3]]

Output:

1

Explanation:

The third character is weak because the second character has a strictly greater attack and defense.

Constraints:

2 <= properties.length <= 10

5

properties[i].length == 2

1 <= attack

i

, defense

i

<= 10

5

Code Snippets

C++:

```
class Solution {
public:
int numberOfWeakCharacters(vector<vector<int>>& properties) {

}
};
```

Java:

```
class Solution {
public int numberOfWeakCharacters(int[][] properties) {

}
}
```

Python3:

```
class Solution:
def numberOfWeakCharacters(self, properties: List[List[int]]) -> int:
```

Python:

```
class Solution(object):
def numberOfWeakCharacters(self, properties):
```

```
"""
:type properties: List[List[int]]
:rtype: int
"""
```

JavaScript:

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

};
```

TypeScript:

```
function numberOfWeakCharacters(properties: number[][]): number {

};
```

C#:

```
public class Solution {
public int NumberOfWeakCharacters(int[][] properties) {

}
}
```

C:

```
int numberOfWeakCharacters(int** properties, int propertiesSize, int*
propertiesColSize) {

}
```

Go:

```
func numberOfWeakCharacters(properties [][]int) int {

}
```

Kotlin:

```
class Solution {
fun numberOfWeakCharacters(properties: Array<IntArray>): Int {

}
}
```

Swift:

```
class Solution {
func numberOfWeakCharacters(_ properties: [[Int]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn number_of_weak_characters(properties: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[][]} properties
# @return {Integer}
def number_of_weak_characters(properties)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $properties
* @return Integer
*/
function numberOfWeakCharacters($properties) {

}
```

```
}
```

Dart:

```
class Solution {
int numberOfWeakCharacters(List<List<int>> properties) {

}
}
```

Scala:

```
object Solution {
def numberOfWeakCharacters(properties: Array[Array[Int]]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec number_of_weak_characters(properties :: [[integer]]) :: integer
def number_of_weak_characters(properties) do

end
end
```

Erlang:

```
-spec number_of_weak_characters(Properties :: [[integer()]]) -> integer().
number_of_weak_characters(Properties) ->
.
```

Racket:

```
(define/contract (number-of-weak-characters properties)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: The Number of Weak Characters in the Game
* Difficulty: Medium
* Tags: array, greedy, sort, 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 numberOfWeakCharacters(vector<vector<int>>& properties) {

}
};
```

Java Solution:

```
/**
* Problem: The Number of Weak Characters in the Game
* Difficulty: Medium
* Tags: array, greedy, sort, 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 numberOfWeakCharacters(int[][] properties) {

}
}
```

Python3 Solution:

```
"""
Problem: The Number of Weak Characters in the Game
Difficulty: Medium
Tags: array, greedy, sort, 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 numberOfWeakCharacters(self, properties: List[List[int]]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: The Number of Weak Characters in the Game
* Difficulty: Medium
* Tags: array, greedy, sort, 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[][]} properties
* @return {number}
*/
var numberOfWeakCharacters = function(properties) {

};
```

TypeScript Solution:

```
/**
* Problem: The Number of Weak Characters in the Game
* Difficulty: Medium
* Tags: array, greedy, sort, 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 numberOfWeakCharacters(properties: number[][]): number {

};
```

C# Solution:

```
/*
* Problem: The Number of Weak Characters in the Game
* Difficulty: Medium
* Tags: array, greedy, sort, 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 NumberOfWeakCharacters(int[][] properties) {

}
}
```

C Solution:

```
/*
* Problem: The Number of Weak Characters in the Game
* Difficulty: Medium
* Tags: array, greedy, sort, 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 numberOfWeakCharacters(int** properties, int propertiesSize, int*
propertiesColSize) {

}
```

Go Solution:

```
// Problem: The Number of Weak Characters in the Game
// Difficulty: Medium
// Tags: array, greedy, sort, 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 numberOfWeakCharacters(properties [][]int) int {

}
```

Kotlin Solution:

```
class Solution {
fun numberOfWeakCharacters(properties: Array<IntArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func numberOfWeakCharacters(_ properties: [[Int]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: The Number of Weak Characters in the Game
// Difficulty: Medium
```

```
// Tags: array, greedy, sort, 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 number_of_weak_characters(properties: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[][]} properties
# @return {Integer}
def number_of_weak_characters(properties)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $properties
* @return Integer
*/
function numberOfWeakCharacters($properties) {

}
}
```

Dart Solution:

```
class Solution {
int numberOfWeakCharacters(List<List<int>> properties) {

}
}
```

Scala Solution:

```
object Solution {
def numberOfWeakCharacters(properties: Array[Array[Int]]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec number_of_weak_characters(properties :: [[integer]]) :: integer
def number_of_weak_characters(properties) do

end
end
```

Erlang Solution:

```
-spec number_of_weak_characters(Properties :: [[integer()]]) -> integer().
number_of_weak_characters(Properties) ->
.
```

Racket Solution:

```
(define/contract (number-of-weak-characters properties)
(-> (listof (listof exact-integer?)) exact-integer?)
)
```