Back To Course Home

Grokking the Coding Interview: Patterns for Coding Questions

28% completed

Search Course

Problem Challenge 1
Solution Review: Problem Challenge 1
Problem Challenge 2
Solution Review: Problem Challenge 2

Solution Review: Problem Challenge 1

We'll cover the following

- Minimum Meeting Rooms (hard)
- Solution
- Code
  - Time complexity
  - Space complexity
- Similar Problems

Minimum Meeting Rooms (hard)

Given a list of intervals representing the start and end time of 'N' meetings, find the **minimum number of rooms** required to **hold all the meetings**.

Example 1:

```
Meetings: [[1,4], [2,5], [7,9]]
Output: 2
Explanation: Since [1,4] and [2,5] overlap, we need two rooms to hold these two meet
ings. [7,9] can
occur in any of the two rooms later.
```

Example 2:

```
Meetings: [[6,7], [2,4], [8,12]]
Output: 1
Explanation: None of the meetings overlap, therefore we only need one room to hold a
ll meetings.
```

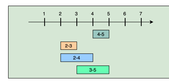
Example 3:

```
Meetings: [[1,4], [2,3], [3,6]]
Output:2
Explanation: Since [1,4] overlaps with the other two meetings [2,3] and [3,6], we ne
ed two rooms to
hold all the meetings.
```

Example 4:

```
Meetings: [[4,5], [2,3], [2,4], [3,5]]
Output: 2
Explanation: We will need one room for [2,3] and [3,5], and another room for [2,4] a
nd [4,5].

Here is a visual representation of Example 4:
```



Solution

Let's take the above-mentioned example (4) and try to follow our Merge Intervals approach:

Meetings: [[4,5], [2,3], [2,4], [3,5]]

Step 1: Sorting these meetings on their start time will give us: [[2,3], [2,4], [3,5], [4,5]]

Step 2: Merging overlapping meetings:

- [2,3] overlaps with [2,4], so after merging we'll have => [[2,4], [3,5], [4,5]]
- [2,4] overlaps with [3,5], so after merging we'll have => [[2,5], [4,5]]
- [2,5] overlaps [4,5], so after merging we'll have => [2,5]

Since all the given meetings have merged into one big meeting ([2,5]), does this mean that they all are overlapping and we need a minimum of four rooms to hold these meetings? You might have already guessed that the answer is NO! As we can clearly see, some meetings are mutually exclusive. For example, [2,3] and [3,5] do not overlap and can happen in one room. So, to correctly solve our problem, we need to keep track of the mutual exclusiveness of the overlapping meetings.

Here is what our strategy will look like:

1. We will sort the meetings based on start time.
2. We will schedule the first meeting (let's call it `m1`) in one room (let's call it `r1`).
3. If the next meeting `m2` is not overlapping with `m1`, we can safely schedule it in the same room `r1`.
4. If the next meeting `m3` is overlapping with `m2` we can't use `r1`, so we will schedule it in another room (let's call it `r2`).
5. Now if the next meeting `m4` is overlapping with `m3`, we need to see if the room `r1` has become free. For this, we need to keep track of the end time of the meeting happening in it. If the end time of `m2` is before the start time of `m4`, we can use that room `r1`, otherwise, we need to schedule `m4` in another room `r3`.

1. We will sort the meetings based on start time.
2. We will schedule the first meeting (let's call it `m1`) in one room (let's call it `r1`).
3. If the next meeting `m2` is not overlapping with `m1`, we can safely schedule it in the same room `r1`.
4. If the next meeting `m3` is overlapping with `m2` we can't use `r1`, so we will schedule it in another room (let's call it `r2`).
5. Now if the next meeting `m4` is overlapping with `m3`, we need to see if the room `r1` has become free. For this, we need to keep track of the end time of the meeting happening in it. If the end time of `m2` is before the start time of `m4`, we can use that room `r1`, otherwise, we need to schedule `m4` in another room `r3`.

1. We will sort the meetings based on start time.
2. We will schedule the first meeting (let's call it `m1`) in one room (let's call it `r1`).
3. If the next meeting `m2` is not overlapping with `m1`, we can safely schedule it in the same room `r1`.
4. If the next meeting `m3` is overlapping with `m2` we can't use `r1`, so we will schedule it in another room (let's call it `r2`).
5. Now if the next meeting `m4` is overlapping with `m3`, we need to see if the room `r1` has become free. For this, we need to keep track of the end time of the meeting happening in it. If the end time of `m2` is before the start time of `m4`, we can use that room `r1`, otherwise, we need to schedule `m4` in another room `r3`.

1. We will sort the meetings based on start time.
2. We will schedule the first meeting (let's call it `m1`) in one room (let's call it `r1`).
3. If the next meeting `m2` is not overlapping with `m1`, we can safely schedule it in the same room `r1`.
4. If the next meeting `m3` is overlapping with `m2` we can't use `r1`, so we will schedule it in another room (let's call it `r2`).
5. Now if the next meeting `m4` is overlapping with `m3`, we need to see if the room `r1` has become free. For this, we need to keep track of the end time of the meeting happening in it. If the end time of `m2` is before the start time of `m4`, we can use that room `r1`, otherwise, we need to schedule `m4` in another room `r3`.

