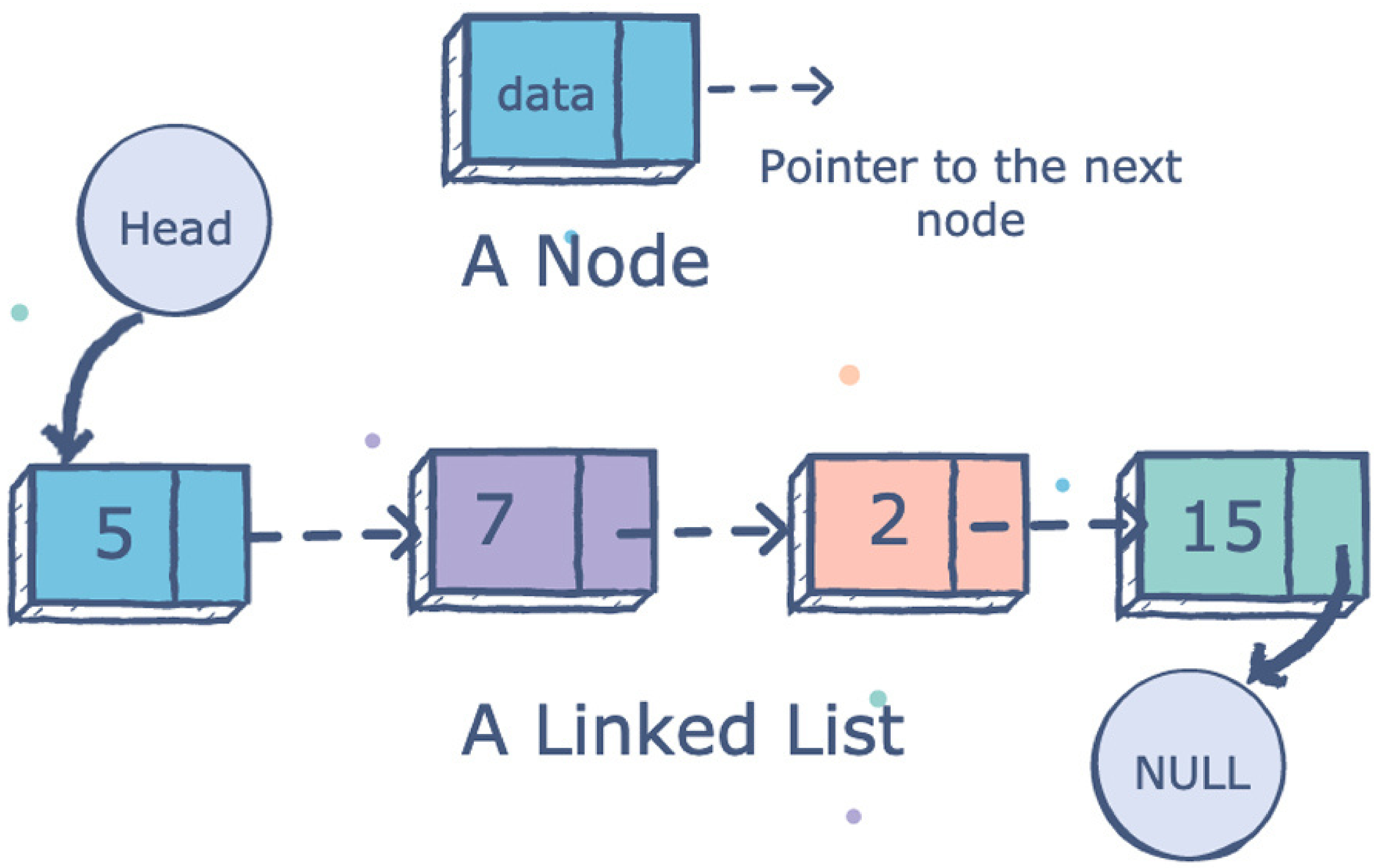
AlgoTutor

MASTER LINKED LIST IN JUST 05 DAYS

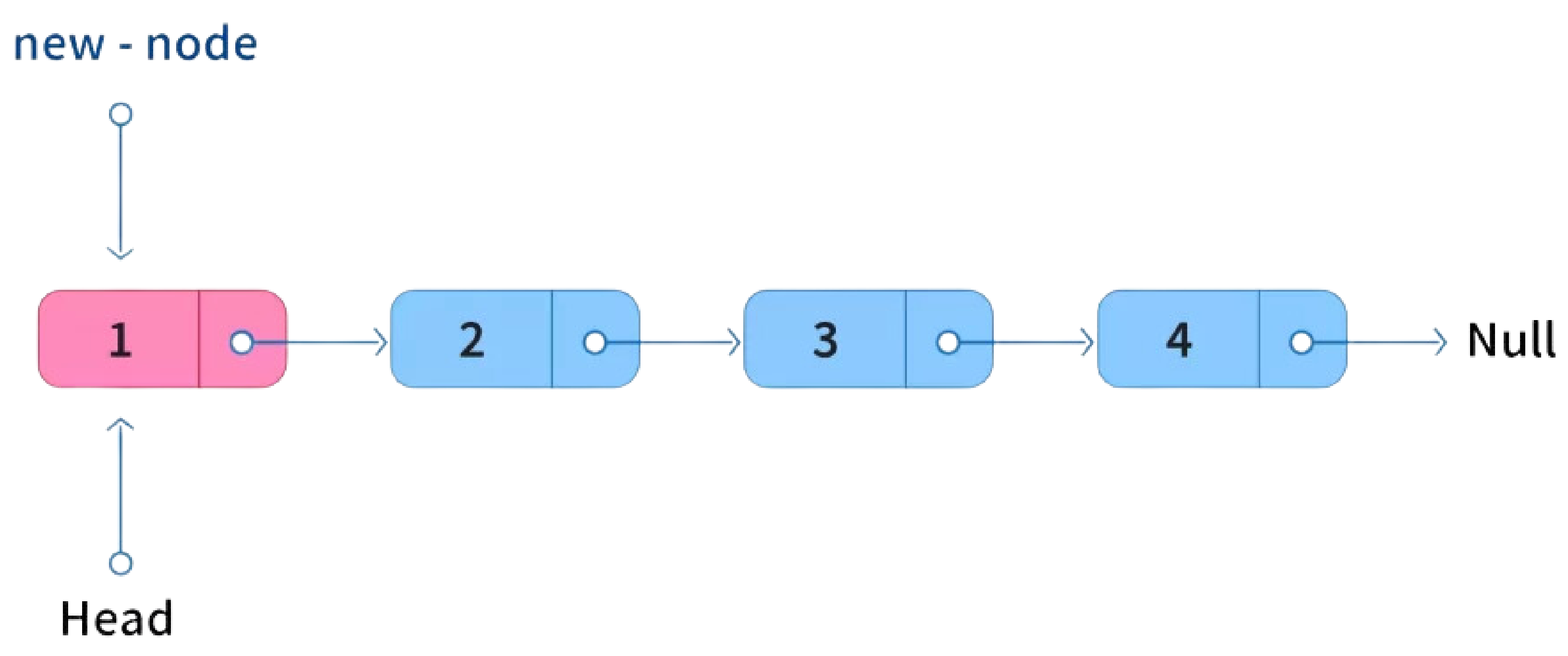


AlgoTutor

Day 1

Introduction to Linked Lists

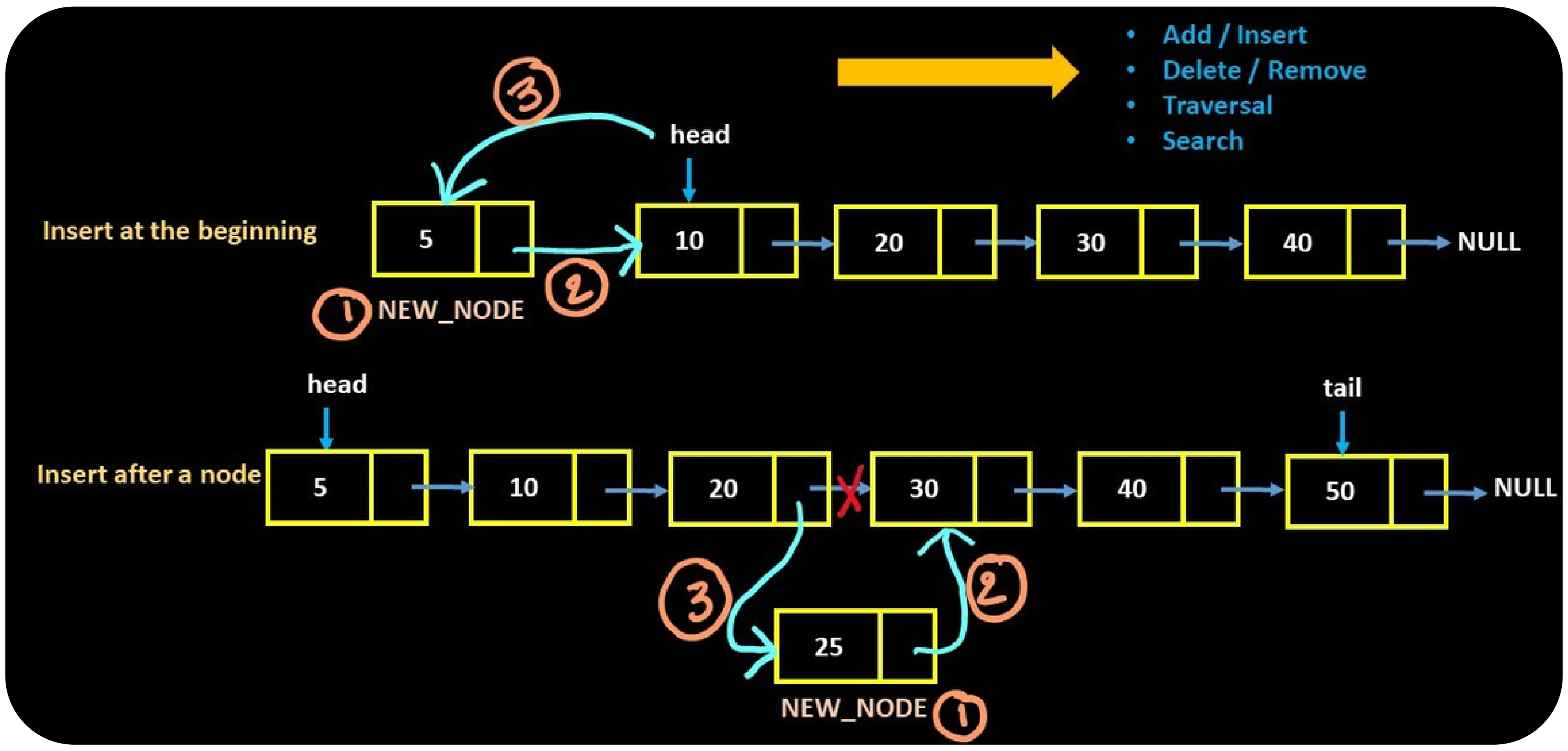
- Learn the basics of linked lists, including singly linked lists and doubly linked lists.
- Understand the structure of nodes and how to create a linked list.



Day 2

Linked List Operations

- Study common linked list operations such as insertion, deletion, and traversal.
- Implement these operations on a linked list.



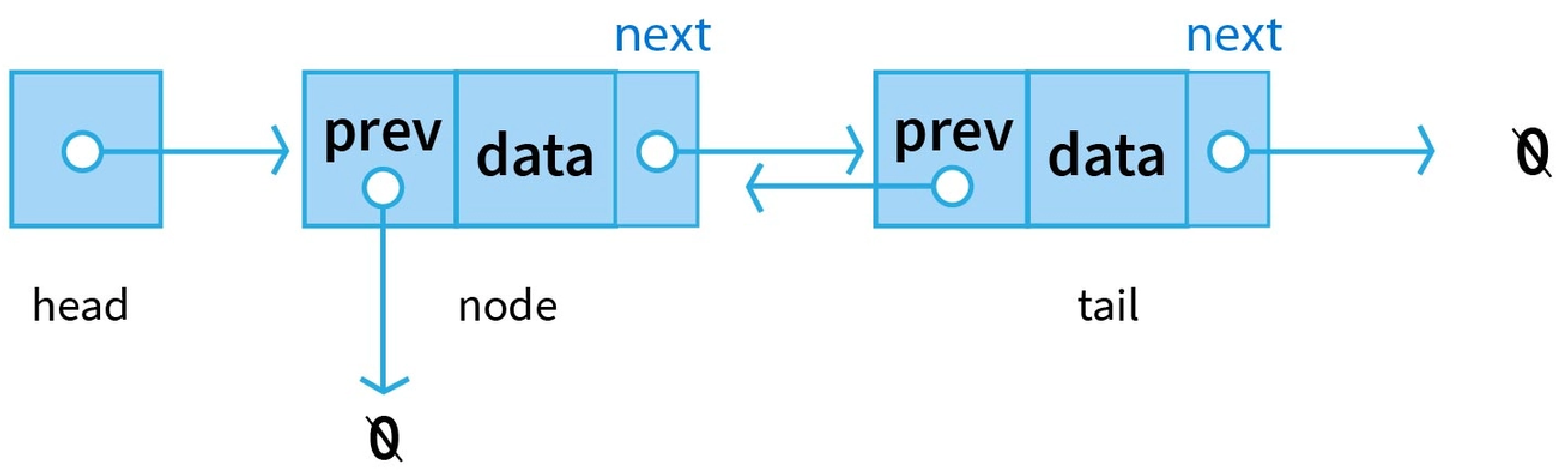
AlgoTutor

Day 3

Advanced Linked List Topics

🔶 Explore more advanced linked list concepts, including circular linked lists and skip lists.

