Previous Next

61. Rotate List

Feb. 20, 2019 | 20.2K views

Average Rating: 4.72 (32 votes)

Given a linked list, rotate the list to the right by *k* places, where *k* is non-negative.

Example 1:

```
Input: 1->2->3->4->5->NULL, k = 2
Output: 4->5->1->2->3->NULL
Explanation:
rotate 1 steps to the right: 5->1->2->3->4->NULL
rotate 2 steps to the right: 4->5->1->2->3->NULL
```

Example 2:

```
Input: 0->1->2->NULL, k = 4
Output: 2->0->1->NULL
Explanation:
rotate 1 steps to the right: 2->0->1->NULL
rotate 2 steps to the right: 1->2->0->NULL
rotate 3 steps to the right: 0->1->2->NULL
rotate 4 steps to the right: 2->0->1->NULL
```

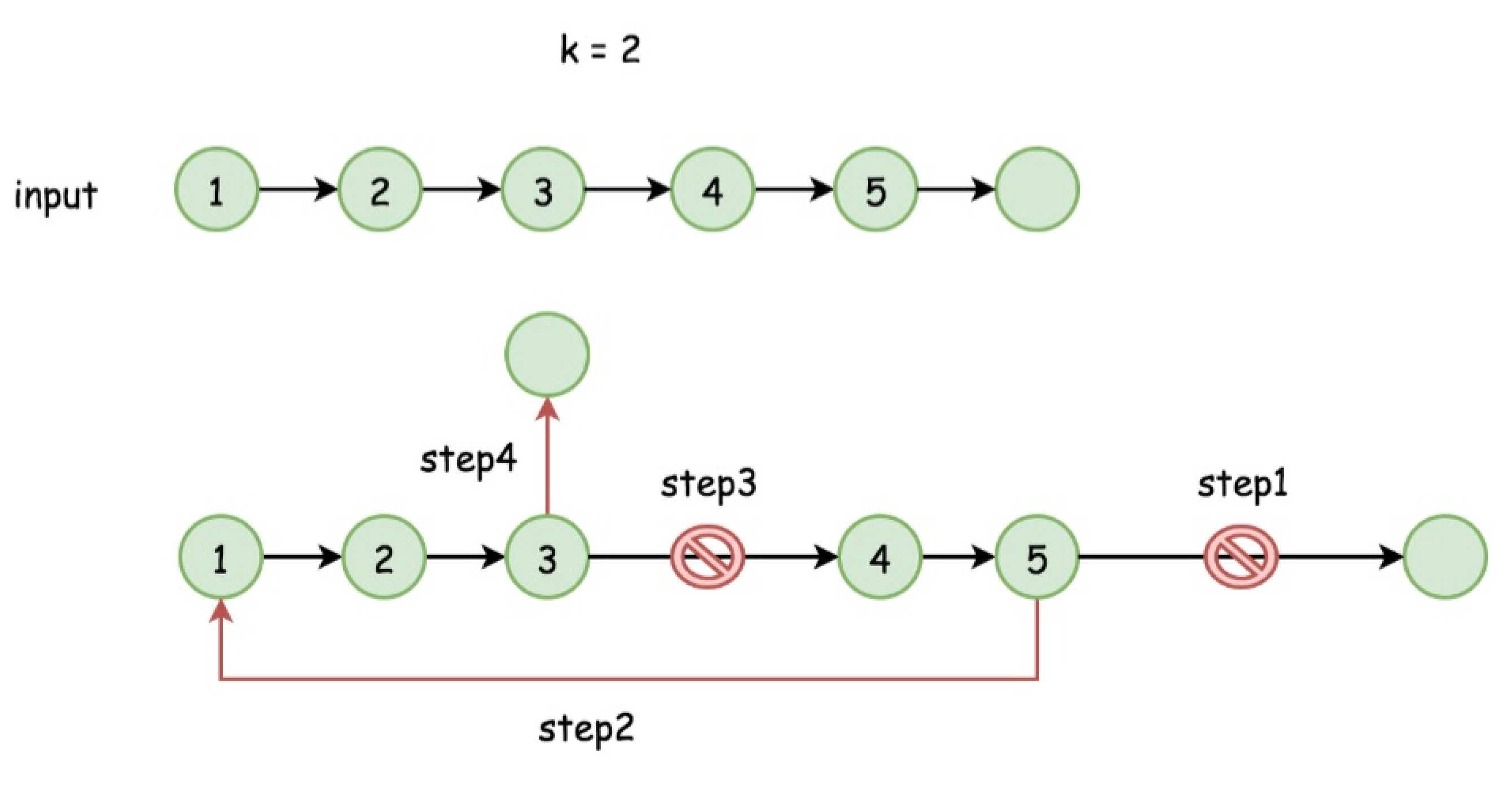
Solution

Approach 1:

