Problem 2409: Count Days Spent Together

Problem Information

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

Problem Description

Alice and Bob are traveling to Rome for separate business meetings.

You are given 4 strings

arriveAlice

,

leaveAlice

,

arriveBob

, and

leaveBob

. Alice will be in the city from the dates

arriveAlice

to

leaveAlice

(

inclusive

), while Bob will be in the city from the dates

arriveBob

to

leaveBob

(

inclusive

). Each will be a 5-character string in the format

"MM-DD"

, corresponding to the month and day of the date.

Return

the total number of days that Alice and Bob are in Rome together.

You can assume that all dates occur in the

same

calendar year, which is

not

a leap year. Note that the number of days per month can be represented as:

[31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

.

Example 1:

Input:

arriveAlice = "08-15", leaveAlice = "08-18", arriveBob = "08-16", leaveBob = "08-19"

Output:

3

Explanation:

Alice will be in Rome from August 15 to August 18. Bob will be in Rome from August 16 to August 19. They are both in Rome together on August 16th, 17th, and 18th, so the answer is 3.

Example 2:

Input:

arriveAlice = "10-01", leaveAlice = "10-31", arriveBob = "11-01", leaveBob = "12-31"

Output:

0

Explanation:

There is no day when Alice and Bob are in Rome together, so we return 0.

Constraints:

All dates are provided in the format

"MM-DD"

.

Alice and Bob's arrival dates are

earlier than or equal to

their leaving dates.

The given dates are valid dates of a

non-leap

year.

Code Snippets

C++:

```
class Solution {
public:
int countDaysTogether(string arriveAlice, string leaveAlice, string
arriveBob, string leaveBob) {

}
};
```

Java:

```
class Solution {
public int countDaysTogether(String arriveAlice, String leaveAlice, String
arriveBob, String leaveBob) {

}
}
```

Python3:

```
class Solution:
def countDaysTogether(self, arriveAlice: str, leaveAlice: str, arriveBob:
str, leaveBob: str) -> int:
```

Python:

```
class Solution(object):
def countDaysTogether(self, arriveAlice, leaveAlice, arriveBob, leaveBob):
"""
:type arriveAlice: str
:type leaveAlice: str
:type arriveBob: str
:type leaveBob: str
:rtype: int
"""
```

JavaScript:

```
/**
* @param {string} arriveAlice
* @param {string} leaveAlice
* @param {string} arriveBob
* @param {string} leaveBob
* @return {number}
*/
var countDaysTogether = function(arriveAlice, leaveAlice, arriveBob,
leaveBob) {

};
```

TypeScript:

```
function countDaysTogether(arriveAlice: string, leaveAlice: string,
arriveBob: string, leaveBob: string): number {

};
```

C#:

```
public class Solution {
public int CountDaysTogether(string arriveAlice, string leaveAlice, string
arriveBob, string leaveBob) {

}
}
```

C:

```
int countDaysTogether(char* arriveAlice, char* leaveAlice, char* arriveBob,
char* leaveBob) {

}
```

Go:

```
func countDaysTogether(arriveAlice string, leaveAlice string, arriveBob
string, leaveBob string) int {

}
```

Kotlin:

```
class Solution {
fun countDaysTogether(arriveAlice: String, leaveAlice: String, arriveBob:
String, leaveBob: String): Int {

}
}
```

Swift:

```
class Solution {
func countDaysTogether(_ arriveAlice: String, _ leaveAlice: String, _
arriveBob: String, _ leaveBob: String) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_days_together(arrive_alice: String, leave_alice: String,
arrive_bob: String, leave_bob: String) -> i32 {

}
}
```

Ruby:

```
# @param {String} arrive_alice
# @param {String} leave_alice
# @param {String} arrive_bob
# @param {String} leave_bob
# @return {Integer}
def count_days_together(arrive_alice, leave_alice, arrive_bob, leave_bob)

end
```

PHP:

```
class Solution {

/**
* @param String $arriveAlice
* @param String $leaveAlice
* @param String $arriveBob
* @param String $leaveBob
* @return Integer
*/
function countDaysTogether($arriveAlice, $leaveAlice, $arriveBob, $leaveBob)
{

}
}
```

Dart:

```
class Solution {
int countDaysTogether(String arriveAlice, String leaveAlice, String
arriveBob, String leaveBob) {

}
}
```

Scala:

```
object Solution {
def countDaysTogether(arriveAlice: String, leaveAlice: String, arriveBob:
String, leaveBob: String): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_days_together(arrive_alice :: String.t, leave_alice :: String.t,
arrive_bob :: String.t, leave_bob :: String.t) :: integer
def count_days_together(arrive_alice, leave_alice, arrive_bob, leave_bob) do

end
end
```