🔶 Study the advantages and disadvantages of various types of linked lists.



Day 4

Common Linked List Problems

🔷 Solve problems that are commonly asked in coding interviews.

🔷 Focus on problems involving reversing a linked list, finding the middle element, and detecting cycles.

AlgoTutor

Day 5

Advanced Problems and Review

- Challenge yourself with advanced linked list problems.
- Review the concepts and problem-solving techniques you've learned.

AlgoTutor

Important Practice Questions

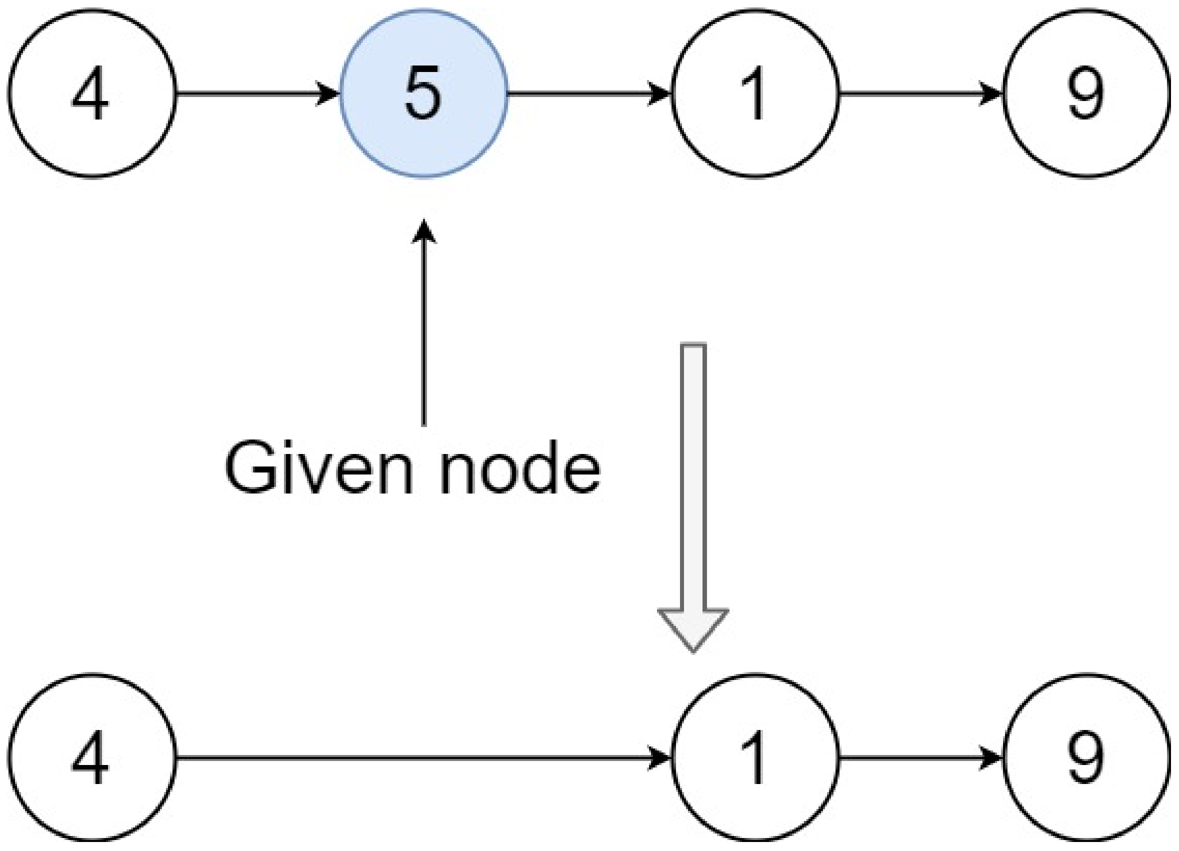
01. Delete Node in a Linked List

There is a singly-linked list head and we want to delete a node node in it.

You are given the node to be deleted node. You will not be given access to the first node of head.

