JavaScript Reverse Nodes in k-Group

Challenge

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.

1st Example

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

2nd Example

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

Constraints

- `1 <= k <= n <= 5000`
- `0 <= Node.val <= 1000`
- The number of nodes in the list is `n`.

Solution

```
const reverseKGroup = (head, k) => {
    let count = 0,
        node  = head;

    while (node && count != k) {
        node = node.next;
        count++;
    }

    if (count == k) {
        node = reverseKGroup(node, k);

        while (count > 0) {
            let temp  = head.next;
            head.next = node;
            node      = head;
            head      = temp;
            count--;
        }

        head = node;
    }

    return head;
};
```

Explanation

I've defined a function called `reverseKGroup` that takes in two parameters: `head` and `k`. The purpose of this function is to reverse a linked list in groups of size `k`.

The function begins by initializing a variable called `count` to 0 and a variable called `node` to the value of `head`.

Next, a while loop is started that continues as long as `node` is not null and `count` is not equal to `k`. This loop is used to iterate through the linked list and count the number of nodes up to `k`.

Inside the loop, the `node` variable is updated to the next node in the linked list, and the `count` variable is incremented by 1.

After the while loop, there is an if statement that checks if `count` is equal to `k`. If it is, it means that a group of `k` nodes has been counted.

In this case, the `reverseKGroup` function is recursively called on the `node` with the same value of `k`. This recursive call is used to reverse the remaining portion of the linked list.

Following the recursive call, another while loop is started that continues as long as `count` is greater than 0. This loop is used to reverse the current group of nodes.

Inside the loop, a temporary variable called `temp` is created and assigned the value of `head.next`. Then, the `head.next` is set to the value of `node`, effectively reversing the link of the current node.

The `node` variable is updated to the value of `head`, and the `head`

variable is updated to the value of `temp`. This process is repeated `count` times to reverse the entire group.

After the while loop, the value of `node` is assigned to `head`, effectively updating the head of the reversed linked list.

Finally, the function returns the value of `head`, which represents the head of the reversed linked list.

In summary, the `reverseKGroup` function reverses a linked list in groups of size `k` by recursively reversing the remaining portion of the list and then reversing the current group of nodes. The function returns the head of the reversed linked list.

Author: Trevor Morin