1. We will sort the meetings based on start time.
2. We will schedule the first meeting (let's call it `m1`) in one room (let's call it `r1`).
3. If the next meeting `m2` is not overlapping with `m1`, we can safely schedule it in the same room `r1`.
4. If the next meeting `m3` is overlapping with `m2` we can't use `r1`, so we will schedule it in another room (let's call it `r2`).
5. Now if the next meeting `m4` is overlapping with `m3`, we need to see if the room `r1` has become free. For this, we need to keep track of the end time of the meeting happening in it. If the end time of `m2` is before the start time of `m4`, we can use that room `r1`, otherwise, we need to schedule `m4` in another room `r3`.

1. We will sort the meetings based on start time.
2. We will schedule the first meeting (let's call it `m1`) in one room (let's call it `r1`).
3. If the next meeting `m2` is not overlapping with `m1`, we can safely schedule it in the same room `r1`.
4. If the next meeting `m3` is overlapping with `m2` we can't use `r1`, so we will schedule it in another room (let's call it `r2`).
5. Now if the next meeting `m4` is overlapping with `m3`, we need to see if the room `r1` has become free. For this, we need to keep track of the end time of the meeting happening in it. If the end time of `m2` is before the start time of `m4`, we can use that room `r1`, otherwise, we need to schedule `m4` in another room `r3`.

1. We will sort the meetings based on start time.
2. We will schedule the first meeting (let's call it `m1`) in one room (let's call it `r1`).
3. If the next meeting `m2` is not overlapping with `m1`, we can safely schedule it in the same room `r1`.
4. If the next meeting `m3` is overlapping with `m2` we can't use `r1`, so we will schedule it in another room (let's call it `r2`).
5. Now if the next meeting `m4` is overlapping with `m3`, we need to see if the room `r1` has become free. For this, we need to keep track of the end time of the meeting happening in it. If the end time of `m2` is before the start time of `m4`, we can use that room `r1`, otherwise, we need to schedule `m4` in another room `r3`.

1. We will sort the meetings based on start time.
2. We will schedule the first meeting (let's call it `m1`) in one room (let's call it `r1`).
3. If the next meeting `m2` is not overlapping with `m1`, we can safely schedule it in the same room `r1`.
4. If the next meeting `m3` is overlapping with `m2` we can't use `r1`, so we will schedule it in another room (let's call it `r2`).
5. Now if the next meeting `m4` is overlapping with `m3`, we need to see if the room `r1` has become free. For this, we need to keep track of the end time of the meeting happening in it. If the end time of `m2` is before the start time of `m4`, we can use that room `r1`, otherwise, we need to schedule `m4` in another room `r3`.

1. We will sort the meetings based on start time.
2. We will schedule the first meeting (let's call it `m1`) in one room (let's call it `r1`).
3. If the next meeting `m2` is not overlapping with `m1`, we can safely schedule it in the same room `r1`.
4. If the next meeting `m3` is overlapping with `m2` we can't use `r1`, so we will schedule it in another room (let's call it `r2`).
5. Now if the next meeting `m4` is overlapping with `m3`, we need to see if the room `r1` has become free. For this, we need to keep track of the end time of the meeting happening in it. If the end time of `m2` is before the start time of `m4`, we can use that room `r1`, otherwise, we need to schedule `m4` in another room `r3`.

1. We will sort the meetings based on start time.
2. We will schedule the first meeting (let's call it `m1`) in one room (let's call it `r1`).
3. If the next meeting `m2` is not overlapping with `m1`, we can safely schedule it in the same room `r1`.
4. If the next meeting `m3` is overlapping with `m2` we can't use `r1`, so we will schedule it in another room (let's call it `r2`).
5. Now if the next meeting `m4` is overlapping with `m3`, we need to see if the room `r1` has become free. For this, we need to keep track of the end time of the meeting happening in it. If the end time of `m2` is before the start time of `m4`, we can use that room `r1`, otherwise, we need to schedule `m4` in another room `r3`.

1. We will sort the meetings based on start time.
2. We will schedule the first meeting (let's call it `m1`) in one room (let's call it `r1`).
3. If the next meeting `m2` is not overlapping with `m1`, we can safely schedule it in the same room `r1`.
4. If the next meeting `m3` is overlapping with `m2` we can't use `r1`, so we will schedule it in another room (let's call it `r2`).
5. Now if the next meeting `m4` is overlapping with `m3`, we need to see if the room `r1` has become free. For this, we need to keep track of the end time of the meeting happening in it. If the end time of `m2` is before the start time of `m4`, we can use that room `r1`, otherwise, we need to schedule `m4` in another room `r3`.

1. We will sort the meetings based on start time.
2. We will schedule the first meeting (let's call it `m1`) in one room (let's call it `r1`).
3. If the next meeting `m2` is not overlapping with `m1`, we can safely schedule it in the same room `r1`.
4. If the next meeting `m3` is overlapping with `m2` we can't use `r1`, so we will schedule it in another room (let's call it `r2`).
5. Now if the next meeting `m4` is overlapping with `m3`, we need to see if the room `r1` has become free. For this, we need to keep track of the end time of the meeting happening in it. If the end time of `m2` is before the start time of `m4`, we can use that room `r1`, otherwise, we need to schedule `m4` in another room `r3`.