Problem 1109: Corporate Flight Bookings

Problem Information

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

Problem Description

There are

n

flights that are labeled from

1

to

n

.

You are given an array of flight bookings

bookings

, where

bookings[i] = [first

i

, last

i

, seats

i

]

represents a booking for flights

first

i

through

last

i

(

inclusive

) with

seats

i

seats reserved for

each flight

in the range.

Return

an array

answer

of length

n

, where

answer[i]

is the total number of seats reserved for flight

i

.

Example 1:

Input:

bookings = [[1,2,10],[2,3,20],[2,5,25]], n = 5

Output:

[10,55,45,25,25]

Explanation:

Flight labels: 1 2 3 4 5 Booking 1 reserved: 10 10 Booking 2 reserved: 20 20 Booking 3 reserved: 25 25 25 25 Total seats: 10 55 45 25 25 Hence, answer = [10,55,45,25,25]

Example 2:

Input:

bookings = [[1,2,10],[2,2,15]], n = 2

Output:

[10,25]

Explanation:

Flight labels: 1 2 Booking 1 reserved: 10 10 Booking 2 reserved: 15 Total seats: 10 25 Hence, answer = [10,25]

Constraints:

1 <= n <= 2 * 10

4

1 <= bookings.length <= 2 * 10

4

bookings[i].length == 3

1 <= first

i

<= last

i

<= n

1 <= seats

i

<= 10

4

Code Snippets

C++:

```
class Solution {
public:
vector<int> corpFlightBookings(vector<vector<int>>& bookings, int n) {

}
};
```

Java:

```
class Solution {
public int[] corpFlightBookings(int[][] bookings, int n) {

}
}
```

Python3:

```
class Solution:
def corpFlightBookings(self, bookings: List[List[int]], n: int) -> List[int]:
```

Python:

```
class Solution(object):
def corpFlightBookings(self, bookings, n):
"""
:type bookings: List[List[int]]
:type n: int
:rtype: List[int]
"""
```

JavaScript:

```
/**
* @param {number[][]} bookings
* @param {number} n
* @return {number[]}
*/
var corpFlightBookings = function(bookings, n) {
```

```
};
```

TypeScript:

```
function corpFlightBookings(bookings: number[][], n: number): number[] {

};
```

C#:

```
public class Solution {
public int[] CorpFlightBookings(int[][] bookings, int n) {

}
}
```

C:

```
/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* corpFlightBookings(int** bookings, int bookingsSize, int*
bookingsColSize, int n, int* returnSize) {

}
```

Go:

```
func corpFlightBookings(bookings [][]int, n int) []int {

}
```

Kotlin:

```
class Solution {
fun corpFlightBookings(bookings: Array<IntArray>, n: Int): IntArray {

}
}
```

Swift:

```
class Solution {
func corpFlightBookings(_ bookings: [[Int]], _ n: Int) -> [Int] {

}
}
```

Rust:

```
impl Solution {
pub fn corp_flight_bookings(bookings: Vec<Vec<i32>>, n: i32) -> Vec<i32> {

}
}
```

Ruby:

```
# @param {Integer[][]} bookings
# @param {Integer} n
# @return {Integer[]}
def corp_flight_bookings(bookings, n)

end
```

PHP:

```
class Solution {

/**
* @param Integer[][] $bookings
* @param Integer $n
* @return Integer[]
*/
function corpFlightBookings($bookings, $n) {

}
}
```

Dart:

```
class Solution {
List<int> corpFlightBookings(List<List<int>> bookings, int n) {

}
```

```
}
```

Scala:

```
object Solution {
def corpFlightBookings(bookings: Array[Array[Int]], n: Int): Array[Int] = {

}
}
```

Elixir:

```
defmodule Solution do
@spec corp_flight_bookings(bookings :: [[integer]], n :: integer) ::
[integer]
def corp_flight_bookings(bookings, n) do

end
end
```

