OpenAI

Reverse Linked List K Group

Write java code for the 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.
Input: head = [1,2,3,4,5], k = 2
Output: [2,1,4,3,5]

Here's a Java implementation to reverse the nodes of a linked list in groups of `k`:

```java
// Definition for singly-linked list.
class ListNode {
    int val;
    ListNode next;
    ListNode(int val) {
        this.val = val;
        this.next = null;
    }
}

public class Solution {
    public ListNode reverseKGroup(ListNode head, int k) {
        if (head == null || k == 1) return head;

        // Dummy node to handle edge cases like when head needs to be reversed
        ListNode dummy = new ListNode(0);
        dummy.next = head;

        // Initialize pointers
        ListNode prevGroupEnd = dummy;
        ListNode current = head;
        ListNode next = null;
        ListNode groupStart = null;

        // Count the number of nodes in the linked list
        int length = 0;
        while (current != null) {
            length++;
            current = current.next;
        }

        // Reset current to the head of the list
        current = head;

        // Loop over the linked list to reverse nodes in k-groups
        while (length >= k) {
            groupStart = current;
            ListNode prev = null;

            // Reverse k nodes
            for (int i = 0; i < k; i++) {
                next = current.next;
                current.next = prev;
                prev = current;
                current = next;
            }

            // Connect the reversed group to the previous part of the list
            prevGroupEnd.next = prev;
            groupStart.next = current;

            // Move prevGroupEnd to the end of the reversed group
            prevGroupEnd = groupStart;

            // Reduce the remaining length by k
            length -= k;
        }

        // Return the new head of the modified list
        return dummy.next;
    }

    // Helper method to print the linked list
    public static void printList(ListNode head) {
        while (head != null) {
            System.out.print(head.val + " ");
            head = head.next;
        }
        System.out.println();
    }

    public static void main(String[] args) {
        // Creating the linked list 1 -> 2 -> 3 -> 4 -> 5
        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);

        // Define the group size k
        int k = 2;

        // Reverse in k-group and print the new list
        Solution solution = new Solution();
        ListNode newHead = solution.reverseKGroup(head, k);

        // Print the modified list
        solution.printList(newHead);  // Expected output: 2 1 4 3 5
    }
}
```

Explanation:

- **ListNode class**: Defines the structure of a node in the linked list.
- **reverseKGroup method**:
  - The dummy node helps to deal with edge cases like reversing the head of the list.
  - The outer loop processes each group of size `k`.
  - For each group, it reverses the nodes using a standard reversing technique.
  - After reversing, it connects the reversed group with the rest of the list.
  - If there are fewer than `k` nodes left, the loop stops, leaving them as is.

Example:

For input `[1, 2, 3, 4, 5]` and `k = 2`, the output will be `[2, 1, 4, 3, 5]`.