All the values of the linked list are unique, and it is guaranteed that the given node node is not the last node in the linked list.

Example 1:



Input: head = [4,5,1,9], node = 5

Output: [4,1,9]

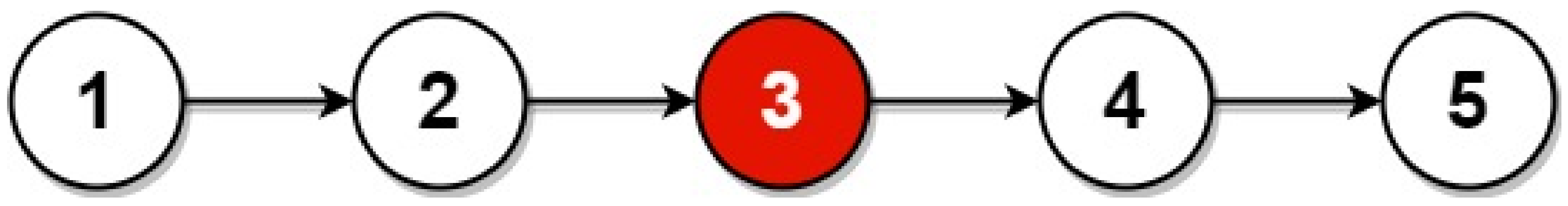
Practice

02. Middle of the Linked List

Given the head of a singly linked list, return the middle node of the linked list.

If there are two middle nodes, return the second middle node.

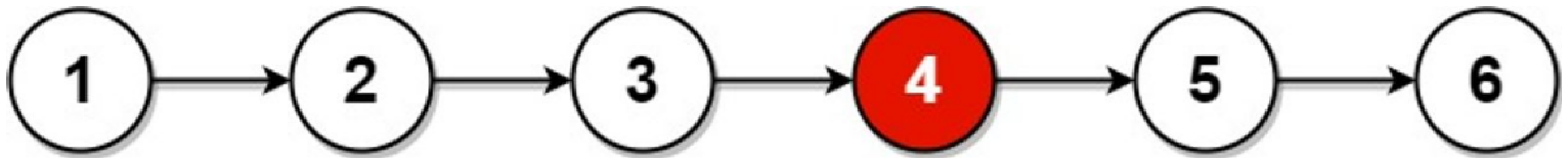
Example 1:



Input: head = [1,2,3,4,5]

Output: [3,4,5]

Example 2:



Input: head = [1,2,3,4,5,6]

Output: [4,5,6]

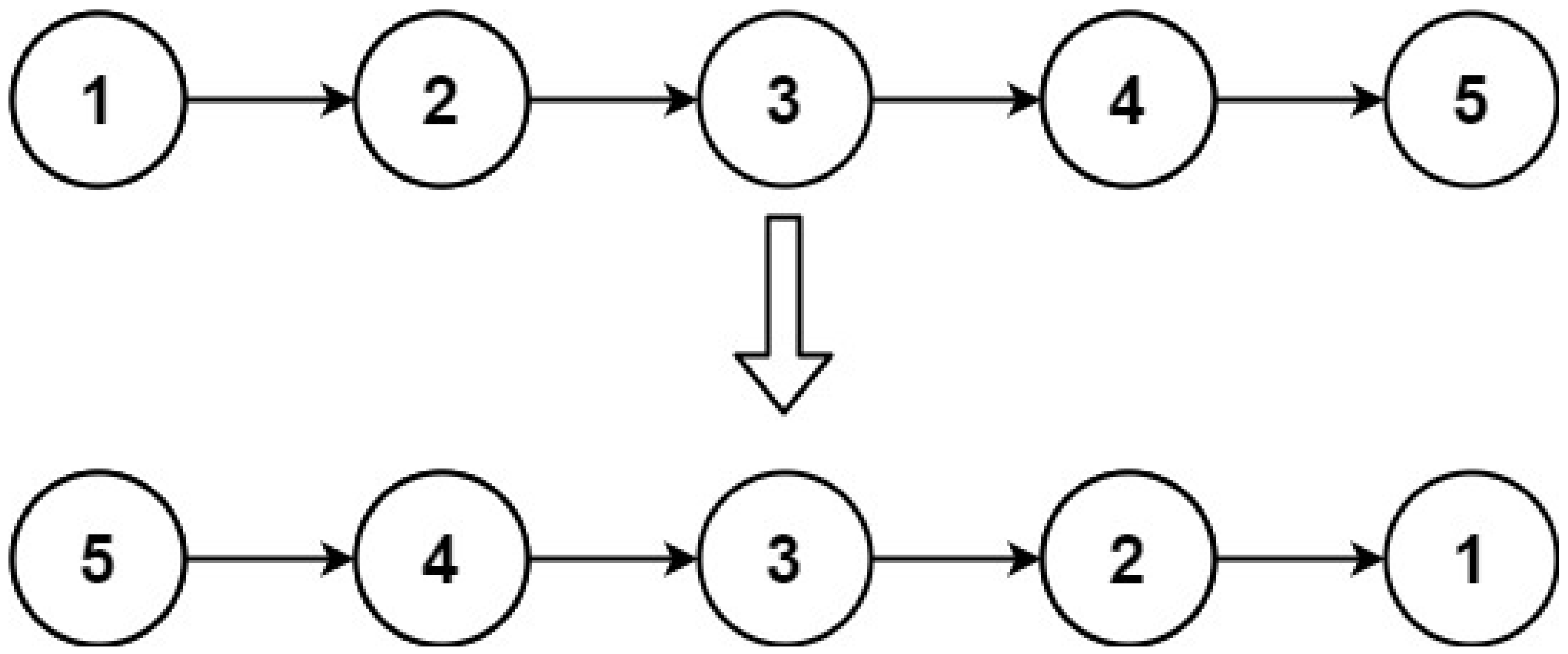
Practice

AlgoTutor

03. Reverse Linked List

Given the head of a singly linked list, reverse the list, and return the reversed list.

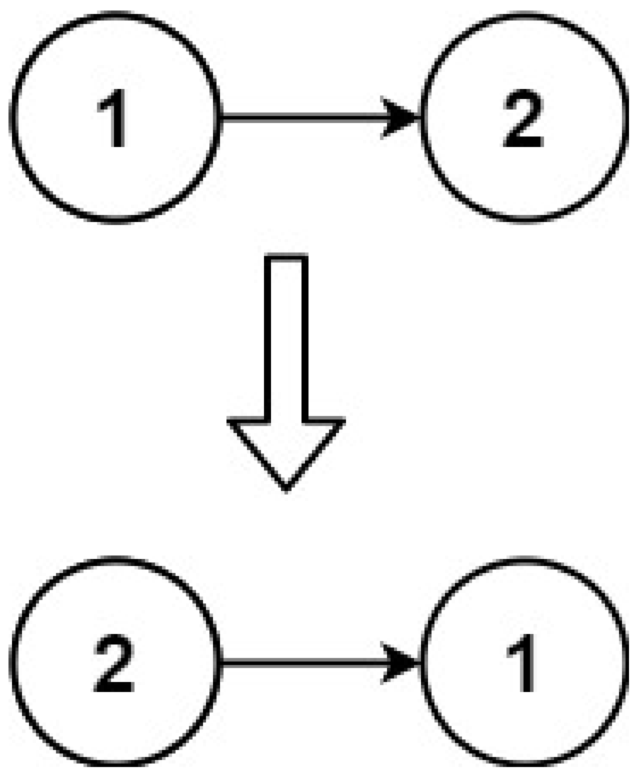
Example 1:



Input: head = [1,2,3,4,5]

Output: [5,4,3,2,1]

Example 2:



Input: head = [1,2]

Output: [2,1]

