Conflicting Appointments (medium)

We'll cover the following

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 atten
d both of these appointments.
```

Example 2:

```
Appointments: [[6,7], [2,4], [8,12]]
Output: true
Explanation: None of the appointments overlap, therefore a perso
n 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 atten
d both of these appointments.
```

Try it yourself

Try solving this question here:

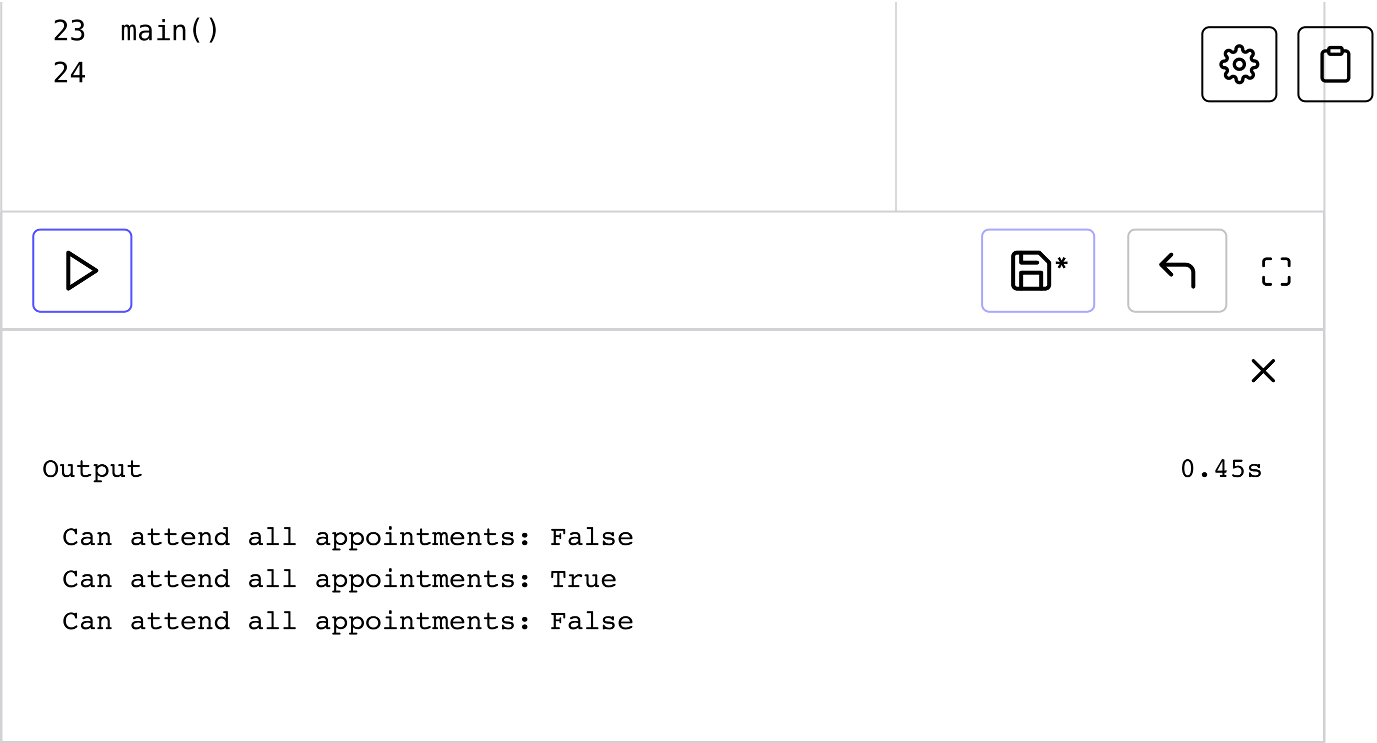
Java Python3 JS C++

```
def can_attend_all_appointments(intervals):
  # TODO: Write your code here
  if len(intervals)<2:
    return True
  # sort
  intervals.sort(key=lambda x:x[0])
  # iterate
  # find overlap - start > next.end
  for i in range(1,len(intervals)):
    if intervals[i][0]<intervals[i-1][1]:
      return False
  return True




def main():
  print("Can attend all appointments: " + str(ca
  print("Can attend all appointments: " + str(ca
  print("Can attend all appointments: " + str(ca


```



```
main()

```

Output 0.45s

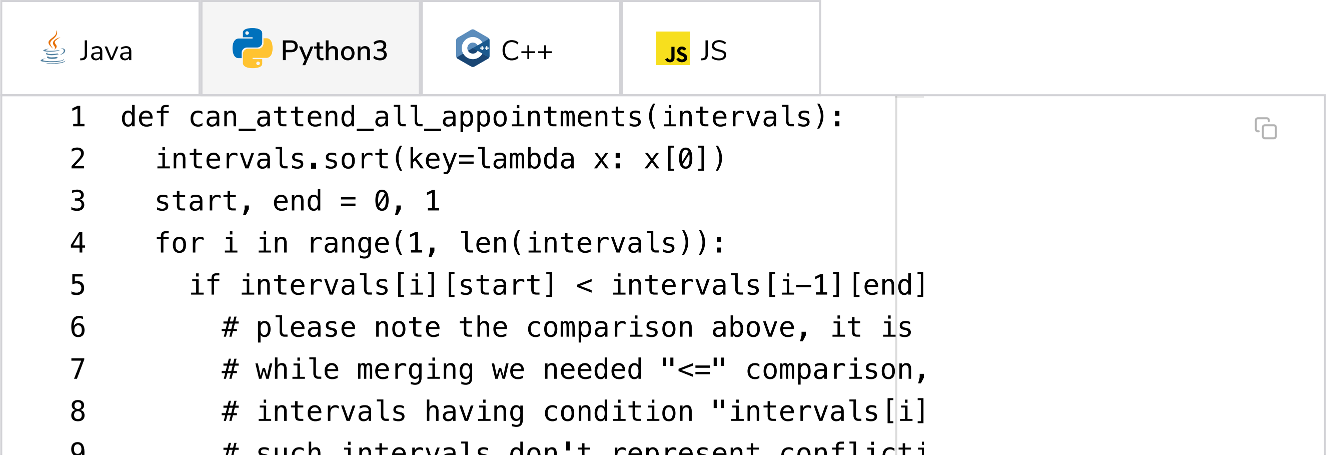
```
Can attend all appointments: False
Can attend all appointments: True
Can attend all appointments: False
```

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:



```
def can_attend_all_appointments(intervals):
  intervals.sort(key=lambda x: x[0])
  start, end = 0, 1
  for i in range(1, len(intervals)):
    if intervals[i][start] < intervals[i-1][end]
      # please note the comparison above, it is
      # while merging we needed "<=" comparison,
      # intervals having condition "intervals[i]
      # such intervals don't represent conflicti
      # such intervals don't represent conflict
      # after the other
      return False
  return True


def main():
  print("Can attend all appointments: " + str(ca
  print("Can attend all appointments: " + str(ca
  print("Can attend all appointments: " + str(ca


main()

```

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 (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

Intervals Intersection (medium)

Next →

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)