Problem 3169: Count Days Without Meetings

Problem Information

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

Problem Description

You are given a positive integer

days

representing the total number of days an employee is available for work (starting from day 1). You are also given a 2D array

meetings

of size

n

where,

meetings[i] = [start_i, end_i]

represents the starting and ending days of meeting

i

(inclusive).

Return the count of days when the employee is available for work but no meetings are scheduled.

Note:

The meetings may overlap.

Example 1:

Input:

days = 10, meetings = [[5,7],[1,3],[9,10]]

Output:

2

Explanation:

There is no meeting scheduled on the 4

th

and 8

th

days.

Example 2:

Input:

days = 5, meetings = [[2,4],[1,3]]

Output:

1

Explanation:

There is no meeting scheduled on the 5

th

day.

Example 3:

Input:

days = 6, meetings = [[1,6]]

Output:

0

Explanation:

Meetings are scheduled for all working days.

Constraints:

1 <= days <= 10

9

1 <= meetings.length <= 10

5

meetings[i].length == 2

1 <= meetings[i][0] <= meetings[i][1] <= days

Code Snippets

C++:

```
class Solution {
public:
```

```
int countDays(int days, vector<vector<int>>& meetings) {

}
};
```

Java:

```
class Solution {
public int countDays(int days, int[][] meetings) {

}
}
```

Python3:

```
class Solution:
def countDays(self, days: int, meetings: List[List[int]]) -> int:
```

Python:

```
class Solution(object):
def countDays(self, days, meetings):
"""
:type days: int
:type meetings: List[List[int]]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} days
* @param {number[][]} meetings
* @return {number}
*/
var countDays = function(days, meetings) {

};
```

TypeScript:

```
function countDays(days: number, meetings: number[][]): number {

};
```

C#:

```
public class Solution {
public int CountDays(int days, int[][] meetings) {

}
}
```

C:

```
int countDays(int days, int** meetings, int meetingsSize, int*
meetingsColSize) {

}
```

Go:

```
func countDays(days int, meetings [][]int) int {

}
```

Kotlin:

```
class Solution {
fun countDays(days: Int, meetings: Array<IntArray>): Int {

}
}
```

Swift:

```
class Solution {
func countDays(_ days: Int, _ meetings: [[Int]]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_days(days: i32, meetings: Vec<Vec<i32>>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} days
# @param {Integer[][]} meetings
# @return {Integer}
def count_days(days, meetings)

end
```

PHP:

```
class Solution {

/**
* @param Integer $days
* @param Integer[][] $meetings
* @return Integer
*/
function countDays($days, $meetings) {

}
}
```

Dart:

```
class Solution {
int countDays(int days, List<List<int>> meetings) {

}
}
```

Scala:

```
object Solution {
def countDays(days: Int, meetings: Array[Array[Int]]): Int = {

}
```

```
}
```

Elixir:

```
defmodule Solution do
@spec count_days(days :: integer, meetings :: [[integer]]) :: integer
def count_days(days, meetings) do

end
end
```

Erlang:

```
-spec count_days(Days :: integer(), Meetings :: [[integer()]]) -> integer().
count_days(Days, Meetings) ->
.
```

Racket:

```
(define/contract (count-days days meetings)
(-> exact-integer? (listof (listof exact-integer?)) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Days Without Meetings
* Difficulty: Medium
* Tags: array, sort
*
* 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 countDays(int days, vector<vector<int>>& meetings) {
```

```
}
};
```

Java Solution:

```
/**
* Problem: Count Days Without Meetings
* Difficulty: Medium
* Tags: array, sort
*
* 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 countDays(int days, int[][] meetings) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Days Without Meetings
Difficulty: Medium
Tags: array, sort

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 countDays(self, days: int, meetings: List[List[int]]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def countDays(self, days, meetings):
"""
:type days: int
:type meetings: List[List[int]]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Count Days Without Meetings
* Difficulty: Medium
* Tags: array, sort
*
* 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} days
* @param {number[][]} meetings
* @return {number}
*/
var countDays = function(days, meetings) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Days Without Meetings
* Difficulty: Medium
* Tags: array, sort
*
* 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 countDays(days: number, meetings: number[][]): number {
```

```
};
```

C# Solution:

```
/*
* Problem: Count Days Without Meetings
* Difficulty: Medium
* Tags: array, sort
*
* 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 CountDays(int days, int[][] meetings) {

}
}
```

C Solution:

```
/*
* Problem: Count Days Without Meetings
* Difficulty: Medium
* Tags: array, sort
*
* 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 countDays(int days, int** meetings, int meetingsSize, int*
meetingsColSize) {

}
```

Go Solution:

```
// Problem: Count Days Without Meetings
// Difficulty: Medium
// Tags: array, sort
//
// 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 countDays(days int, meetings [][]int) int {

}
```

Kotlin Solution:

```
class Solution {
fun countDays(days: Int, meetings: Array<IntArray>): Int {

}
}
```

Swift Solution:

```
class Solution {
func countDays(_ days: Int, _ meetings: [[Int]]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Count Days Without Meetings
// Difficulty: Medium
// Tags: array, sort
//
// 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 count_days(days: i32, meetings: Vec<Vec<i32>>) -> i32 {

}
```

```
}
```

Ruby Solution:

```
# @param {Integer} days
# @param {Integer[][]} meetings
# @return {Integer}
def count_days(days, meetings)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $days
* @param Integer[][] $meetings
* @return Integer
*/
function countDays($days, $meetings) {

}
}
```

Dart Solution:

```
class Solution {
int countDays(int days, List<List<int>> meetings) {

}
}
```

Scala Solution:

```
object Solution {
def countDays(days: Int, meetings: Array[Array[Int]]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_days(days :: integer, meetings :: [[integer]]) :: integer
def count_days(days, meetings) do

end
end
```

Erlang Solution:

```
-spec count_days(Days :: integer(), Meetings :: [[integer()]]) -> integer().
count_days(Days, Meetings) ->
.
```

Racket Solution:

```
(define/contract (count-days days meetings)
(-> exact-integer? (listof (listof exact-integer?)) exact-integer?)
)
```