Practice

AlgoTutor

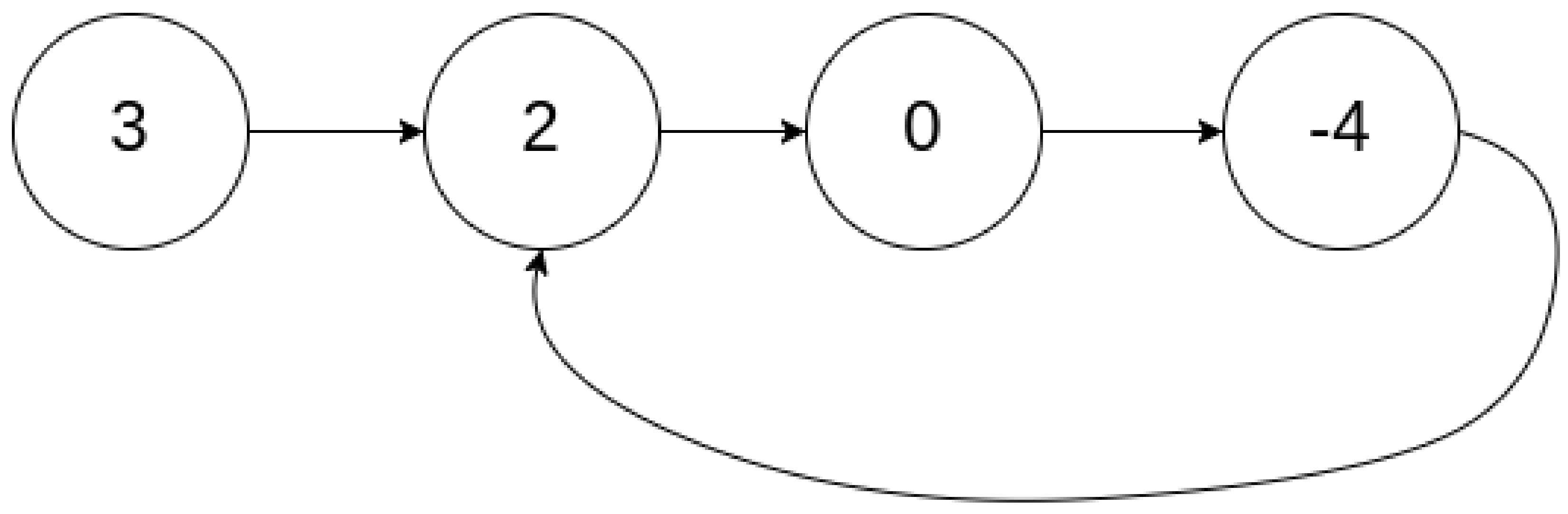
04. Linked List Cycle

Given head, the head of a linked list, determine if the linked list has a cycle in it.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to. Note that pos is not passed as a parameter.

Return true if there is a cycle in the linked list. Otherwise, return false.

Example 1:



Input: head = [3,2,0,-4], pos = 1

Output: true

Practice

AlgoTutor

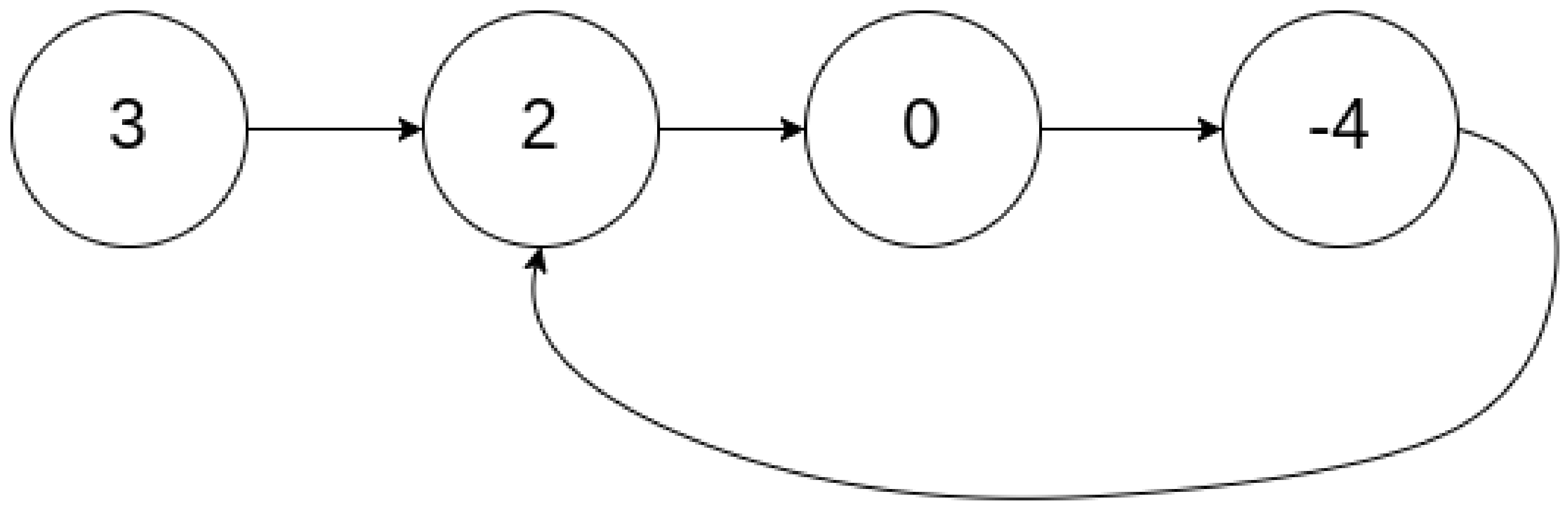
05. Linked List Cycle II

Given the head of a linked list, return the node where the cycle begins. If there is no cycle, return null.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to (0-indexed). It is -1 if there is no cycle. Note that pos is not passed as a parameter.

Do not modify the linked list.

Example 1:



Input: head = [3,2,0,-4], pos = 1

Output: tail connects to node index 1

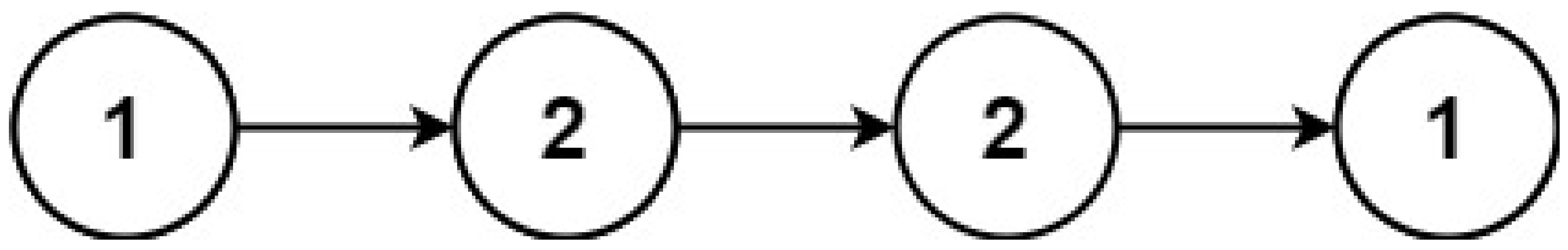
Practice

AlgoTutor

06. Palindrome Linked List

Given the head of a singly linked list, return true if it is a palindrome or false otherwise.

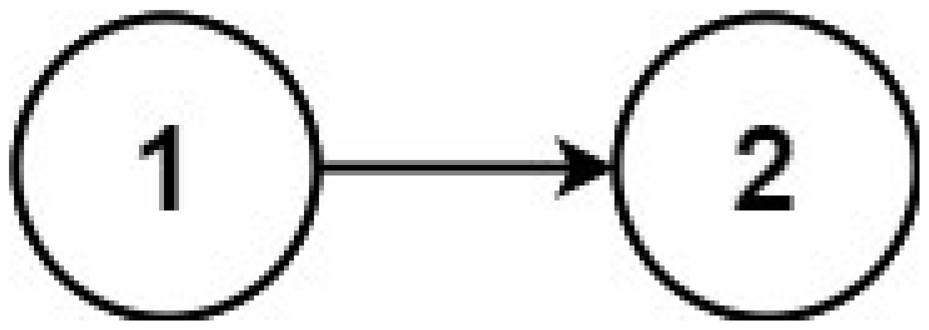
Example 1:



Input: head = [1,2,2,1]

Output: true

Example 1:



Input: head = [1,2]

