25. Reverse Nodes in k-Group ★

Question | Editorial Solution

My Submissions (/problems/reverse-nodes-in-k-group/submissions/)

Total Accepted: **70050**   Total Submissions: **242638**   Difficulty: **Hard**

Notes

Given a linked list, reverse the nodes of a linked list *k* at a time and return its modified 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 nodes, only nodes itself may be changed.

Only constant memory is allowed.

For example,

Given this linked list: `1->2->3->4->5`

For *k* = 2, you should return: `2->1->4->3->5`

For *k* = 3, you should return: `3->2->1->4->5`

Subscribe (/subscribe/) to see which companies asked this question

Show Tags

Show Similar Problems

Have you met this question in a real interview? Yes No

Discuss (https://leetcode.com/discuss/questions/oj/reverse-nodes-in-k-group)

Pick One (/problems/random-one-question/)

C++

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

✉ Send Feedback (mailto:admin@leetcode.com?subject=Feedback)

```
        bool isGroup = true;
        do {
            groupBegin = groupEnd;
            for (int i = 0; i < k; ++i) {
                if (groupEnd)
                    groupEnd = groupEnd->next;
                else {
                    isGroup = false;
                    break;
                }
            }
            if (isGroup) {
                stack<ListNode*> Stack;
                for (auto it = groupBegin; it != groupEnd; it = it->next) {
                    Stack.push(it);
                }
                while (!Stack.empty()) {
                    nowPtr->next = Stack.top();
                    Stack.pop();
                    nowPtr = nowPtr->next;
                    nowPtr->next = NULL;
                }
            }
            else {
                nowPtr->next = groupBegin;
            }
        } while (groupEnd);

        return dummy.next;
    }
};
```

Notes

Custom Testcase ☐

Run Code   Submit Solution