

Grokking the Coding Interview: Patterns for Coding Questions

17% completed

- Happy Number (medium)
- Middle of the LinkedList (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: Merge 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

Solution Review: Problem Challenge 2

We'll cover the following

- Rearrange a LinkedList (medium)
- Solution
 - Code
 - Time Complexity
 - Space Complexity

Rearrange a LinkedList (medium)

Given the head of a Singly LinkedList, write a method to modify the LinkedList such that the **nodes from the second half of the LinkedList are inserted alternately to the nodes from the first half in reverse order**. So if the LinkedList has nodes 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> null, your method should return 1 -> 6 -> 2 -> 5 -> 3 -> 4 -> null.

Your algorithm should not use any extra space and the input LinkedList should be modified in-place.

Example 1:

```
Input: 2 -> 4 -> 6 -> 8 -> 10 -> 12 -> null
Output: 2 -> 12 -> 4 -> 10 -> 6 -> 8 -> null
```

Example 2:

```
Input: 2 -> 4 -> 6 -> 8 -> 10 -> null
Output: 2 -> 10 -> 4 -> 8 -> 6 -> null
```

Solution

This problem shares similarities with [Palindrome LinkedList](#). To rearrange the given LinkedList we will follow the following steps:

- We can use the **Fast & Slow pointers** method similar to [Middle of the LinkedList](#) to find the middle node of the LinkedList.
- Once we have the middle of the LinkedList, we will reverse the second half of the LinkedList.
- Finally, we'll iterate through the first half and the reversed second half to produce a LinkedList in the required order.

Code

Here is what our algorithm will look like:

```
Java Python3 C++ JS

1 class Node {
2   constructor(value, next = null) {
3     this.value = value;
4     this.next = next;
5   }
6
7
8   print_list() {
9     temp = this;
10    while (temp !== null) {
11      process.stdout.write(`${temp.value} `);
12      temp = temp.next;
13    }
14    console.log();
15  }
16 }
17
18 function reorder(head) {
19   if (head === null || head.next === null) {
20     return;
21   }
22
23   // find middle of the LinkedList
24   let slow = head,
25       fast = head;
26   while (fast !== null && fast.next !== null) {
27     slow = slow.next;
28     fast = fast.next.next;
29   }
```

RUN

SAVE

RESET



Close

Output

2.600s

2 12 4 10 6 8

Tracks

My Courses

Edpresso

Refer a Friend

Create

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)

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

Time Complexity

The above algorithm will have a time complexity of $O(N)$ where 'N' is the number of nodes in the LinkedList.

Space Complexity

The algorithm runs in constant space $O(1)$.

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