Output: false

Practice

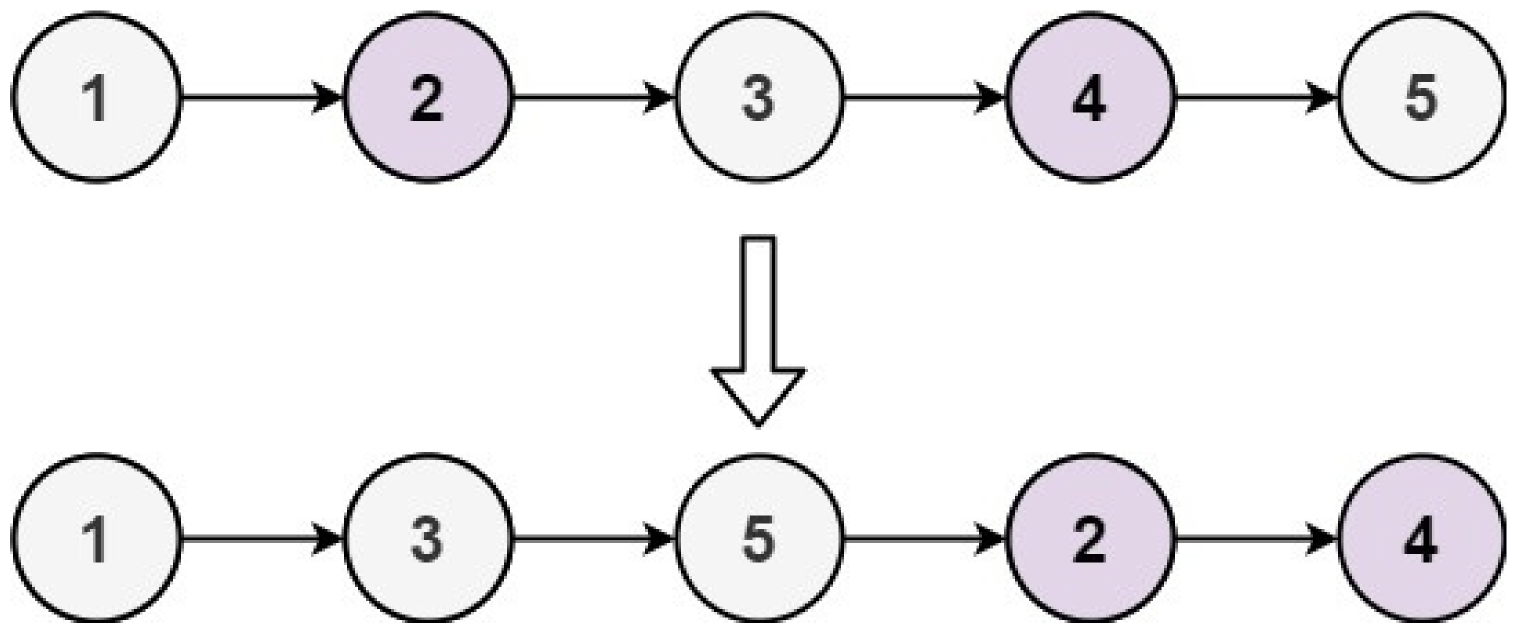
07. Odd Even Linked List

Given the head of a singly linked list, group all the nodes with odd indices together followed by the nodes with even indices, and return the reordered list.

The first node is considered odd, and the second node is even, and so on.

Note that the relative order inside both the even and odd groups should remain as it was in the input.
You must solve the problem in O(1) extra space complexity and O(n) time complexity.

Example 1:



Input: head = [1,2,3,4,5]

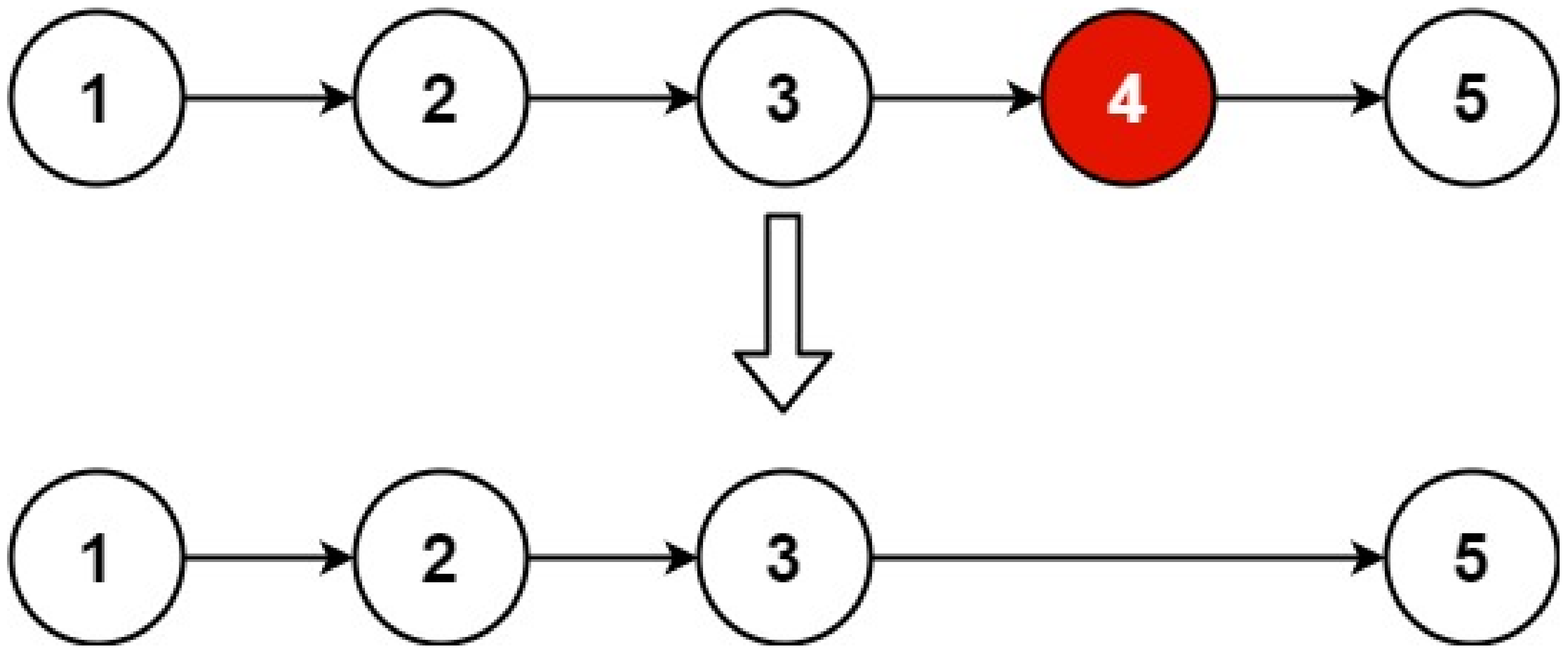
Output: [1,3,5,2,4]

Practice

08. Remove Nth Node From End of List

Given the head of a linked list, remove the nth node from the end of the list and return its head.

Example 1:



Input: head = [1,2,3,4,5], n = 2

Output: [1,2,3,5]

Example 2:

Input: head = [1], n = 1

Output: []

Practice

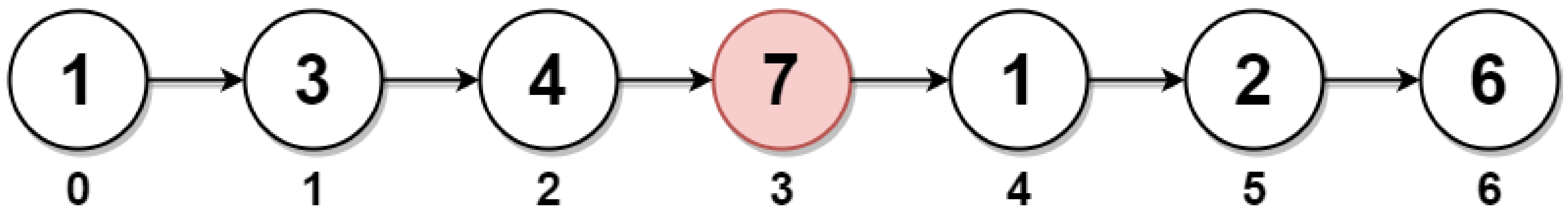
09. Delete the Middle Node of a Linked List

You are given the head of a linked list. Delete the middle node, and return the head of the modified linked list.

The middle node of a linked list of size n is the ⌊n / 2⌋th node from the start using 0-based indexing, where ⌊x⌋ denotes the largest integer less than or equal to x.

- For n = 1, 2, 3, 4, and 5, the middle nodes are 0, 1, 1, 2, and 2, respectively.

Example 1:



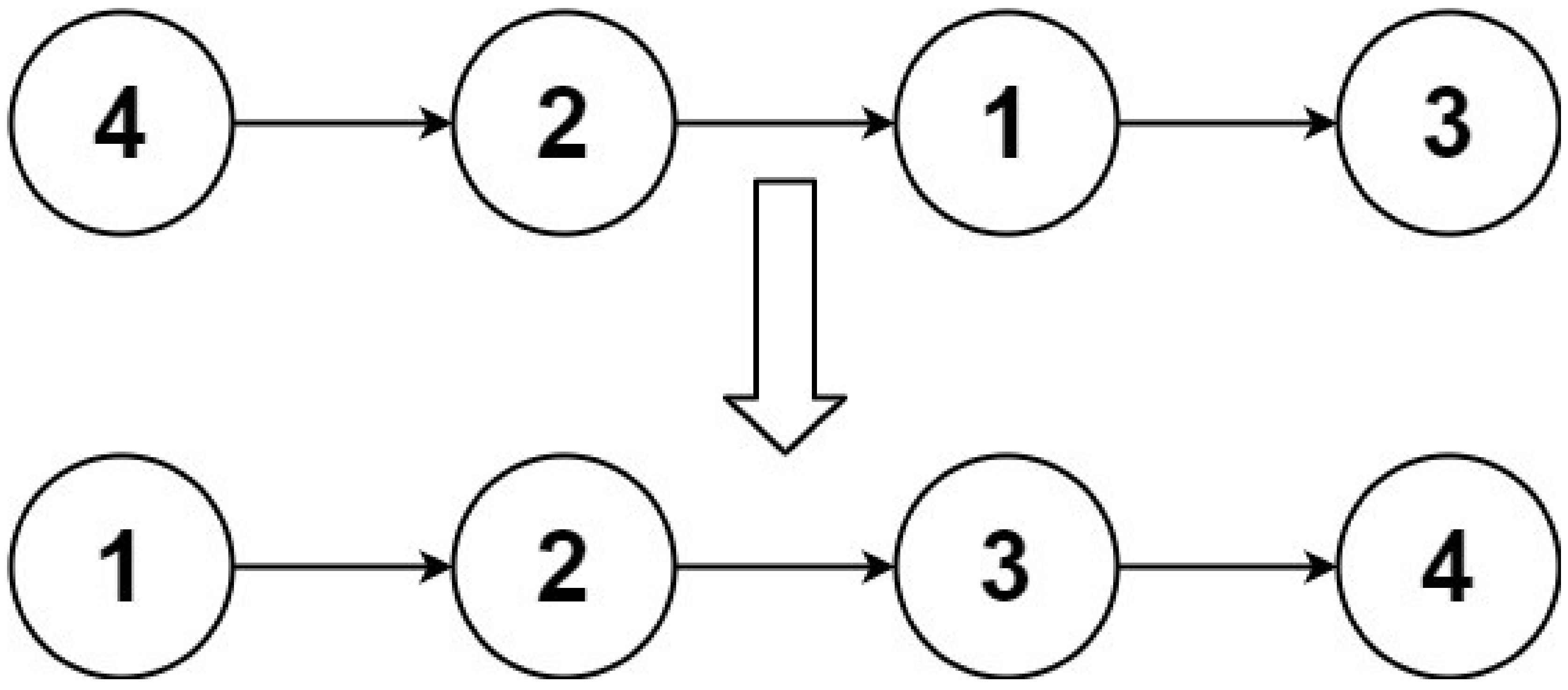
Input: head = [1,3,4,7,1,2,6]
Output: [1,3,4,1,2,6]