Erlang:

```
-spec corp_flight_bookings(Bookings :: [[integer()]], N :: integer()) ->
[integer()].
corp_flight_bookings(Bookings, N) ->
.
```

Racket:

```
(define/contract (corp-flight-bookings bookings n)
(-> (listof (listof exact-integer?)) exact-integer? (listof exact-integer?))
)
```

Solutions

C++ Solution:

```
/*
* Problem: Corporate Flight Bookings
* Difficulty: Medium
```

```
* Tags: array
*
* 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:
vector<int> corpFlightBookings(vector<vector<int>>& bookings, int n) {

}
};
```

Java Solution:

```
/**
* Problem: Corporate Flight Bookings
* Difficulty: Medium
* Tags: array
*
* 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[] corpFlightBookings(int[][] bookings, int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Corporate Flight Bookings
Difficulty: Medium
Tags: array

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 corpFlightBookings(self, bookings: List[List[int]], n: int) -> List[int]:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def corpFlightBookings(self, bookings, n):
"""
:type bookings: List[List[int]]
:type n: int
:rtype: List[int]
"""
```

JavaScript Solution:

```
/**
* Problem: Corporate Flight Bookings
* Difficulty: Medium
* Tags: array
*
* 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[][]} bookings
* @param {number} n
* @return {number[]}
*/
var corpFlightBookings = function(bookings, n) {

};
```

TypeScript Solution:

```
/**
* Problem: Corporate Flight Bookings
* Difficulty: Medium
* Tags: array
*
* 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 corpFlightBookings(bookings: number[][], n: number): number[] {

};
```

C# Solution:

```
/*
* Problem: Corporate Flight Bookings
* Difficulty: Medium
* Tags: array
*
* 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[] CorpFlightBookings(int[][] bookings, int n) {

}
}
```

C Solution:

```
/*
* Problem: Corporate Flight Bookings
* Difficulty: Medium
* Tags: array
*
* 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
```

```
*/

/**
* Note: The returned array must be malloced, assume caller calls free().
*/
int* corpFlightBookings(int** bookings, int bookingsSize, int*
bookingsColSize, int n, int* returnSize) {

}
```

Go Solution:

```
// Problem: Corporate Flight Bookings
// Difficulty: Medium
// Tags: array
//
// 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 corpFlightBookings(bookings [][]int, n int) []int {

}
```

Kotlin Solution:

```
class Solution {
fun corpFlightBookings(bookings: Array<IntArray>, n: Int): IntArray {

}
}
```

Swift Solution:

```
class Solution {
func corpFlightBookings(_ bookings: [[Int]], _ n: Int) -> [Int] {

}
}
```

Rust Solution:

```
// Problem: Corporate Flight Bookings
// Difficulty: Medium
// Tags: array
//
// 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 corp_flight_bookings(bookings: Vec<Vec<i32>>, n: i32) -> Vec<i32> {

}
}
```

Ruby Solution:

```
# @param {Integer[][]} bookings
# @param {Integer} n
# @return {Integer[]}
def corp_flight_bookings(bookings, n)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[][] $bookings
* @param Integer $n
* @return Integer[]
*/
function corpFlightBookings($bookings, $n) {

}
}
```

Dart Solution:

```
class Solution {
List<int> corpFlightBookings(List<List<int>> bookings, int n) {
```

```
}
}
```

Scala Solution:

```
object Solution {
def corpFlightBookings(bookings: Array[Array[Int]], n: Int): Array[Int] = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec corp_flight_bookings(bookings :: [[integer]], n :: integer) ::
[integer]
def corp_flight_bookings(bookings, n) do

end
end
```

Erlang Solution:

```
-spec corp_flight_bookings(Bookings :: [[integer()]], N :: integer()) ->
[integer()].
corp_flight_bookings(Bookings, N) ->
.
```

Racket Solution:

```
(define/contract (corp-flight-bookings bookings n)
(-> (listof (listof exact-integer?)) exact-integer? (listof exact-integer?))
)
```