

Grokking the Coding Interview: Patterns for Coding Questions

19% completed

Intervals

- Introduction
- Merge Intervals (medium)**
- Insert Interval (medium)
- Intervals Intersection (medium)
- Conflicting Appointments (medium)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2
- Problem Challenge 3
- Solution Review: Problem Challenge 3

Pattern: Cyclic Sort

- Introduction
- Cyclic Sort (easy)
- Find the Missing Number (easy)
- Find all Missing Numbers (easy)
- Find the Duplicate Number (easy)
- Find all Duplicate Numbers (easy)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2
- Problem Challenge 3
- Solution Review: Problem Challenge 3

Pattern: In-place

Reversal of a LinkedList

- Introduction
- Reverse a LinkedList (easy)
- Reverse a Sub-list (medium)
- Reverse every K-element Sub-list (medium)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2

Pattern: Tree Breadth First Search

- Introduction
- Binary Tree Level Order Traversal (easy)
- Reverse Level Order Traversal (easy)
- Zigzag Traversal (medium)
- Level Averages in a Binary Tree (easy)
- Minimum Depth of a Binary Tree (easy)

Merge Intervals (medium)

We'll cover the following

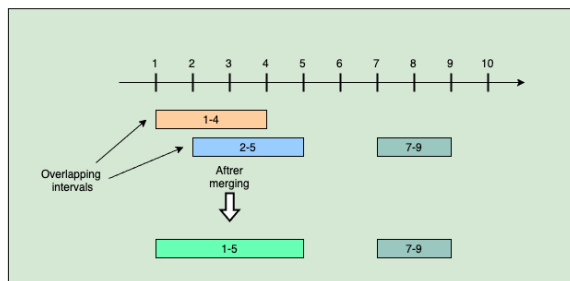
- Problem Statement
- Try it yourself
- Solution
- Code
 - Time complexity
 - Space complexity
- Similar Problems

Problem Statement

Given a list of intervals, **merge all the overlapping intervals** to produce a list that has only mutually exclusive intervals.

Example 1:

```
Intervals: [[1,4], [2,5], [7,9]]
Output: [[1,5], [7,9]]
Explanation: Since the first two intervals [1,4] and [2,5] overlap, we merged them into one [1,5].
```



Example 2:

```
Intervals: [[6,7], [2,4], [5,9]]
Output: [[2,4], [5,9]]
Explanation: Since the intervals [6,7] and [5,9] overlap, we merged them into one [5,9].
```

Example 3:

```
Intervals: [[1,4], [2,6], [3,5]]
Output: [[1,6]]
Explanation: Since all the given intervals overlap, we merged them into one.
```

Try it yourself

Try solving this question here:

```
Java Python3 JS C++

14 merged = []
15 // TODO: Write your code here
16 return merged;
17 };
18
19 merged_intervals = merge([new Interval(1, 4), new Interval(2, 5), new Interval(7, 9)]);
20 result = "";
21 for(i=0; i < merged_intervals.length; i++) {
22     result += merged_intervals[i].get_interval() + " ";
23 }
24 console.log('Merged intervals: ${result}')
25
26 merged_intervals = merge([new Interval(6, 7), new Interval(2, 4), new Interval(5, 9)]);
27 result = "";
28 for(i=0; i < merged_intervals.length; i++) {
29     result += merged_intervals[i].get_interval() + " ";
30 }
31 console.log('Merged intervals: ${result}')
32
33 merged_intervals = merge([new Interval(1, 4), new Interval(2, 6), new Interval(3, 5)]);
34 result = "";
35 for(i=0; i < merged_intervals.length; i++) {
36     result += merged_intervals[i].get_interval() + " ";
37 }
```

- Level Order Successor (easy)
- Connect Level Order Siblings (medium)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2

Pattern: Tree Depth

First Search

- Introduction
- Binary Tree Path Sum (easy)
- All Paths for a Sum (medium)
- Sum of Path Numbers (medium)
- Path With Given Sequence (medium)
- Count Paths for a Sum (medium)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2

Pattern: Two Heaps

- Introduction
- Find the Median of a Number Stream (medium)
- Sliding Window Median (hard)
- Maximize Capital (hard)
- Problem Challenge 1
- Solution Review: Problem Challenge 1

Pattern: Subsets

- Introduction
- Subsets (easy)
- Subsets With Duplicates (easy)
- Permutations (medium)
- String Permutations by changing case (medium)
- Balanced Parentheses (hard)
- Unique Generalized Abbreviations (hard)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2
- Problem Challenge 3
- Solution Review: Problem Challenge 3

