

# Grokking the Coding Interview: Patterns for Coding Questions



## Pattern: Merge Intervals

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

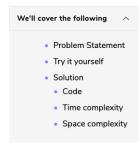
## Pattern: Cyclic Sort



## Pattern: In-place Reversal of a LinkedList



# Middle of the LinkedList (easy)



#### **Problem Statement**

Given the head of a Singly LinkedList, write a method to return the middle node of the LinkedList.

If the total number of nodes in the LinkedList is even, return the second middle node.

### Example 1:

```
Input: 1 -> 2 -> 3 -> 4 -> 5 -> null
Output: 3
```

### Example 2:

```
Input: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> null
Output: 4
```

#### Example 3:

```
Input: 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7 -> null
Output: 4
```

## Try it yourself

Try solving this question here:

```
Python3
                                           G C++
    class ListNode {
      ListNode next:
      ListNode(int value) {
         this.value = value;
    class MiddleOfLinkedList {
      public static ListNode findMiddle(ListNode head) {
         return head;
      public static void main(String[] args) {
         ListNode head = new ListNode(1);
head.next = new ListNode(2);
         head.next.next = new ListNode(3);
         head.next.next.next = new ListNode(4);
         head.next.next.next.next = new ListNode(5);
System.out.println("Middle Node: " + MiddleOfLinkedList.findMiddle(head).value);
25
26
         head.next.next.next.next = new ListNode(6);
System.out.println("Middle Node: " + MiddleOfLinkedList.findMiddle(head).value);
                                                                                                                               03
```

## Solution

One brute force strategy could be to first count the number of nodes in the LinkedList and then find the middle node in the second iteration. Can we do this in one iteration?

We can use the **Fast & Slow pointers** method such that the fast pointer is always twice the nodes ahead of the slow pointer. This way, when the fast pointer reaches the end of the LinkedList, the slow pointer will be pointing at the middle node.

### Code

Here is what our algorithm will look like:



## 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)

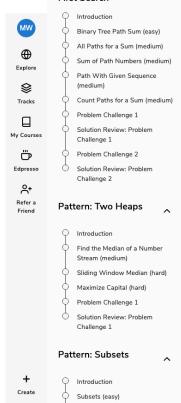
Problem Challenge 1

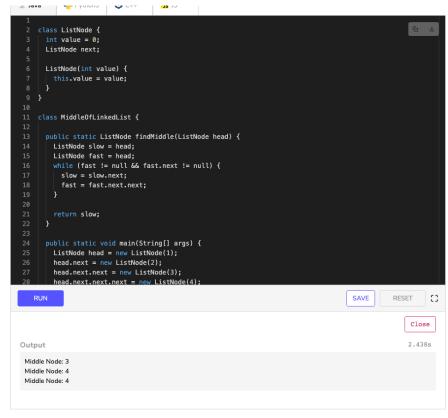
Solution Review: Problem Challenge 1

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  $\quad\times$  applying to them. See how  $\odot$ 

