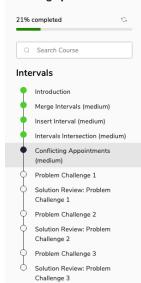


Grokking the Coding Interview: Patterns for Coding Questions



Pattern: Cyclic Sort



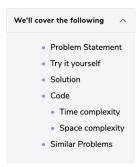
Pattern: In-place Reversal of a LinkedList



Pattern: Tree Breadth First Search



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:

```
Python3 Js JS
                               G C++
  constructor(start, end) {
    this.start = start;
    this.end = end;
  print_interval() {
    process.stdout.write(`[${this.start}, ${this.end}]`);
const can_attend_all_appointments = function(intervals) {
console.log(`Can attend all appointments: ${can_attend_all_appointments([
  new Interval(2, 5),
1)}`);
console.log(`Can attend all appointments: ${can_attend_all_appointments([
  new Interval(2, 4),
  new Interval(8, 12),
                                                                               SAVE
                                                                                          RESET []
```

Solution

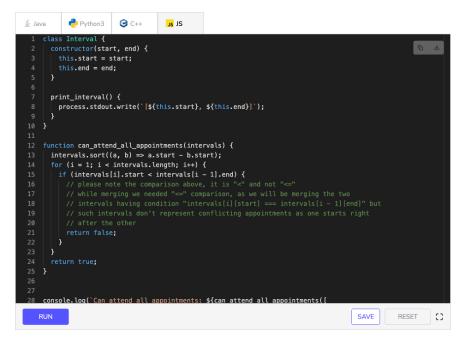
The problem follows the Merge Intervals pattern. We can sort all the intervals by start time and then check if any two intervals overlap. A person will not be able to attend all appointments if any two appointments overlap.

Code

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

Intervals Intersection (medium)

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

Example:

```
Appointments: [[4,5], [2,3], [3,6], [5,7], [7,8]]
[4,5] and [3,6] conflict.
[3,6] and [5,7] conflict.
Interviewing soon? We've partnered with Hired so that companies apply to you instead of you
```

applying to them. See how ①

