

Conflicting Appointments (medium)

We'll cover the following ^

- Problem Statement
- Try it yourself
- Solution
- Code
 - Time complexity
 - Space complexity
- Similar Problems

Problem Statement

Given an array of intervals representing 'N' appointments, find out if a person can **attend all the appointments**.

Example 1:

```
Appointments: [[1,4], [2,5], [7,9]]
Output: false
Explanation: Since [1,4] and [2,5] overlap, a person cannot attend both of these appointments.
```

Example 2:

```
Appointments: [[6,7], [2,4], [8,12]]
Output: true
Explanation: None of the appointments overlap, therefore a person can attend all of them.
```

Example 3:

```
Appointments: [[4,5], [2,3], [3,6]]
Output: false
Explanation: Since [4,5] and [3,6] overlap, a person cannot attend both of these appointments.
```

Try it yourself

Try solving this question here:



educative

 Java

 Python3

 JS

 C++

```
1 def can_attend_all_appointments(intervals):
2     # TODO: Write your code here
3     return False
4
5
6 def main():
7     print("Can attend all appointments: " + str(can_attend_all_appointments([[1, 4], [2, 5],
8     print("Can attend all appointments: " + str(can_attend_all_appointments([[6, 7], [2, 4],
9     print("Can attend all appointments: " + str(can_attend_all_appointments([[4, 5], [2, 3],
10
11
12 main()
13
```





Solution

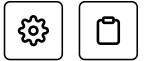
The problem follows the Merge Intervals

(<https://www.educative.io/collection/page/5668639101419520/5671464854355968/5652017242439680/>) pattern. We can sort all the intervals by start time and then check if any two intervals overlap. A person will not be able to attend all appointments if any two appointments overlap.

Code

Here is what our algorithm will look like:

 Java	 Python3	 C++	 JS
<pre>1 def can_attend_all_appointments(intervals): 2 intervals.sort(key=lambda x: x[0]) 3 start, end = 0, 1 4 for i in range(1, len(intervals)): 5 if intervals[i][start] < intervals[i-1][end]: 6 # please note the comparison above, it is "<" and not "<=" 7 # while merging we needed "<=" comparison, as we will be merging the two 8 # intervals having condition "intervals[i][start] == intervals[i - 1][end]" but 9 # such intervals don't represent conflicting appointments as one starts right 10 # after the other 11 return False 12 return True 13 14 15 def main(): 16 print("Can attend all appointments: " + str(can_attend_all_appointments([[1, 4], [2, 5], 17 print("Can attend all appointments: " + str(can_attend_all_appointments([[6, 7], [2, 4], 18 print("Can attend all appointments: " + str(can_attend_all_appointments([[4, 5], [2, 3], 19 20 21 main() 22</pre>			



The time complexity of the above algorithm is $O(N * \log N)$, where 'N' is the total number of appointments. Though we are iterating the intervals only once, our algorithm will take $O(N * \log N)$ since we need to sort them in the beginning.

Space complexity

The space complexity of the above algorithm will be $O(N)$, which we need for sorting. For Java, `Arrays.sort()` uses Timsort (<https://en.wikipedia.org/wiki/Timsort>), which needs $O(N)$ space.

Similar Problems

Problem 1: Given a list of appointments, find all the conflicting appointments.

Example:

```
Appointments: [[4,5], [2,3], [3,6], [5,7], [7,8]]
Output:
[4,5] and [3,6] conflict.
[3,6] and [5,7] conflict.
```

[← Back](#)

[Next →](#)

Intervals Intersection (medium)

Problem Challenge 1

☒ **Mark as Completed**



Report an
Issue



Ask a Question

(https://discuss.educative.io/tag/conflicting-appointments-medium__pattern-merge-intervals__grokking-the-coding-interview-patterns-for-coding-questions)