Problem 732: My Calendar III

Problem Information

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

Problem Description

A

k

-booking happens when

k

events have some non-empty intersection (i.e., there is some time that is common to all

k

events.)

You are given some events

[startTime, endTime)

, after each given event, return an integer

k

representing the maximum

k

-booking between all the previous events.

Implement the

MyCalendarThree

class:

MyCalendarThree()

Initializes the object.

int book(int startTime, int endTime)

Returns an integer

k

representing the largest integer such that there exists a

k

-booking in the calendar.

Example 1:

Input

["MyCalendarThree", "book", "book", "book", "book", "book", "book"] [[], [10, 20], [50, 60], [10, 40], [5, 15], [5, 10], [25, 55]]

Output

[null, 1, 1, 2, 3, 3, 3]

Explanation

MyCalendarThree myCalendarThree = new MyCalendarThree(); myCalendarThree.book(10, 20); // return 1 myCalendarThree.book(50, 60); // return 1 myCalendarThree.book(10, 40); //

return 2 myCalendarThree.book(5, 15); // return 3 myCalendarThree.book(5, 10); // return 3 myCalendarThree.book(25, 55); // return 3

Constraints:

0 <= startTime < endTime <= 10

9

At most

400

calls will be made to

book

.

Code Snippets

C++:

```
class MyCalendarThree {
public:
MyCalendarThree() {

}

int book(int startTime, int endTime) {

}
};

/**
* Your MyCalendarThree object will be instantiated and called as such:
* MyCalendarThree* obj = new MyCalendarThree();
* int param_1 = obj->book(startTime,endTime);
*/
```

Java:

```
class MyCalendarThree {

public MyCalendarThree() {

}

public int book(int startTime, int endTime) {

}
}

/**
* Your MyCalendarThree object will be instantiated and called as such:
* MyCalendarThree obj = new MyCalendarThree();
* int param_1 = obj.book(startTime,endTime);
*/
```

Python3:

```
class MyCalendarThree:

def __init__(self):



def book(self, startTime: int, endTime: int) -> int:




# Your MyCalendarThree object will be instantiated and called as such:
# obj = MyCalendarThree()
# param_1 = obj.book(startTime,endTime)
```

Python:

```
class MyCalendarThree(object):

def __init__(self):



def book(self, startTime, endTime):
"""
```

```
:type startTime: int
:type endTime: int
:rtype: int
"""



# Your MyCalendarThree object will be instantiated and called as such:
# obj = MyCalendarThree()
# param_1 = obj.book(startTime,endTime)
```

JavaScript:

```
var MyCalendarThree = function() {

};

/**
* @param {number} startTime
* @param {number} endTime
* @return {number}
*/
MyCalendarThree.prototype.book = function(startTime, endTime) {

};

/**
* Your MyCalendarThree object will be instantiated and called as such:
* var obj = new MyCalendarThree()
* var param_1 = obj.book(startTime,endTime)
*/
```

TypeScript:

```
class MyCalendarThree {
constructor() {

}

book(startTime: number, endTime: number): number {
}
}

/**
* Your MyCalendarThree object will be instantiated and called as such:
* var obj = new MyCalendarThree()
* var param_1 = obj.book(startTime,endTime)
*/
```

C#:

```
public class MyCalendarThree {

public MyCalendarThree() {

}

public int Book(int startTime, int endTime) {

}
}

/**
* Your MyCalendarThree object will be instantiated and called as such:
* MyCalendarThree obj = new MyCalendarThree();
* int param_1 = obj.Book(startTime,endTime);
*/
```

C:

```



typedef struct {

} MyCalendarThree;


MyCalendarThree* myCalendarThreeCreate() {

}
```

```
int myCalendarThreeBook(MyCalendarThree* obj, int startTime, int endTime) {

}

void myCalendarThreeFree(MyCalendarThree* obj) {

}

/**
* Your MyCalendarThree struct will be instantiated and called as such:
* MyCalendarThree* obj = myCalendarThreeCreate();
* int param_1 = myCalendarThreeBook(obj, startTime, endTime);

* myCalendarThreeFree(obj);
*/
```