Practice

10. Sort List

Given the head of a linked list, return the list after sorting it in ascending order.

Example 1:



Input: head = [4,2,1,3]

Output: [1,2,3,4]

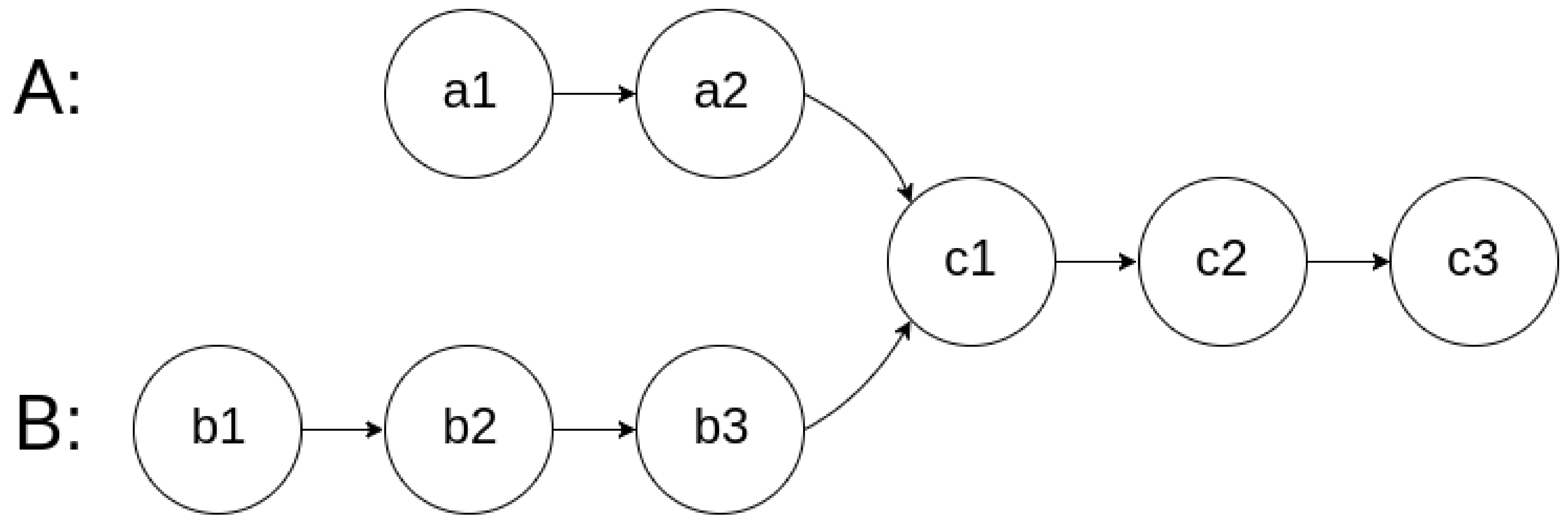
Practice

11. Intersection of Two Linked Lists

Given the heads of two singly linked-lists headA and headB, return the node at which the two lists intersect. If the two linked lists have no intersection at all, return null.

For example, the following two linked lists begin to intersect at node c1:

Example 1:



Input: intersectVal = 8, listA = [4,1,8,4,5], listB = [5,6,1,8,4,5], skipA = 2, skipB = 3
Output: Intersected at '8'

Practice

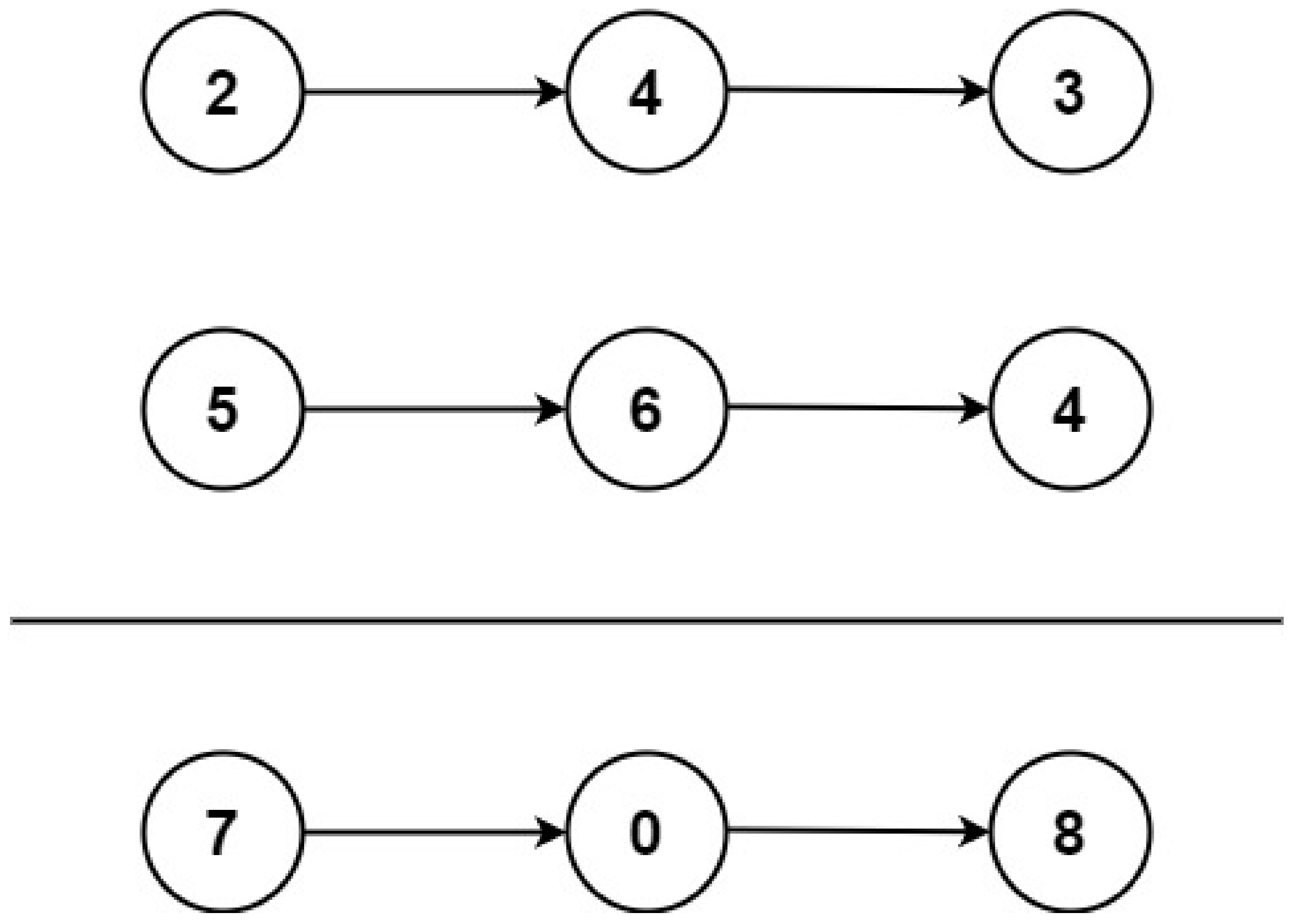
AlgoTutor

12. Add Two Numbers

You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Example 1:



