<https://www.lintcode.com/problem/920/>

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.

**Attempt 1: 2023-03-04**

**Solution 1: Sort and check if over lapping (10 min)**

/\*\*

\* Definition of Interval:

\* public class Interval {

\* int start, end;

\* Interval(int start, int end) {

\* this.start = start;

\* this.end = end;

\* }

\* }

\*/

public class Solution {

/\*\*

\* @param intervals: an array of meeting time intervals

\* @return: if a person could attend all meetings

\*/

public boolean canAttendMeetings(List<Interval> intervals) {

if(intervals.size() <= 1) {

return true;

}

Collections.sort(intervals, (a, b) -> a.start - b.start);

for(int i = 1; i < intervals.size(); i++) {

if(intervals.get(i - 1).end > intervals.get(i).start) {

return false;

}

}

return true;

}

}

Time Complexity:O(nlogn), sorting take nlogn time

Space Complexity:O(1)

**Refer to**

<https://www.lintcode.com/problem/920/solution/19526>

按照区间start从小到大排序，满足题目要求的区间应该是没有交集的。所以要比较看当前区间end是否大于下一个区间start, 并且不同区间start应该不相同。

/\*\*

\* Definition of Interval:

\* public classs Interval {

\* int start, end;

\* Interval(int start, int end) {

\* this.start = start;

\* this.end = end;

\* }

\* }

\*/

public class Solution {

/\*\*

\* @param intervals: an array of meeting time intervals

\* @return: if a person could attend all meetings

\*/

public boolean canAttendMeetings(List<Interval> intervals) {

if (intervals == null || intervals.isEmpty()) {

return true;

}

Collections.sort(intervals, Comparator.comparingInt(interval -> interval.start));

for (int idx = 1; idx < intervals.size(); idx++) {

if (intervals.get(idx).start < intervals.get(idx - 1).end) {

return false;

}

}

return true;

}

}

**复杂度分析**

* 时间复杂度 : *O*(*n*log*n*)。时间复杂度由排序决定。一旦排序完成，只需要*O*(*n*)的时间来判断交叠。
* 空间复杂度 : *O*(1)。没有使用额外空间。