Go:

```
type MyCalendarThree struct {

}


func Constructor() MyCalendarThree {

}


func (this *MyCalendarThree) Book(startTime int, endTime int) int {

}


/**
* Your MyCalendarThree object will be instantiated and called as such:
* obj := Constructor();
* param_1 := obj.Book(startTime,endTime);
*/
```

Kotlin:

```
class MyCalendarThree() {

fun book(startTime: Int, endTime: Int): Int {

}

}

/**
* Your MyCalendarThree object will be instantiated and called as such:
* var obj = MyCalendarThree()
* var param_1 = obj.book(startTime,endTime)
*/
```

Swift:

```

class MyCalendarThree {

init() {

}

func book(_ startTime: Int, _ endTime: Int) -> Int {

}
}

/**
* Your MyCalendarThree object will be instantiated and called as such:
* let obj = MyCalendarThree()
* let ret_1: Int = obj.book(startTime, endTime)
*/
```

Rust:

```
struct MyCalendarThree {

}



/**
* `&self` means the method takes an immutable reference.
* If you need a mutable reference, change it to `&mut self` instead.
*/
impl MyCalendarThree {

fn new() -> Self {

}

fn book(&self, start_time: i32, end_time: i32) -> i32 {

}
}

/**
* Your MyCalendarThree object will be instantiated and called as such:
* let obj = MyCalendarThree::new();
* let ret_1: i32 = obj.book(startTime, endTime);
*/
```

Ruby:

```
class MyCalendarThree
def initialize()

end



=begin
:type start_time: Integer
:type end_time: Integer
:rtype: Integer
=end
def book(start_time, end_time)

end



end

# Your MyCalendarThree object will be instantiated and called as such:
# obj = MyCalendarThree.new()
# param_1 = obj.book(start_time, end_time)
```

PHP:

```
class MyCalendarThree {
/**
*/
function __construct() {

}

/**
* @param Integer $startTime
* @param Integer $endTime
* @return Integer
*/
function book($startTime, $endTime) {

}
}

/**
* Your MyCalendarThree object will be instantiated and called as such:
* $obj = MyCalendarThree();
* $ret_1 = $obj->book($startTime, $endTime);
*/
```

Dart:

```
class MyCalendarThree {

MyCalendarThree() {

}

int book(int startTime, int endTime) {

}
}

/**
* Your MyCalendarThree object will be instantiated and called as such:
* MyCalendarThree obj = MyCalendarThree();
* int param1 = obj.book(startTime,endTime);
```

```
*/
```

Scala:

```
class MyCalendarThree() {

def book(startTime: Int, endTime: Int): Int = {

}

}

/**
* Your MyCalendarThree object will be instantiated and called as such:
* val obj = new MyCalendarThree()
* val param_1 = obj.book(startTime,endTime)
*/
```

Elixir:

```
defmodule MyCalendarThree do
@spec init_() :: any
def init_() do

end

@spec book(start_time :: integer, end_time :: integer) :: integer
def book(start_time, end_time) do

end
end

# Your functions will be called as such:
# MyCalendarThree.init_()
# param_1 = MyCalendarThree.book(start_time, end_time)

# MyCalendarThree.init_ will be called before every test case, in which you
can do some necessary initializations.
```

Erlang:

```
-spec my_calendar_three_init_() -> any().
my_calendar_three_init_() ->
.

-spec my_calendar_three_book(StartTime :: integer(), EndTime :: integer()) ->
integer().
my_calendar_three_book(StartTime, EndTime) ->
.


%% Your functions will be called as such:
%% my_calendar_three_init_(),
%% Param_1 = my_calendar_three_book(StartTime, EndTime),

%% my_calendar_three_init_ will be called before every test case, in which
you can do some necessary initializations.
```