Input: l1 = [2,4,3], l2 = [5,6,4]
Output: [7,0,8]

Practice

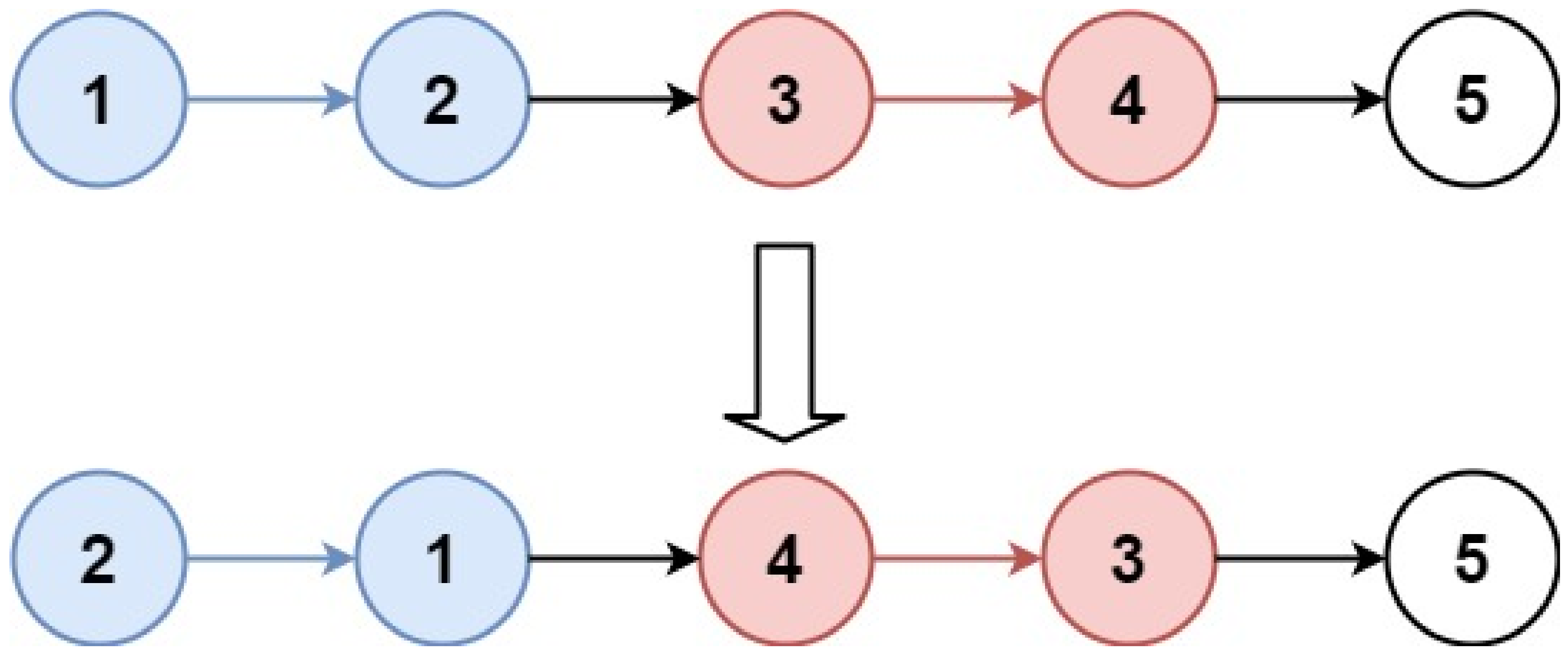
13. Reverse Nodes in k-Group

Given the head of a linked list, reverse the nodes of the list k at a time, and return the modified list.

k is a positive integer and is less than or equal to the length of the linked list. If the number of nodes is not a multiple of k then left-out nodes, in the end, should remain as it is.

You may not alter the values in the list's nodes, only nodes themselves may be changed.

Example 1:



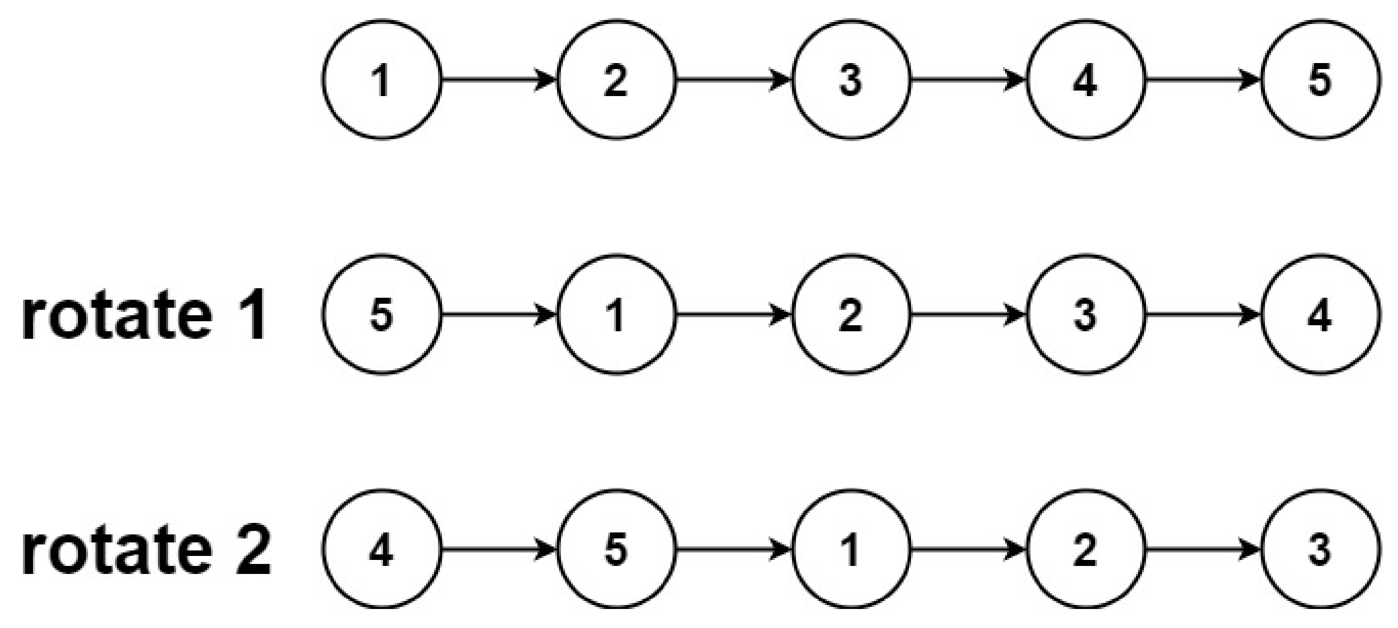
Input: head = [1,2,3,4,5], k = 2
Output: [2,1,4,3,5]

Practice

14. Rotate List

Given the head of a linked list, rotate the list to the right by k places.

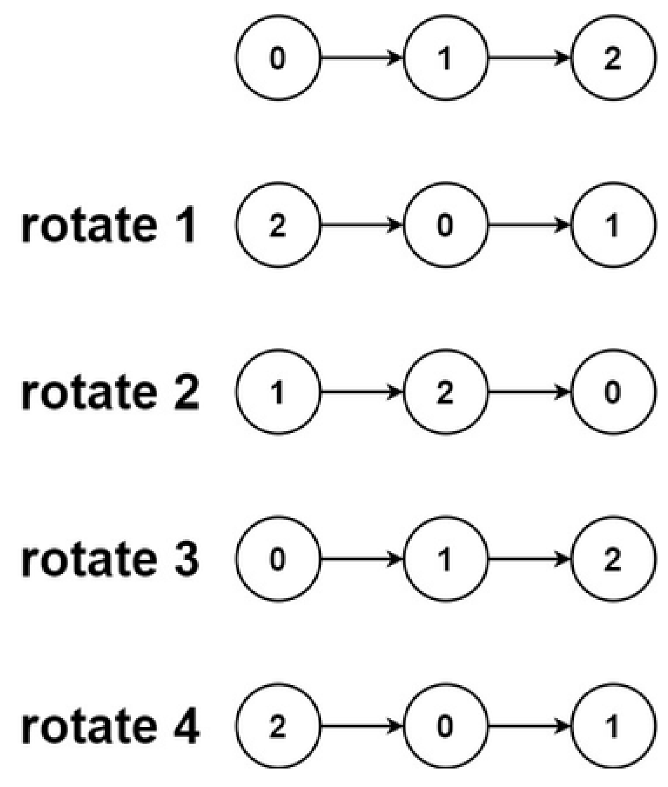
Example 1:



Input: head = [1,2,3,4,5], k = 2

Output: [4,5,1,2,3]

Example 1:



Input: head = [0,1,2], k = 4

Output: [2,0,1]

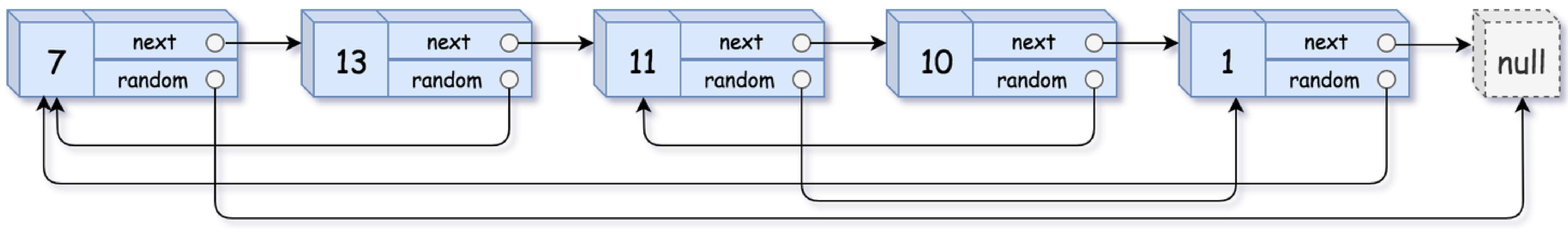
Practice

15. Copy List with Random Pointer

A linked list of length n is given such that each node contains an additional random pointer, which could point to any node in the list, or null.

