Meeting Rooms

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

For example,
Given `[[0, 30],[5, 10],[15, 20]]`,
return `false`.

Solution 1

```
public boolean canAttendMeetings(Interval[] intervals) {
  if (intervals == null)
    return false;

  // Sort the intervals by start time
  Arrays.sort(intervals, new Comparator<Interval>() {
    public int compare(Interval a, Interval b) { return a.start - b.start; }
  });

  for (int i = 1; i < intervals.length; i++)
    if (intervals[i].start < intervals[i - 1].end)
      return false;

  return true;
}
```

written by jeantimex original link here

Solution 2

The idea is pretty simple: first we sort the `intervals` in the ascending order of `start`; then we check for the overlapping of each pair of neighboring intervals. If they do, then return `false`; after we finish all the checks and have not returned `false`, just return `true`.

Sorting takes `O(nlogn)` time and the overlapping checks take `O(n)` time, so this idea is `O(nlogn)` time in total.

The code is as follows.

```
class Solution {
public:
    bool canAttendMeetings(vector<Interval>& intervals) {
        sort(intervals.begin(), intervals.end(), compare);
        int n = intervals.size();
        for (int i = 0; i < n - 1; i++)
            if (overlap(intervals[i], intervals[i + 1]))
                return false;
        return true;
    }
private:
    static bool compare(Interval& interval1, Interval& interval2) {
        return interval1.start < interval2.start;
    }
    bool overlap(Interval& interval1, Interval& interval2) {
        return interval1.end > interval2.start;
    }
};
```

written by jianchao.li.fighter original link here

Solution 3

```
public boolean canAttendMeetings(Interval[] intervals) {
        int len=intervals.length;
        if(len==0){
            return true;
        }
        int[]begin=new int[len];
        int[]stop=new int[len];
        for(int i=0;i<len;i++){
            begin[i]=intervals[i].start;
            stop[i]=intervals[i].end;
        }
        Arrays.sort(begin);
        Arrays.sort(stop);
        int endT=0;
        for(int i=1;i<len;i++){
            if(begin[i]<stop[i-1]){
                return false;
            }
        }
        return true;
}
```

written by printf_ll_ original link here