Racket:

```
(define my-calendar-three%
(class object%
(super-new)

(init-field)

; book : exact-integer? exact-integer? -> exact-integer?
(define/public (book start-time end-time)
)))

;; Your my-calendar-three% object will be instantiated and called as such:
;; (define obj (new my-calendar-three%))
;; (define param_1 (send obj book start-time end-time))
```

Solutions

C++ Solution:

```
/*
* Problem: My Calendar III
* Difficulty: Hard
* Tags: array, tree, search
```

```
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class MyCalendarThree {
public:
MyCalendarThree() {

}

int book(int startTime, int endTime) {

}
};

/**
* Your MyCalendarThree object will be instantiated and called as such:
* MyCalendarThree* obj = new MyCalendarThree();
* int param_1 = obj->book(startTime,endTime);
*/
```

Java Solution:

```
/**
* Problem: My Calendar III
* Difficulty: Hard
* Tags: array, tree, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class MyCalendarThree {

public MyCalendarThree() {

}
public int book(int startTime, int endTime) {

}
}

/**
* Your MyCalendarThree object will be instantiated and called as such:
* MyCalendarThree obj = new MyCalendarThree();
* int param_1 = obj.book(startTime,endTime);
*/
```

Python3 Solution:

```
"""
Problem: My Calendar III
Difficulty: Hard
Tags: array, tree, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

class MyCalendarThree:

def __init__(self):



def book(self, startTime: int, endTime: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class MyCalendarThree(object):

def __init__(self):



def book(self, startTime, endTime):
"""
```

```
:type startTime: int
:type endTime: int
:rtype: int
"""



# Your MyCalendarThree object will be instantiated and called as such:
# obj = MyCalendarThree()
# param_1 = obj.book(startTime,endTime)
```

JavaScript Solution:

```
/**
* Problem: My Calendar III
* Difficulty: Hard
* Tags: array, tree, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/



var MyCalendarThree = function() {

};

/**
* @param {number} startTime
* @param {number} endTime
* @return {number}
*/
MyCalendarThree.prototype.book = function(startTime, endTime) {

};

/**
* Your MyCalendarThree object will be instantiated and called as such:
* var obj = new MyCalendarThree()
* var param_1 = obj.book(startTime,endTime)
```

```
*/
```

TypeScript Solution:

```
/**
* Problem: My Calendar III
* Difficulty: Hard
* Tags: array, tree, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class MyCalendarThree {
constructor() {

}

book(startTime: number, endTime: number): number {

}
}

/**
* Your MyCalendarThree object will be instantiated and called as such:
* var obj = new MyCalendarThree()
* var param_1 = obj.book(startTime,endTime)
*/
```

C# Solution:

```
/*
* Problem: My Calendar III
* Difficulty: Hard
* Tags: array, tree, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/
```

```
public class MyCalendarThree {

public MyCalendarThree() {

}

public int Book(int startTime, int endTime) {

}
}

/**
* Your MyCalendarThree object will be instantiated and called as such:
* MyCalendarThree obj = new MyCalendarThree();
* int param_1 = obj.Book(startTime,endTime);
*/
```

C Solution:

```
/*
* Problem: My Calendar III
* Difficulty: Hard
* Tags: array, tree, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/



typedef struct {

} MyCalendarThree;


MyCalendarThree* myCalendarThreeCreate() {

}
```

```
int myCalendarThreeBook(MyCalendarThree* obj, int startTime, int endTime) {

}

void myCalendarThreeFree(MyCalendarThree* obj) {

}

/**
* Your MyCalendarThree struct will be instantiated and called as such:
* MyCalendarThree* obj = myCalendarThreeCreate();
* int param_1 = myCalendarThreeBook(obj, startTime, endTime);

* myCalendarThreeFree(obj);
*/
```

Go Solution:

```
// Problem: My Calendar III
// Difficulty: Hard
// Tags: array, tree, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

type MyCalendarThree struct {

}


func Constructor() MyCalendarThree {

}


func (this *MyCalendarThree) Book(startTime int, endTime int) int {

}
```

```
/**
* Your MyCalendarThree object will be instantiated and called as such:
* obj := Constructor();
* param_1 := obj.Book(startTime,endTime);
*/
```