Intuition

The nodes in the list are already linked, and hence the rotation basically means

- To close the linked list into the ring.
- To break the ring after the new tail and just in front of the new head.



> Where is the new head?

In the position `n - k`, where `n` is a number of nodes in the list. The new tail is just before, in the position `n - k - 1`.

> We were assuming that `k < n`. What about the case of `k >= n`?

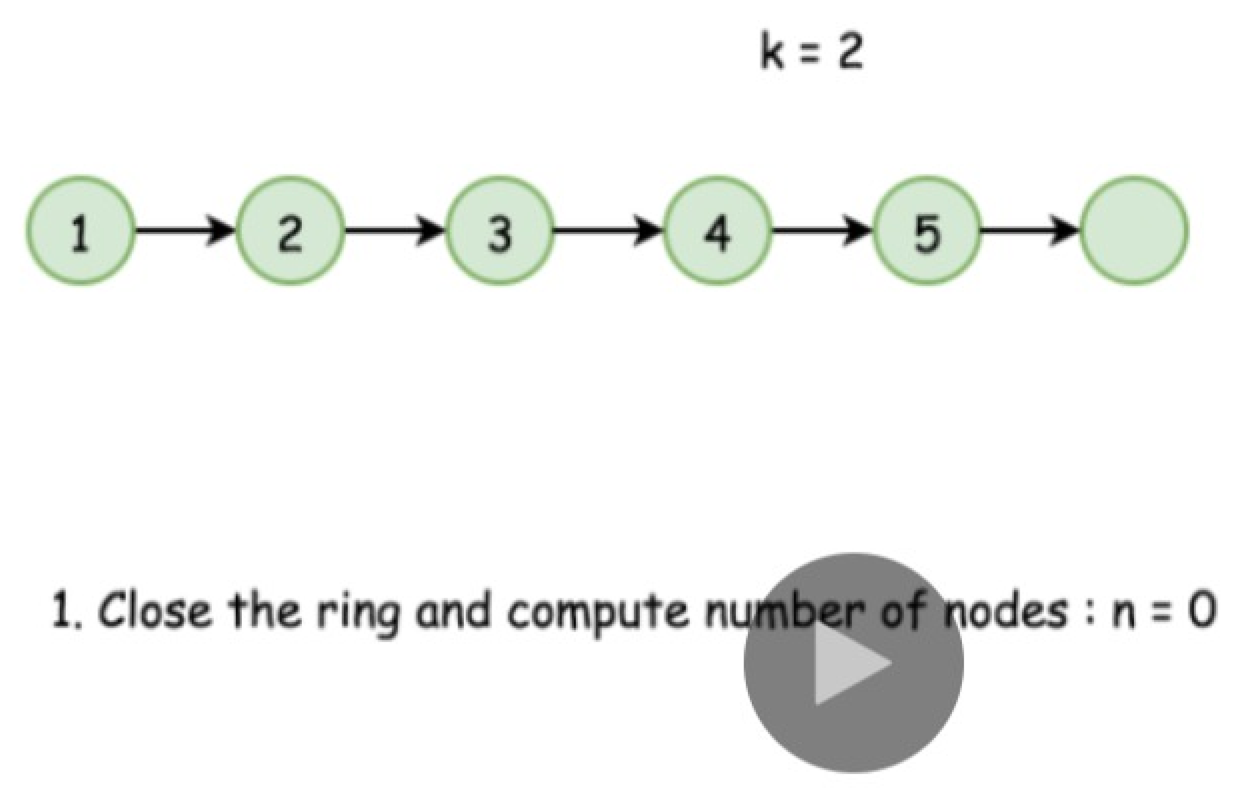
`k` could be rewritten as a sum `k = (k // n) * n + k % n`, where the first term doesn't result in any rotation. Hence one could simply replace `k` by `k % n` to always have number of rotation places smaller than `n`.