Pattern: Modified

Binary Search

- Introduction
- Order-agnostic Binary Search (easy)
- Ceiling of a Number (medium)
- Next Letter (medium)
- Number Range (medium)
- Search in a Sorted Infinite Array (medium)
- Minimum Difference Element (medium)
- Bitonic Array Maximum (easy)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2

```
37 for (i = 0; i < merged_intervals.length; i++) {
38   result += merged_intervals[i].get_interval() + " ";
39 }
40 console.log('Merged intervals: ${result}')
41
```

RUN

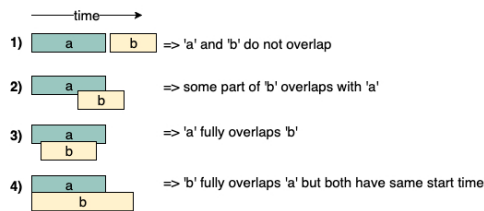
SAVE

RESET

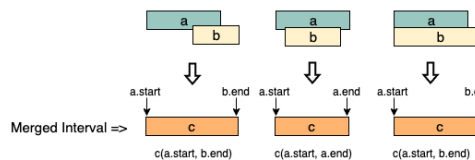


Solution

Let's take the example of two intervals ('a' and 'b') such that $a.start \leq b.start$. There are four possible scenarios:



Our goal is to merge the intervals whenever they overlap. For the above-mentioned three overlapping scenarios (2, 3, and 4), this is how we will merge them:



The diagram above clearly shows a merging approach. Our algorithm will look like this:

- Sort the intervals on the start time to ensure $a.start \leq b.start$
- If 'a' overlaps 'b' (i.e. $b.start \leq a.end$), we need to merge them into a new interval 'c' such that:

```
c.start = a.start
c.end = max(a.end, b.end)
```

- We will keep repeating the above two steps to merge 'c' with the next interval if it overlaps with 'c'.

Code

Here is what our algorithm will look like:

Java

Python3

C++

JS

```
1 class Interval {
2   constructor(start, end) {
3     this.start = start;
4     this.end = end;
5   }
6
7   print_interval() {
8     process.stdout.write(`${this.start}, ${this.end}`);
9   }
10 }
11
12
13 function merge(intervals) {
14   if (intervals.length < 2) {
15     return intervals;
16   }
17   // sort the intervals on the start time
18   intervals.sort((a, b) => a.start - b.start);
19
20   const mergedIntervals = [];
21   let start = intervals[0].start,
22       end = intervals[0].end;
23   for (i = 1; i < intervals.length; i++) {
24     const interval = intervals[i];
25     if (interval.start <= end) { // overlapping intervals, adjust the 'end'
26       end = Math.max(interval.end, end);
27     } else { // non-overlapping interval, add the previous interval and reset
28       mergedIntervals.push(new Interval(start, end));
```

RUN

SAVE

RESET



Close

Output

1.946s

```
Merged intervals: [1, 5][7, 9]
Merged intervals: [2, 4][5, 9]
Merged intervals: [1, 6]
```

Time complexity

MW

Explore

Tracks

My Courses

Edpresso

Refer a Friend

Create

Problem Challenge 2

Solution Review: Problem Challenge 2

Problem Challenge 3

Solution Review: Problem Challenge 3

Pattern: Bitwise XOR

Introduction

Single Number (easy)

Two Single Numbers (medium)

Complement of Base 10 Number (medium)

Problem Challenge 1

Solution Review: Problem Challenge 1

Pattern: Top 'K' Elements

Introduction

Top 'K' Numbers (easy)

Kth Smallest Number (easy)

'K' Closest Points to the Origin (easy)

Connect Ropes (easy)

Top 'K' Frequent Numbers (medium)

Frequency Sort (medium)

Kth Largest Number in a Stream (medium)

'K' Closest Numbers (medium)

Maximum Distinct Elements (medium)

Sum of Elements (medium)

Rearrange String (hard)

Problem Challenge 1

Solution Review: Problem Challenge 1

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

Space complexity

The space complexity of the above algorithm will be $O(N)$ as we need to return a list containing all the merged intervals. We will also need $O(N)$ space for sorting. For Java, depending on its version, `Collection.sort()` either uses Merge sort or Timsort, and both these algorithms need $O(N)$ space. Overall, our algorithm has a space complexity of $O(N)$.

Similar Problems #

Problem 1: Given a set of intervals, find out if any two intervals overlap.

Example:

```
Intervals: [[1,4], [2,5], [7,9]]
Output: true
Explanation: Intervals [1,4] and [2,5] overlap
```

Solution: We can follow the same approach as discussed above to find if any two intervals overlap.

Interviewing soon? We've partnered with Hired so that companies apply to you instead of you applying to them. [See how](#)

← Back

Introduction

MARK AS COMPLETED

Next →

Insert Interval (medium)

Report an Issue