Kotlin Solution:

```
class MyCalendarThree() {

fun book(startTime: Int, endTime: Int): Int {

}

}

/**
* Your MyCalendarThree object will be instantiated and called as such:
* var obj = MyCalendarThree()
* var param_1 = obj.book(startTime,endTime)
*/
```

Swift Solution:

```
class MyCalendarThree {

init() {

}

func book(_ startTime: Int, _ endTime: Int) -> Int {

}
}

/**
* Your MyCalendarThree object will be instantiated and called as such:
* let obj = MyCalendarThree()
* let ret_1: Int = obj.book(startTime, endTime)
```

```
*/
```

Rust Solution:

```
// Problem: My Calendar III
// Difficulty: Hard
// Tags: array, tree, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

struct MyCalendarThree {

}



/**
* `&self` means the method takes an immutable reference.
* If you need a mutable reference, change it to `&mut self` instead.
*/
impl MyCalendarThree {

fn new() -> Self {

}

fn book(&self, start_time: i32, end_time: i32) -> i32 {

}
}

/**
* Your MyCalendarThree object will be instantiated and called as such:
* let obj = MyCalendarThree::new();
* let ret_1: i32 = obj.book(startTime, endTime);
*/
```

Ruby Solution:

```
class MyCalendarThree
def initialize()

end



=begin
:type start_time: Integer
:type end_time: Integer
:rtype: Integer
=end
def book(start_time, end_time)

end



end

# Your MyCalendarThree object will be instantiated and called as such:
# obj = MyCalendarThree.new()
# param_1 = obj.book(start_time, end_time)
```

PHP Solution:

```
class MyCalendarThree {
/**
*/
function __construct() {

}

/**
* @param Integer $startTime
* @param Integer $endTime
* @return Integer
*/
function book($startTime, $endTime) {

}
}

/**
```

```
* Your MyCalendarThree object will be instantiated and called as such:
* $obj = MyCalendarThree();
* $ret_1 = $obj->book($startTime, $endTime);
*/
```

Dart Solution:

```
class MyCalendarThree {

MyCalendarThree() {

}

int book(int startTime, int endTime) {

}
}

/**
* Your MyCalendarThree object will be instantiated and called as such:
* MyCalendarThree obj = MyCalendarThree();
* int param1 = obj.book(startTime,endTime);
*/
```

Scala Solution:

```
class MyCalendarThree() {

def book(startTime: Int, endTime: Int): Int = {

}

}

/**
* Your MyCalendarThree object will be instantiated and called as such:
* val obj = new MyCalendarThree()
* val param_1 = obj.book(startTime,endTime)
*/
```

Elixir Solution:

```
defmodule MyCalendarThree do
@spec init_() :: any
def init_() do

end

@spec book(start_time :: integer, end_time :: integer) :: integer
def book(start_time, end_time) do

end
end

# Your functions will be called as such:
# MyCalendarThree.init_()
# param_1 = MyCalendarThree.book(start_time, end_time)

# MyCalendarThree.init_ will be called before every test case, in which you
can do some necessary initializations.
```

Erlang Solution:

```
-spec my_calendar_three_init_() -> any().
my_calendar_three_init_() ->
.

-spec my_calendar_three_book(StartTime :: integer(), EndTime :: integer()) ->
integer().
my_calendar_three_book(StartTime, EndTime) ->
.


%% Your functions will be called as such:
%% my_calendar_three_init_(),
%% Param_1 = my_calendar_three_book(StartTime, EndTime),

%% my_calendar_three_init_ will be called before every test case, in which
you can do some necessary initializations.
```

Racket Solution:

```
(define my-calendar-three%
(class object%
(super-new)

(init-field)

; book : exact-integer? exact-integer? -> exact-integer?
(define/public (book start-time end-time)
)))

;; Your my-calendar-three% object will be instantiated and called as such:
;; (define obj (new my-calendar-three%))
;; (define param_1 (send obj book start-time end-time))
```