252. Meeting Rooms (/problems/meeting-rooms/)

April 12, 2016 | 29.1K views

Average Rating: 4.96 (25 votes)

Given an array of meeting time intervals consisting of start and end times `[[s1,e1],[s2,e2],...]` ($s_i < e_i$), determine if a person could attend all meetings.

Example 1:

```
Input: [[0,30],[5,10],[15,20]]
Output: false
```

Example 2:

```
Input: [[7,10],[2,4]]
Output: true
```

NOTE: input types have been changed on April 15, 2019. Please reset to default code definition to get new method signature.

Solution

Approach #1 (Brute Force) [Accepted]

The straight-forward solution is to compare every two meetings in the array, and see if they conflict with each other (i.e. if they overlap). Two meetings overlap if one of them starts while the other is still taking place.

Java

```
public boolean canAttendMeetings(Interval[] intervals) {
    for (int i = 0; i < intervals.length; i++) {
        for (int j = i + 1; j < intervals.length; j++) {
            if (overlap(intervals[i], intervals[j])) return false;
        }
    }
    return true;
}

private boolean overlap(Interval i1, Interval i2) {
    return ((i1.start >= i2.start && i1.start < i2.end)
         || (i2.start >= i1.start && i2.start < i1.end));
}
```

Overlap Condition

The overlap condition in the code above can be written in a more concise way. Consider two non-overlapping meetings. The earlier meeting ends before the later meeting begins. Therefore, the *minimum* end time of the two meetings (which is the end time of the earlier meeting) is smaller than or equal the *maximum* start time of the two meetings (which is the start time of the later meeting).

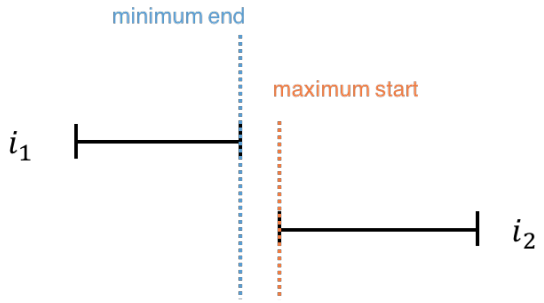

Figure 1. Two non-overlapping intervals.

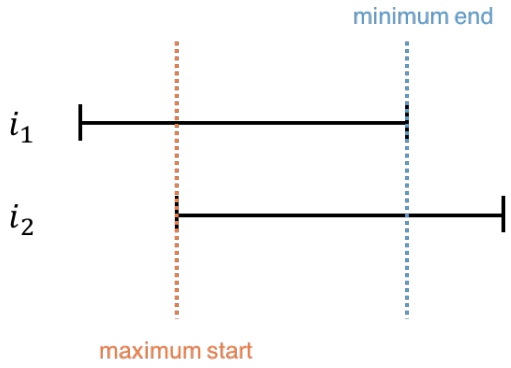

Figure 2. Two overlapping intervals.

So the condition can be rewritten as follows.

```
private boolean overlap(Interval i1, Interval i2) {
    return (Math.min(i1.end, i2.end) >
            Math.max(i1.start, i2.start));
}
```

Complexity Analysis

Because we have two check every meeting with every other meeting, the total run time is $O(n^2)$. No additional space is used, so the space complexity is $O(1)$.

Approach #2 (Sorting) [Accepted]

The idea here is to sort the meetings by starting time. Then, go through the meetings one by one and make sure that each meeting ends before the next one starts.

Java

```
public boolean canAttendMeetings(Interval[] intervals) {
    Arrays.sort(intervals, new Comparator<Interval>() {
        public int compare(Interval i1, Interval i2) {
            return i1.start - i2.start;
        }
    });
    for (int i = 0; i < intervals.length - 1; i++) {
        if (intervals[i].end > intervals[i + 1].start) return false;
    }
    return true;
}
```

Complexity Analysis

- Time complexity : $O(n \log n)$. The time complexity is dominated by sorting. Once the array has been sorted, only $O(n)$ time is taken to go through the array and determine if there is any overlap.
- Space complexity : $O(1)$. Since no additional space is allocated.

Analysis written by: @noran

Rate this article:

Previous (/articles/shortest-word-distance/) Next (/articles/implement-trie-prefix-tree/)

Comments: 11

Articles > 252. Meeting Rooms

Sort By

Type comment here... (Markdown is supported)

Preview Post

jiangyucara (jiangyucara) ★ 44 May 27, 2019 10:43 AM

(/jiangyucara)

Here is the solution that solves the new version:

```
public boolean canAttendMeetings(int[][] intervals) {
        Comparator<int[]> c=(int[] a, int[] b) -> (a[0]-b[0]);
        Arrays.sort(intervals, c);
```

Read More

9 Share Reply

meganlee (meganlee) ★ 682 June 13, 2018 10:14 AM

(/meganlee)

Rewrite solution 2 using functional style

```
    public boolean canAttendMeetings(Interval[] intervals) {
        Arrays.sort(intervals, (i1, i2) -> i1.start - i2.start);
        // i start from, compare if current interval overlap with prev
```

Read More

6 Share Reply

winner1 (winner1) ★ 1 October 10, 2017 2:31 PM

(/winner1)

Yeah sorting by end time will also work.

1 Share Reply

Javava (javava) ★ 1 May 17, 2019 12:19 PM

(/javava)

In an interview, for this problem's 2nd solution, shouldn't we implement the sorting algorithm ourselves?
I think that's the whole point behind this solution...using an api to sort makes the problem meaningless...Correct me if I'm wrong. Thanks.

0 Share Reply

SHOW 3 REPLIES

Kalindigupta (kalindigupta) ★ 0 April 14, 2019 12:37 AM

(/kalindigupta)

can some one post main method for this code.
How to pass Interval[] array to the method canAttendMeetings(Interval[] intervals)?

0 Share Reply

SHOW 1 REPLY

(/sudhakarreddy)

SudhakarReddy (sudhakarreddy) ★ 2 November 12, 2018 8:08 AM

can some one post main method for this code.

How to pass Interval[] array to the method canAttendMeetings(Interval[] intervals)?

0 ▲ ▼ Share Reply

(/shuoweic)

shuoweic (shuoweic) ★ 1 October 4, 2018 8:51 PM

I'd simply rewrite overlap function into

```
bool overlap(Interval i1, Interval i2) {
    return !(i1.end <= i2.start || i1.start >= i2.end);
}
```

Read More

0 ▲ ▼ Share Reply

SHOW 1 REPLY

(/cavalos0086)

cavalos0086 (cavalos0086) ★ 6 July 7, 2018 7:41 PM

Javascript solution:

```
var canAttendMeetings = function(intervals) {
    intervals.sort((a, b) => a.start - b.start);
    for (let i=0; i < intervals.length-1;i++) {
```

Read More

0 ▲ ▼ Share Reply

(/prakashmanwani)

prakashmanwani (prakashmanwani) ★ 1 February 17, 2018 1:26 PM

```
/**
 * Definition for an interval.
 * public class Interval {
 *     int start;
```

Read More

0 ▲ ▼ Share Reply

(/piyush121)

piyush121 (piyush121) ★ 90 August 23, 2016 8:55 PM

I believe if you sort by end time then it'll also work for this problem.

0 ▲ ▼ Share Reply

< 1 2 >