Algorithm

The algorithm is quite straightforward :

- Find the old tail and connect it with the head `old_tail.next = head` to close the ring. Compute the length of the list `n` at the same time.
- Find the new tail, which is `(n - k % n - 1)` th node from the `head` and the new head, which is `(n - k % n)` th node.
- Break the ring `new_tail.next = None` and return `new_head`.

Implementation



1 / 14

Java Python Copy

```python
class Solution:
    def rotateRight(self, head: 'ListNode', k: 'int') -> 'ListNode':
        # base cases
        if not head:
            return None
        if not head.next:
            return head

        # close the linked list into the ring
        old_tail = head
        n = 1
        while old_tail.next:
            old_tail = old_tail.next
            n += 1
        old_tail.next = head

        # find new tail : (n - k % n - 1)th node
        # and new head : (n - k % n)th node
        new_tail = head
        for i in range(n - k % n - 1):
            new_tail = new_tail.next
        new_head = new_tail.next

        # break the ring
        new_tail.next = None

        return new_head
```

Complexity Analysis

- Time complexity : $\mathcal{O}(N)$ where N is a number of elements in the list.
- Space complexity : $\mathcal{O}(1)$ since it's a constant space solution.

Rate this article:

Previous Next

Comments: 11

Sort By

Type comment here... (Markdown is supported)

Preview Post

yanshengjia ★61 October 8, 2019 5:34 AM

Why can't I come up with something like this?

22 Share Reply

SHOW 1 REPLY

DelandF ★14 February 21, 2019 7:38 AM

Amazing

14 Share Reply

jenildesai25 ★5 February 22, 2019 12:57 AM Report

```
def rotate_list(self, head, k):
    try:
        count = 0
        curr = head
```

Read More

3 Share Reply

super_saiyan_2 ★29 April 8, 2020 6:34 AM

Is anyone understanding the math behind
`new tail (n - k % n - 1)`
or
`new head : (n - k % n)`
???

Read More

1 Share Reply

SHOW 2 REPLIES

hans_bala ★1 January 17, 2020 10:23 AM Report

Java solution. Coded from scratch.

```
public ListNode rotateRight(ListNode head, int k) {
        // Compute the length of the linked list
```

Read More

1 Share Reply

yyfyifan ★48 November 8, 2019 5:18 AM

Excellent solution!

1 Share Reply

coriaedu ★2 July 1, 2020 11:13 PM Report

A slight optimization would be to check whether `k % size` equals zero before linking the tail to the head, as it should just return the same list.

0 Share Reply

junlzhan ★0 June 27, 2020 3:58 AM

A Java solution that does not create a ring for the list:

```
    public ListNode rotateRight(ListNode head, int k) {
        if(head == null || head.next == null)
            return head;
```

Read More

0 Share Reply

mitiphadnis ★0 May 21, 2020 2:43 AM Report

Question has similar logic as leetcode problem 19 "Remove nth node from end of list" with some extra conditions. (https://leetcode.com/problems/remove-nth-node-from-end-of-list/)
Here is my solution:

```
def rotateRight(self, head: ListNode, k: int) -> ListNode:
```

Read More

0 Share Reply

vidushi567 ★23 December 17, 2019 5:37 PM Report

I use similar logic as the provided solution but its slightly more naive and separated out. Time complexity is O(2N) ~ O(N)

```
   public class Solution {
       public ListNode RotateRight(ListNode head, int k) {
```

Read More

0 Share Reply

‹ 1 2 ›