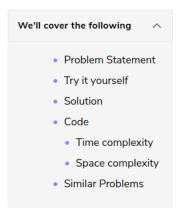




Conflicting Appointments (medium)



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:

```
# Java  

def can_attend_all_appointments(intervals):

# TODO: Write your code here

return False

def main():

print("Can attend all appointments: " + str(can_attend_all_appointments([[1, 4], [2, 5], [7, 9]])))

print("Can attend all appointments: " + str(can_attend_all_appointments([[6, 7], [2, 4], [8, 12]])))

print("Can attend all appointments: " + str(can_attend_all_appointments([[4, 5], [2, 3], [3, 6]])))

main()

Run

Save Reset (3)
```

Solution

The problem follows the Merge Intervals 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 1

Here is what our algorithm will look like:

Time complexity

The time complexity of the above algorithm is O(N*logN), where 'N' is the total number of appointments. Though we are iterating the intervals only once, our algorithm will take O(N*logN) 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, which needs O(N) space.

Similar Problems

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

Example:

← Back

Intervals Intersection (medium)

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



Next →

Problem Challenge 1

Completed