731. My Calendar II

Medium 566 78 Add to List Share

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.
```

```java
TreeMap<Integer,Integer> map=new TreeMap<>();;
public MyCalendarTwo() {
    map.clear();
}

public boolean book(int start, int end) {
    map.put(start,map.getOrDefault(start,0)+1);
    map.put(end,map.getOrDefault(end,0)-1);

    int count=0;
    for(Map.Entry<Integer,Integer> key: map.entrySet()){
       count+=key.getValue();
       if(count>2){
        map.put(start,map.getOrDefault(start,0) - 1);
        map.put(end,map.getOrDefault(end,0) + 1);
        if(map.get(start)==0) map.remove(start);
        if(map.get(end)==0) map.remove(end);
        return  false;
       }
    }
    return true;
}
```

1. same as we did in meeting rooms II question.
2. just put start in treemap but increasing its frequency and end by decreasing its frequency.
3. intialize count by 0.
4. and iterate over treeMap and add its value in count as soon it exceed 2 return false.