252. Meeting Rooms

April 12, 2016 | 45.9K views

Average Rating: 4.87 (32 votes)

Given an array of meeting time intervals consisting of start and end times `[[s1,e1],[s2,e2],...]` (si < ei), 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

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

Copy

```
class Solution {

  public boolean canAttendMeetings(int[][] 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;
  }

  public static boolean overlap(int[] i1, int[] i2) {
    return ((i1[0] >= i2[0] && i1[0] < i2[1]) || (i2[0] >= i1[0] && i2[0] < i1[1]));
  }
}
```

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).

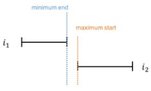

Figure 1. Two non-overlapping intervals.

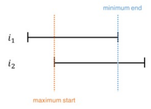

Figure 2. Two overlapping intervals.

So the condition can be rewritten as follows.

Java

Copy

```
public static boolean overlap(int[] i1, int[] i2) {
    return (Math.min(i1[1], i2[1]) >
            Math.max(i1[0], i2[0]));
}
```

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

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

Copy

```
class Solution {

  public boolean canAttendMeetings(int[][] intervals) {
    Arrays.sort(intervals, new Comparator<int[]>() {
      public int compare(int[] i1, int[] i2) {
        return i1[0] - i2[0];
      }
    });

    for (int i = 0; i < intervals.length - 1; i++) {
      if (intervals[i][1] > intervals[i + 1][0])
        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 Next

Comments: 17

Sort By

Type comment here... (Markdown is supported)

Preview Post

jiangyucara ★ 105 May 27, 2019 11:13 PM Report

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]);
```

Read More

17 Share Reply

SHOW 2 REPLIES

meganlee ★ 952 June 13, 2018 10:44 PM

Rewrite solution 2 using functional style

```
    public boolean canAttendMeetings(Interval[] intervals) {
        Arrays.sort(intervals, (i1, i2) -> i1.start - i2.start);
```

Read More

6 Share Reply

SudhakarReddy ★ 2 November 12, 2018 9:38 PM

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

winner1 ★ 0 October 11, 2017 3:01 AM Report

Yeah sorting by end time will also work.

0 Share Reply

Kalindigupta ★ 0 April 14, 2019 1:07 PM

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

shuoweic ★ 2 October 5, 2018 9:21 AM

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 ★ 7 July 8, 2018 8:11 AM Report

Javascript solution:

```
var canAttendMeetings = function(intervals) {
    intervals.sort((a, b) => a.start - b.start);
```

Read More

0 Share Reply

prakashmanwani ★ 1 February 18, 2018 2:56 AM

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

Read More

0 Share Reply

piyush121 ★ 143 August 24, 2016 9:25 AM

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

0 Share Reply

SHOW 2 REPLIES

roireshef ★ 4 a day ago Report

Python 2-liner O(nlogn):

```
def canAttendMeetings(self, intervals: List[List[int]]) -> bool:
    srtd = sorted(intervals, key=lambda x: x[0])
```

Read More

0 Share Reply