

## Grokking the Coding Interview: Patterns for Coding Questions

14% completed

Challenge 1

Problem Challenge 2

Solution Review: Problem Challenge 2

Problem Challenge 3

Solution Review: Problem Challenge 3

### Pattern: Fast & Slow pointers

Introduction

LinkedList Cycle (easy)

Start of LinkedList Cycle (medium)

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

## Start of LinkedList Cycle (medium)

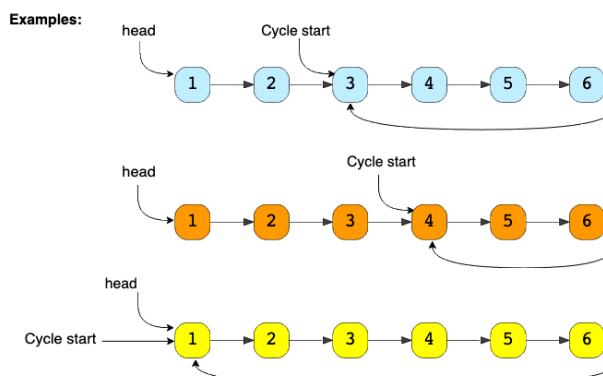
We'll cover the following

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

### Problem Statement

Given the head of a **Singly LinkedList** that contains a cycle, write a function to find the **starting node of the cycle**.

Examples:



### Try it yourself

Try solving this question here:

JavaPython3JS C++

```
1 constructor(value, next=null){
2   this.value = value;
3   this.next = next;
4 }
5
6
7
8 const find_cycle_start = function(head){
9   // TODO: Write your code here
10  return head;
11 };
12
13
14 head = new Node(1)
15 head.next = new Node(2)
16 head.next.next = new Node(3)
17 head.next.next.next = new Node(4)
18 head.next.next.next.next = new Node(5)
19 head.next.next.next.next.next = new Node(6)
20
21 head.next.next.next.next.next.next = head.next.next
22 console.log('LinkedList cycle start: ${find_cycle_start(head).value}')
23
24 head.next.next.next.next.next.next = head.next.next.next
25 console.log('LinkedList cycle start: ${find_cycle_start(head).value}')
26
27 head.next.next.next.next.next.next = head
28 console.log('LinkedList cycle start: ${find_cycle_start(head).value}')
29
```

RUNSAVERESET

### Solution

If we know the length of the **LinkedList** cycle, we can find the start of the cycle through the following steps:

1. Take two pointers. Let's call them **pointer1** and **pointer2**.
2. Initialize both pointers to point to the start of the LinkedList.
3. We can find the length of the LinkedList cycle using the approach discussed in [LinkedList Cycle](#). Let's assume that the length of the cycle is 'K' nodes.
4. Move **pointer2** ahead by 'K' nodes.

## 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)
- 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

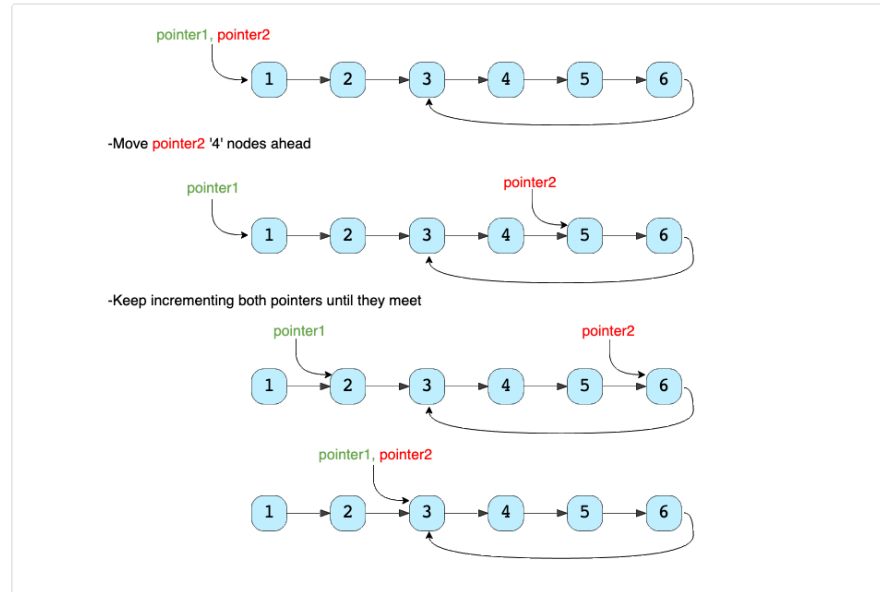
## Pattern: Subsets

- Introduction
- Subsets (easy)
- Subsets With Duplicates (easy)
- Permutations (medium)
- String Permutations by changing case (medium)
- Balanced Parentheses (hard)
- Unique Generalized

5. Now, keep incrementing `pointer1` and `pointer2` until they both meet.

6. As `pointer2` is 'K' nodes ahead of `pointer1`, which means, `pointer2` must have completed one loop in the cycle when both pointers meet. Their meeting point will be the start of the cycle.

Let's visually see this with the above-mentioned Example-1:



We can use the algorithm discussed in [LinkedList Cycle](#) to find the length of the cycle and then follow the above-mentioned steps to find the start of the cycle.

Code

Here is what our algorithm will look like:

JavaPython3C++JS

```
1 class Node {
2   constructor(value, next = null) {
3     this.value = value;
4     this.next = next;
5   }
6 }
7
8 function find_cycle_start(head) {
9   cycle_length = 0;
10  // find the LinkedList cycle
11  let slow = head;
12  fast = head;
13  while ((fast !== null && fast.next !== null)) {
14    fast = fast.next.next;
15    slow = slow.next;
16    if (slow === fast) { // found the cycle
17      cycle_length = calculate_cycle_length(slow);
18      break;
19    }
20  }
21  return find_start(head, cycle_length);
22 }
23
24
25 function calculate_cycle_length(slow) {
26   let current = slow;
27   cycle_length = 0;
28   while (true) {
```

RUNSAVERESET

Close

Output2.306s

LinkedList cycle start: 3  
LinkedList cycle start: 4  
LinkedList cycle start: 1

Time Complexity

As we know, finding the cycle in a LinkedList with 'N' nodes and also finding the length of the cycle requires  $O(N)$ . Also, as we saw in the above algorithm, we will need  $O(N)$  to find the start of the cycle. Therefore, the overall time complexity of our algorithm will be  $O(N)$ .

Space Complexity

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

+

Create

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

← Back

LinkedList Cycle (easy)

✓ MARK AS COMPLETED

Next →

Happy Number (medium)

🚩 Report an Issue