

Grokking the Coding Interview: Patterns for Coding Questions

23% 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)

Solution Review: Problem Challenge 2

We'll cover the following

- Maximum CPU Load (hard)
- Solution
- Code
 - Time complexity
 - Space complexity

Maximum CPU Load (hard)

We are given a list of Jobs. Each job has a Start time, an End time, and a CPU load when it is running. Our goal is to find the **maximum CPU load** at any time if all the **jobs are running on the same machine**.

Example 1:

```
Jobs: [[1,4,3], [2,5,4], [7,9,6]]
Output: 7
Explanation: Since [1,4,3] and [2,5,4] overlap, their maximum CPU load (3+4=7) will be when both the jobs are running at the same time i.e., during the time interval (2,4).
```

Example 2:

```
Jobs: [[6,7,10], [2,4,11], [8,12,15]]
Output: 15
Explanation: None of the jobs overlap, therefore we will take the maximum load of any job which is 15.
```

Example 3:

```
Jobs: [[1,4,2], [2,4,1], [3,6,5]]
Output: 8
Explanation: Maximum CPU load will be 8 as all jobs overlap during the time interval [3,4].
```

Solution

The problem follows the [Merge Intervals](#) pattern and can easily be converted to [Minimum Meeting Rooms](#). Similar to 'Minimum Meeting Rooms' where we were trying to find the maximum number of meetings happening at any time, for 'Maximum CPU Load' we need to find the maximum number of jobs running at any time. We will need to keep a running count of the maximum CPU load at any time to find the overall maximum load.

Code

Here is what our algorithm will look like:

```
Java Python3 C++ JS JS
1 const Heap = require('./collections/heap'); //http://www.collectionsjs.com
2
3 class Job {
4   constructor(start, end, cpuLoad) {
5     this.start = start;
6     this.end = end;
7     this.cpuLoad = cpuLoad;
8   }
9
10
11 function find_max_cpu_load(jobs) {
12   // sort the jobs by start time
13   jobs.sort((a, b) => a.start - b.start);
14
15   let maxCPULoad = 0,
16       currentCPULoad = 0;
17   const minHeap = new Heap([], null, ((a, b) => b.end - a.end));
18
19   for (j = 0; j < jobs.length; j++) {
20     // remove all the jobs that have ended
21     while (minHeap.length > 0 && jobs[j].start >= minHeap.peek().end) {
22       currentCPULoad -= minHeap.pop().cpuLoad;
23     }
24     // add the current job into min_heap
25     minHeap.push(jobs[j]);
26     currentCPULoad += jobs[j].cpuLoad;
27     maxCPULoad = Math.max(maxCPULoad, currentCPULoad);
28   }
29 }
```

RUN

SAVE

RESET



Close

MW

Explore

Tracks

My Courses

Edpresso

Refer a Friend

Create

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

Output2.208s

Maximum CPU load at any time: 7
Maximum CPU load at any time: 15
Maximum CPU load at any time: 8

Time complexity

The time complexity of the above algorithm is $O(N * \log N)$, where 'N' is the total number of jobs. This is due to the sorting that we did in the beginning. Also, while iterating the jobs, we might need to poll/offer jobs to the priority queue. Each of these operations can take $O(\log N)$. Overall our algorithm will take $O(N \log N)$.

Space complexity

The space complexity of the above algorithm will be $O(N)$, which is required for sorting. Also, in the worst case, we have to insert all the jobs into the priority queue (when all jobs overlap) which will also take $O(N)$ space. The overall space complexity of our algorithm is $O(N)$.

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

← Back

Problem Challenge 2

MARK AS COMPLETED

Next →

Problem Challenge 3

Report an Issue