729. My Calendar I

DescriptionHintsSubmissionsDiscussSolution

Implement a `MyCalendar` class to store your events. A new event can be added if adding the event will not cause a double booking.

Your class will have the method, `book(int start, int end)`. Formally, this represents a booking on the half open interval `[start, end)`, the range of real numbers `x` such that `start <= x < end`.

A *double booking* happens when two events have some non-empty intersection (ie., there is some time that is common to both events.)

For each call to the method `MyCalendar.book`, return `true` if the event can be added to the calendar successfully without causing a double booking. Otherwise, return `false` and do not add the event to the calendar.

Your class will be called like this: `MyCalendar cal = new MyCalendar(); MyCalendar.book(start, end)`

Example 1:

```
MyCalendar();
MyCalendar.book(10, 20); // returns true
MyCalendar.book(15, 25); // returns false
MyCalendar.book(20, 30); // returns true
Explanation:
The first event can be booked.  The second can't because time 15 is already
booked by another event.
The third event can be booked, as the first event takes every time less than 20,
but not including 20.
```

Note:

- The number of calls to `MyCalendar.book` per test case will be at most `1000`.
- In calls to `MyCalendar.book(start, end)`, `start` and `end` are integers in the range `[0, 10^9]`.

Seen this question in a real interview before?

- Difficulty:Medium
- Total Accepted:2K
- Total Submissions:5.8K
- Contributor: ccyjoshua
-
- Subscribe to see which companies asked this question.

Related Topics

Similar Questions

My Calendar II

731. My Calendar II

DescriptionHintsSubmissionsDiscussSolution

Implement a `MyCalendarTwo` class to store your events. A new event can be added if adding the event will not cause a **triple** booking.

Your class will have one method, `book(int start, int end)`. Formally, this represents a booking on the half open interval `[start, end)`, the range of real numbers `x` such that `start <= x < end`.

A *triple booking* happens when **three** events have some non-empty intersection (ie., there is some time that is common to all 3 events.)

For each call to the method `MyCalendar.book`, return `true` if the event can be added to the calendar successfully without causing a **triple** booking. Otherwise, return `false` and do not add the event to the calendar.

Your class will be called like this: `MyCalendar cal = new MyCalendar(); MyCalendar.book(start, end)`

Example 1:

```
MyCalendar();
MyCalendar.book(10, 20); // returns true
MyCalendar.book(50, 60); // returns true
MyCalendar.book(10, 40); // returns true
MyCalendar.book(5, 15); // returns false
MyCalendar.book(5, 10); // returns true
MyCalendar.book(25, 55); // returns true
Explanation:
The first two events can be booked.  The third event can be double booked.
The fourth event (5, 15) can't be booked, because it would result in a triple
booking.
The fifth event (5, 10) can be booked, as it does not use time 10 which is
already double booked.
The sixth event (25, 55) can be booked, as the time in [25, 40) will be double
booked with the third event;
the time [40, 50) will be single booked, and the time [50, 55) will be double
booked with the second event.
```

Note:

- The number of calls to `MyCalendar.book` per test case will be at most `1000`.

- In calls to `MyCalendar.book(start, end)`, `start` and `end` are integers in the range `[0, 10^9]`.

Seen this question in a real interview before?

- Difficulty:Easy
- Total Accepted:937
- Total Submissions:3.1K
- Contributor: ccyjoshua
-
- Subscribe to see which companies asked this question.

  Related Topics

  Similar Questions

  My Calendar I

Short brute force python solution

```
class MyCalendar:
    def __init__(self):
        self.intervals = []

    def book(self, start, end):
        for s, e in self.intervals:
            if not (start >= e or end <= s): return False
        self.intervals.append((start, end))
        return True



class MyCalendarTwo:
    def __init__(self):
        self.overlaps = []
        self.calendar = []

    def book(self, start, end):
```

```
for i, j in self.overlaps:
    if start < j and end > i:
        return False
for i, j in self.calendar:
    if start < j and end > i:
        self.overlaps.append((max(start, i), min(end, j)))
self.calendar.append((start, end))
return True
```