Construct a deep copy of the list. The deep copy should consist of exactly n brand new nodes, where each new node has its value set to the value of its corresponding original node. Both the next and random pointer of the new nodes should point to new nodes in the copied list such that the pointers in the original list and copied list represent the same list state. None of the pointers in the new list should point to nodes in the original list.

Example 1:



Input: head = [[7,null],[13,0],[11,4],[10,2],[1,0]]

Output: [[7,null],[13,0],[11,4],[10,2],[1,0]]

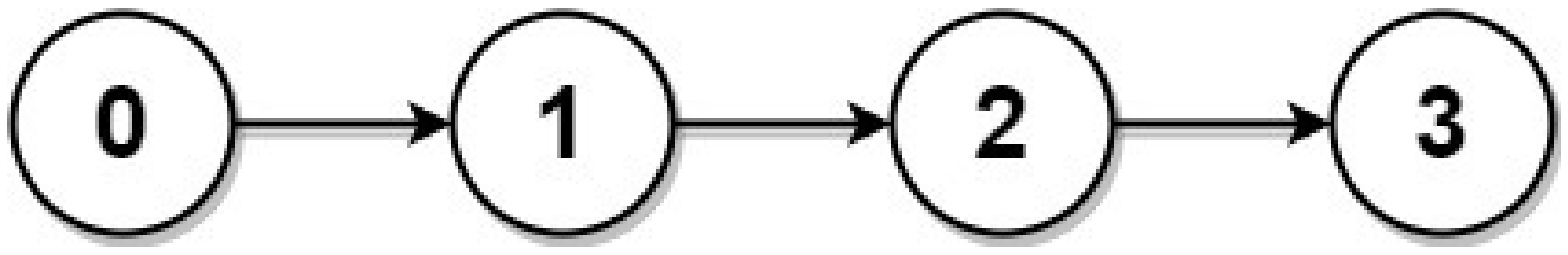
Practice

16. Linked List Components

You are given the head of a linked list containing unique integer values and an integer array nums that is a subset of the linked list values.

Return the number of connected components in nums where two values are connected if they appear consecutively in the linked list.

Example 1:



Input: head = [0,1,2,3], nums = [0,1,3]
Output: 2

Practice

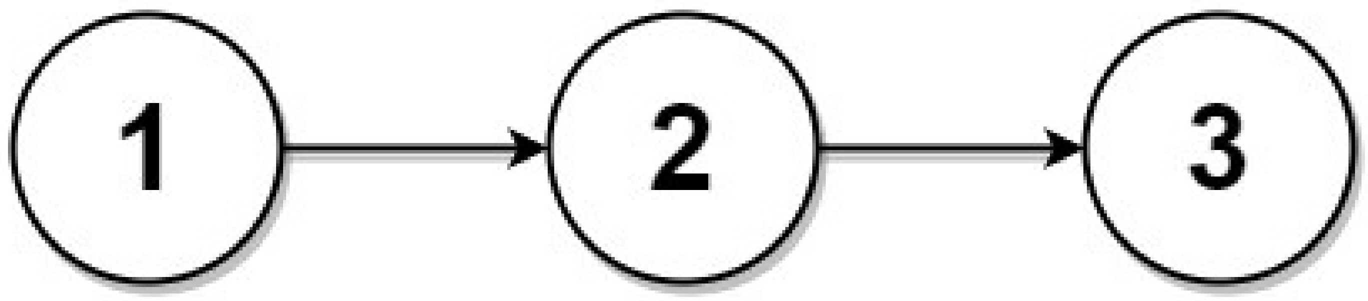
17. Split Linked List in Parts

Given the head of a singly linked list and an integer k, split the linked list into k consecutive linked list parts.

The length of each part should be as equal as possible: no two parts should have a size differing by more than one. This may lead to some parts being null.

The parts should be in the order of occurrence in the input list, and parts occurring earlier should always have a size greater than or equal to parts occurring later.

Example 1:



Input: head = [1,2,3], k = 5
Output: [[1],[2],[3],[],[]]

Practice

18. Remove Zero Sum Consecutive Nodes from Linked List

Given the head of a linked list, we repeatedly delete consecutive sequences of nodes that sum to 0 until there are no such sequences.

After doing so, return the head of the final linked list. You may return any such answer.

(Note that in the examples below, all sequences are serializations of ListNode objects.)

Example 1:
Input: head = [1,2,-3,3,1]
Output: [3,1]

Example 2:
Input: head = [1,2,3,-3,4]
Output: [1,2,4]

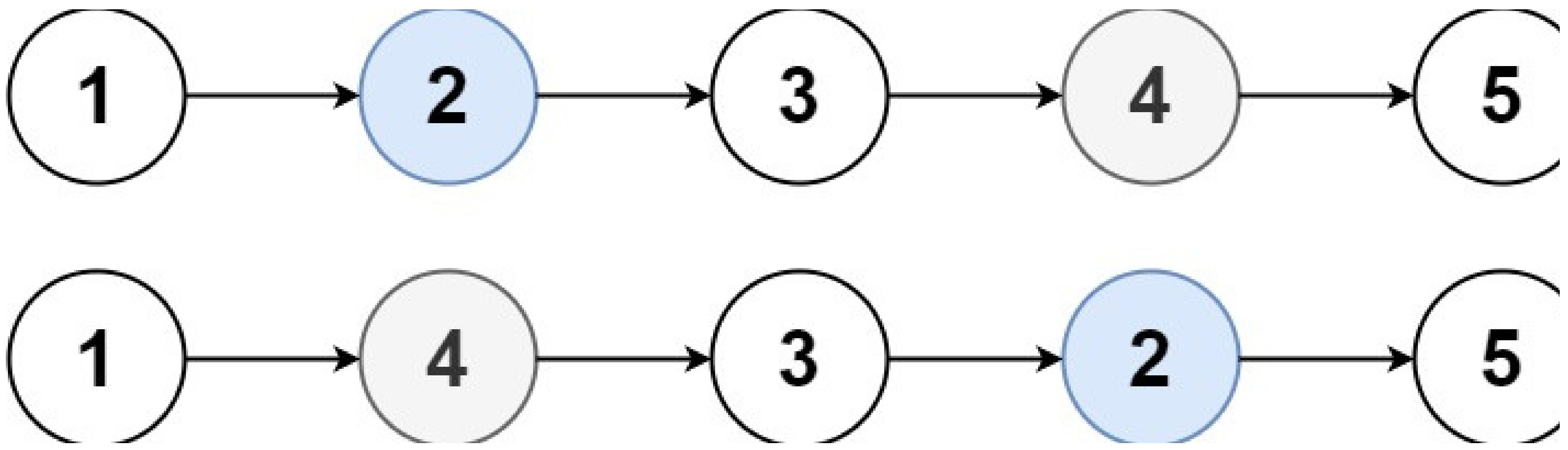
Practice

19. Swapping Nodes in a Linked List

You are given the head of a linked list, and an integer k.

Return the head of the linked list after swapping the values of the kth node from the beginning and the kth node from the end (the list is 1-indexed).

Example 1:



Input: head = [1,2,3,4,5], k = 2
Output: [1,4,3,2,5]

Practice

20. Design Browser History

You have a browser of one tab where you start on the homepage and you can visit another url, get back in the history number of steps or move forward in the history number of steps.

Example:

Input:

["BrowserHistory","visit","visit","visit","back","back","forward","visit","forward","back","back"]
[["leetcode.com"],["google.com"],["facebook.com"],["youtube.com"],[1],[1],[1],["linkedin.com"],[2],[2],[7]]

Output:

[null,null,null,null,"facebook.com","google.com","facebook.com",null,"linkedin.com","google.com","leetcode.com"]

Practice