Erlang:

```
-spec count_days_together(ArriveAlice :: unicode:unicode_binary(), LeaveAlice
:: unicode:unicode_binary(), ArriveBob :: unicode:unicode_binary(), LeaveBob
:: unicode:unicode_binary()) -> integer().
count_days_together(ArriveAlice, LeaveAlice, ArriveBob, LeaveBob) ->
.
```

Racket:

```
(define/contract (count-days-together arriveAlice leaveAlice arriveBob
leaveBob)
(-> string? string? string? string? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Count Days Spent Together
* Difficulty: Easy
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int countDaysTogether(string arriveAlice, string leaveAlice, string
arriveBob, string leaveBob) {

}
};
```

Java Solution:

```
/**
* Problem: Count Days Spent Together
* Difficulty: Easy
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int countDaysTogether(String arriveAlice, String leaveAlice, String
arriveBob, String leaveBob) {

}
}
```

Python3 Solution:

```
"""
Problem: Count Days Spent Together
Difficulty: Easy
Tags: string, math

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def countDaysTogether(self, arriveAlice: str, leaveAlice: str, arriveBob:
str, leaveBob: str) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def countDaysTogether(self, arriveAlice, leaveAlice, arriveBob, leaveBob):
"""
:type arriveAlice: str
:type leaveAlice: str
:type arriveBob: str
:type leaveBob: str
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Count Days Spent Together
* Difficulty: Easy
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} arriveAlice
* @param {string} leaveAlice
* @param {string} arriveBob
* @param {string} leaveBob
* @return {number}
*/
var countDaysTogether = function(arriveAlice, leaveAlice, arriveBob,
leaveBob) {

};
```

TypeScript Solution:

```
/**
* Problem: Count Days Spent Together
* Difficulty: Easy
* Tags: string, math
*
```

```
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function countDaysTogether(arriveAlice: string, leaveAlice: string,
arriveBob: string, leaveBob: string): number {

};
```

C# Solution:

```
/*
* Problem: Count Days Spent Together
* Difficulty: Easy
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int CountDaysTogether(string arriveAlice, string leaveAlice, string
arriveBob, string leaveBob) {

}
}
```

C Solution:

```
/*
* Problem: Count Days Spent Together
* Difficulty: Easy
* Tags: string, math
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/
```

```
int countDaysTogether(char* arriveAlice, char* leaveAlice, char* arriveBob,
char* leaveBob) {

}
```

Go Solution:

```
// Problem: Count Days Spent Together
// Difficulty: Easy
// Tags: string, math
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func countDaysTogether(arriveAlice string, leaveAlice string, arriveBob
string, leaveBob string) int {

}
```

Kotlin Solution:

```
class Solution {
fun countDaysTogether(arriveAlice: String, leaveAlice: String, arriveBob:
String, leaveBob: String): Int {

}
}
```

Swift Solution:

```
class Solution {
func countDaysTogether(_ arriveAlice: String, _ leaveAlice: String, _
arriveBob: String, _ leaveBob: String) -> Int {

}
}
```

Rust Solution:

```
// Problem: Count Days Spent Together
// Difficulty: Easy
// Tags: string, math
//
// Approach: String manipulation with hash map or two pointers
// 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_together(arrive_alice: String, leave_alice: String,
arrive_bob: String, leave_bob: String) -> i32 {

}
}
```

Ruby Solution:

```
# @param {String} arrive_alice
# @param {String} leave_alice
# @param {String} arrive_bob
# @param {String} leave_bob
# @return {Integer}
def count_days_together(arrive_alice, leave_alice, arrive_bob, leave_bob)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $arriveAlice
* @param String $leaveAlice
* @param String $arriveBob
* @param String $leaveBob
* @return Integer
*/
function countDaysTogether($arriveAlice, $leaveAlice, $arriveBob, $leaveBob)
{

}
}
```

Dart Solution:

```
class Solution {
int countDaysTogether(String arriveAlice, String leaveAlice, String
arriveBob, String leaveBob) {

}
}
```

Scala Solution:

```
object Solution {
def countDaysTogether(arriveAlice: String, leaveAlice: String, arriveBob:
String, leaveBob: String): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_days_together(arrive_alice :: String.t, leave_alice :: String.t,
arrive_bob :: String.t, leave_bob :: String.t) :: integer
def count_days_together(arrive_alice, leave_alice, arrive_bob, leave_bob) do

end
end
```

Erlang Solution:

```
-spec count_days_together(ArriveAlice :: unicode:unicode_binary(), LeaveAlice
:: unicode:unicode_binary(), ArriveBob :: unicode:unicode_binary(), LeaveBob
:: unicode:unicode_binary()) -> integer().
count_days_together(ArriveAlice, LeaveAlice, ArriveBob, LeaveBob) ->
.
```

Racket Solution:

```
(define/contract (count-days-together arriveAlice leaveAlice arriveBob
leaveBob)
(-> string? string? string? string? exact-integer?)
)
```