25. Reverse Nodes in k-Group

Hard 3306 395 Add to List Share

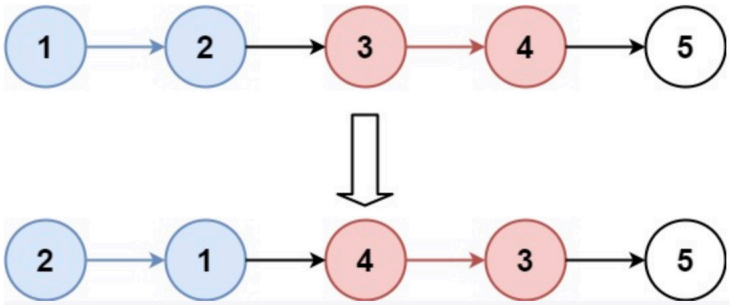
Given a linked list, reverse the nodes of a linked list *k* at a time and return its modified list.

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.

Follow up:

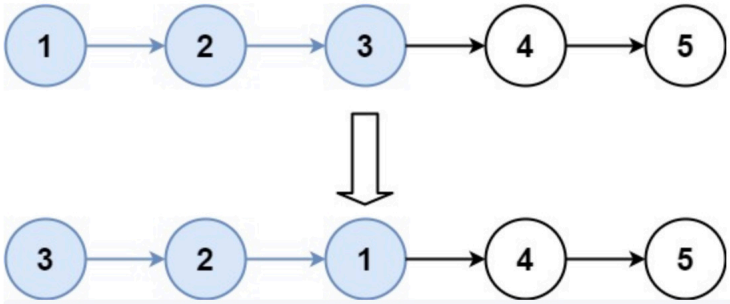
- Could you solve the problem in `O(1)` extra memory space?
- You may not alter the values in the list's nodes, only nodes itself may be changed.

Example 1:




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

Example 2:




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

Example 3:

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

Example 4:

```
Input: head = [1], k = 1
Output: [1]
```

head
1 → 2 → 3 → 4 → 5 → 6 → NULL

1. Reverse k elements started with head.

newHead head
NULL ← 1 ← 2 3 → 4 → 5 → 6 → NULL

2. Take the k+1 element as head to call the reverseKGroup recursion function.

newHead head
NULL ← 1 ← 2 3 → 4 → 5 → 6 → NULL

3. Connect the results.

newHead head
NULL ← 1 ← 2 3 → 4 → 5 → 6 → NULL
└→ reveseKGroup (head , 2)

```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* reverseKGroup(ListNode* head, int k) {
        if (head == NULL) {
            return head;
        }

        ListNode *a = head;
        ListNode *b = a;
        for (int i = 0; i < k; i++) {
            if (b == NULL) {
                return head;
            }
            b = b->next;
        }

        // reverse k elements
        ListNode *newHead = reverse(a, b);

        // reverse remaining elements recursively and connect results
        a->next = reverseKGroup(b, k);

        return newHead;
    }

private:
    ListNode* reverse(ListNode* a, ListNode* b) {
        ListNode *pre = NULL, *cur = a, *next;
        while(cur != b) {
            next = cur->next;
            cur->next = pre;
            pre = cur;
            cur = next;
        }
        return pre;
    }
};
```