

Grokking the Coding Interview: Patterns for Coding Questions

15% 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

Middle of the LinkedList (easy)

We'll cover the following

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

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:

JavaPython3JSJSC++

```
1
2 class ListNode {
3     int value = 0;
4     ListNode next;
5
6     ListNode(int value) {
7         this.value = value;
8     }
9 }
10
11 class MiddleOfLinkedList {
12
13     public static ListNode findMiddle(ListNode head) {
14         // TODO: Write your code here
15         return head;
16     }
17
18     public static void main(String[] args) {
19         ListNode head = new ListNode(1);
20         head.next = new ListNode(2);
21         head.next.next = new ListNode(3);
22         head.next.next.next = new ListNode(4);
23         head.next.next.next.next = new ListNode(5);
24         System.out.println("Middle Node: " + MiddleOfLinkedList.findMiddle(head).value);
25
26         head.next.next.next.next.next = new ListNode(6);
27         System.out.println("Middle Node: " + MiddleOfLinkedList.findMiddle(head).value);
28     }
29 }
```

RUNSAVERESET

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:

JavaPython3JSC++JS

MW

Explore

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

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)

```
1
2 class ListNode {
3     int value = 0;
4     ListNode next;
5
6     ListNode(int value) {
7         this.value = value;
8     }
9 }
10
11 class MiddleOfLinkedList {
12
13     public static ListNode findMiddle(ListNode head) {
14         ListNode slow = head;
15         ListNode fast = head;
16         while (fast != null && fast.next != null) {
17             slow = slow.next;
18             fast = fast.next.next;
19         }
20
21         return slow;
22     }
23
24     public static void main(String[] args) {
25         ListNode head = new ListNode(1);
26         head.next = new ListNode(2);
27         head.next.next = new ListNode(3);
28         head.next.next.next = new ListNode(4);
```

RUN

SAVE

RESET

Close

Output

2.438s

Middle Node: 3
Middle Node: 4
Middle Node: 4

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

Happy Number (medium)

✓ MARK AS COMPLETED

Next →

Problem Challenge 1

Report an Issue