

## Grokking the Coding Interview: Patterns for Coding Questions

28% completed

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)

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

## Reverse a Sub-list (medium)

We'll cover the following

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

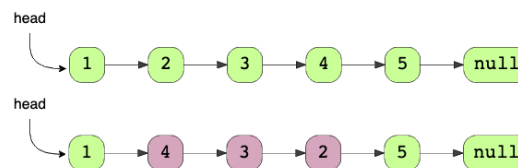
### Problem Statement

Given the head of a LinkedList and two positions 'p' and 'q', reverse the LinkedList from position 'p' to 'q'.

Example:

Original List:

p=2, q=4



### Try it yourself

Try solving this question here:

JavaPython3JS C++

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
class ListNode {
    int value;
    ListNode next;
}

class ReverseSubList {
    public static ListNode reverse(ListNode head, int p, int q) {
        // TODO: Write your code here
        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);

        ListNode result = ReverseSubList.reverse(head, 2, 4);
        System.out.print("Nodes of the reversed LinkedList are: ");
        while (result != null) {
            System.out.print(result.value + " ");
            result = result.next;
        }
    }
}
```

RUNSAVERESET

### Solution

The problem follows the **In-place Reversal of a LinkedList** pattern. We can use a similar approach as discussed in [Reverse a LinkedList](#). Here are the steps we need to follow:

1. Skip the first  $p-1$  nodes, to reach the node at position  $p$ .
2. Remember the node at position  $p-1$  to be used later to connect with the reversed sub-list.
3. Next, reverse the nodes from  $p$  to  $q$  using the same approach discussed in [Reverse a LinkedList](#).
4. Connect the  $p-1$  and  $q+1$  nodes to the reversed sub-list.

Code

Here is what our algorithm will look like:

```
1
2
3
4
import java.util.*;

class ListNode {
    int value = 0;
```

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 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 Binary Search

Introduction

Order-agnostic Binary Search (easy)

Ceiling of a Number (medium)

Next Letter (medium)

Number Range (medium)

Search in a Sorted Infinite Array (medium)

Minimum Difference Element (medium)

Bitonic Array Maximum (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: Bitwise XOR

Introduction

Single Number (easy)

Two Single Numbers (medium)

Complement of Base 10 Number (medium)

Problem Challenge 1

Solution Review: Problem Challenge 1

Pattern: Top 'K' Elements

Introduction

Top 'K' Numbers (easy)

Kth Smallest Number (easy)

'K' Closest Points to the Origin (easy)

Connect Ropes (easy)

MW

Explore

Tracks

My Courses

Edpresso

Refer a Friend

Create

```
5  ListNode next;
6
7  ListNode(int value) {
8      this.value = value;
9  }
10 }
11
12 class ReverseSubList {
13
14     public static ListNode reverse(ListNode head, int p, int q) {
15         if (p == q)
16             return head;
17
18         // after skipping 'p-1' nodes, current will point to 'p'th node
19         ListNode current = head, previous = null;
20         for (int i = 0; current != null && i < p - 1; ++i) {
21             previous = current;
22             current = current.next;
23         }
24
25         // we are interested in three parts of the LinkedList, part before index 'p', part between 'p' and
26         // 'q', and the part after index 'q'
27         ListNode lastNodeOfFirstPart = previous; // points to the node at index 'p-1'
28         // after reversing the LinkedList 'current' will become the last node of the sub-list
```

RUN

SAVE

RESET

Close

Output

2.400s

Nodes of the reversed LinkedList are: 1 4 3 2 5

Time complexity #

The time complexity of our algorithm will be  $O(N)$  where 'N' is the total number of nodes in the LinkedList.

Space complexity #

We only used constant space, therefore, the space complexity of our algorithm is  $O(1)$ .

Similar Questions #

Problem 1: Reverse the first 'k' elements of a given LinkedList.

Solution:

This problem can be easily converted to our parent problem; to reverse the first 'k' nodes of the list, we need to pass `p=1` and `q=k`.

Problem 2: Given a LinkedList with 'n' nodes, reverse it based on its size in the following way:

1. If 'n' is even, reverse the list in a group of n/2 nodes.

2. If n is odd, keep the middle node as it is, reverse the first 'n/2' nodes and reverse the last 'n/2' nodes.

Solution:

When 'n' is even we can perform the following steps:

1. Reverse first 'n/2' nodes: `head = reverse(head, 1, n/2)`

2. Reverse last 'n/2' nodes: `head = reverse(head, n/2 + 1, n)`

When 'n' is odd, our algorithm will look like:

1. `head = reverse(head, 1, n/2)`

2. `head = reverse(head, n/2 + 2, n)`

Please note the function call in the second step. We're skipping two elements as we will be skipping the middle element.

Interviewing soon? We've partnered with Hired so that companies apply to you, instead of the other way around. [See how](#)

MARK AS COMPLETED

Back

Reverse a LinkedList (easy)

Next

Reverse every K-element Sub-list (me...

Report an Issue