Swap Nodes pairwise in a linked list.

Mr. Vikas Kumar
Assistant Professor
Industry Embedded Program

Lesson Plan

Subject/Course	**Competitive Coding**
Lesson Title	**Swap Nodes pairwise in a linked list.**

Lesson Objectives
Rearrange the linked list so every two adjacent nodes are swapped (1↔2, 3↔4, ...).
Do it in O(n) time and O(1) extra space — by changing links (ptrs), not node values.

Problem Statement:

Write a program to Swap Nodes pairwise in a linked list.

Concept

A linked list is a linear data structure composed of a sequence of nodes, where each node contains data and a pointer (or reference) to the next node in the sequence.

Common operations:

- Insertion: Adding a new node to the linked list
- Deletion: Removing a node from the linked list.
- Traversal
- Searching

Algorithm/Logic

1. Create a singly linked list.
2. Traverse the list using a pointer current.

- While current and current.next exist:
- Swap the data of current and current.next.

3. Move current two nodes ahead.
4. Print the list after swapping

Visualization

Diagram (ASCII)

Before:

```rust
head -> 1 -> 2 -> 3 -> 4 -> 5 -> null
```

Swap 1 & 2:

```rust
dummy->2 -> 1 -> 3 -> 4 -> 5
```

Swap 3 & 4:

```rust
dummy->2->1->4->3->5
```

Code Implementation

```
1  public class Practical9_SwapPairs {
2
3      static class ListNode {
4          int data;
5          ListNode next;
6          ListNode(int val) { data = val; }
7      }
8
9      static ListNode swapPairs(ListNode head) {
10         if (head == null || head.next == null) return head;
11         ListNode first = head;
12         ListNode second = head.next;
13
14         first.next = swapPairs(second.next);
15         second.next = first;
16
17         return second;
18     }
```

```
19      public static void main(String[] args) {
20          ListNode head = new ListNode(1);
21          head.next = new ListNode(2);
22          head.next.next = new ListNode(3);
23          head.next.next.next = new ListNode(4);
24
25          System.out.print("Original List: ");
26          printList(head);
27
28          ListNode swapped = swapPairs(head);
29          System.out.print("After Swapping Pairs: ");
30          printList(swapped);
31      }
32
33      static void printList(ListNode node) {
34          while (node != null) {
35              System.out.print(node.data + " ");
36              node = node.next;
37          }
38          System.out.println();
39      }
40  }
```

Output :

```
Original List: 1 2 3 4
After Swapping Pairs: 2 1 4 3
```

Output

Input Linked List: 1 -> 2 -> 3 -> 4

Output Linked List: 2 -> 1 -> 4 -> 3

Time & Space Complexity

Time Complexity: O(N)

The algorithm iterates through the linked list once, processing each pair of nodes. In each iteration of the while loop, a constant number of operations (pointer assignments) are performed. Therefore, the time taken grows linearly with the number of nodes (N) in the linked list.

Space Complexity: O(1)

The algorithm uses a constant amount of extra space, regardless of the input linked list's size. A dummy node and a prev pointer are created, which consume a fixed amount of memory.

No auxiliary data structures are used that scale with the input size.

Summary

- This algorithm efficiently swaps adjacent nodes in a singly linked list.

Practice Exercises

Problem Statement 1

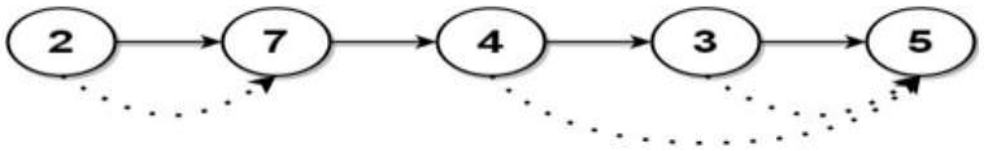
You are given the `head` of a linked list with `n` nodes.

For each node in the list, find the value of the **next greater node**. That is, for each node, find the value of the first node that is next to it and has a **strictly larger** value than it.

Return an integer array `answer` where `answer[i]` is the value of the next greater node of the i^{th} node (**1-indexed**). If the i^{th} node does not have a next greater node, set `answer[i] = 0`.

```
Input: head = [2,1,5]
Output: [5,5,0]
```

Example 2:




```
Input: head = [2,7,4,3,5]
Output: [7,0,5,5,0]
```

Constraints:

- The number of nodes in the list is `n`.
- `1 <= n <= 10^4`
- `1 <= Node.val <= 10^9`

Problem Statement Path:

Competitive Coding

Linked List

Next Greater Node in Linked List

Link:

https://leetcode.com/problems/next-greater-node-in-linked-list/description/

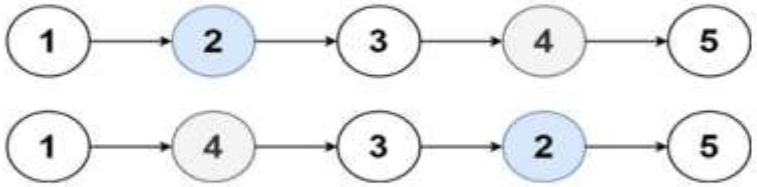
Practice Exercises

Problem Statement 2

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* k^{th} *node from the beginning and the* k^{th} *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]
```

Example 2:

```
Input: head = [7,9,6,6,7,8,3,0,9,5], k = 5
Output: [7,9,6,6,8,7,3,0,9,5]
```

Constraints:

- The number of nodes in the list is `n`.
- `1 <= k <= n <= 10^5`
- `0 <= Node.val <= 100`

Problem Statement Path:

Competitive Coding

Linked List

Swapping Nodes in a Linked List

Link:

https://leetcode.com/problems/swapping-nodes-in-a-linked-list/description/

Practice Exercises

Problem Statement 3

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]
Note: The answer [1,2,1] would also be accepted.
```

Example 2:

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

Constraints:
The given linked list will contain between 1 and 1000 nodes.
Each node in the linked list has -1000 <= node.val <= 1000.

Problem Statement Path:

Competitive Coding

Linked List

Remove-zero-sum-consecutive-nodes-from-linked-list

Link :

https://leetcode.com/problems/remove-zero-sum-consecutive-nodes-from-linked-list/description/

Thanks