# 题目

给定一个链表，旋转链表，将链表每个节点向右移动 k 个位置，其中 k 是非负数。

**示例 1:**

输入: 1->2->3->4->5->NULL, k = 2

输出: 4->5->1->2->3->NULL

解释:

向右旋转 1 步: 5->1->2->3->4->NULL

向右旋转 2 步: 4->5->1->2->3->NULL

**示例 2:**

输入: 0->1->2->NULL, k = 4

输出: 2->0->1->NULL

解释:

向右旋转 1 步: 2->0->1->NULL

向右旋转 2 步: 1->2->0->NULL

向右旋转 3 步: 0->1->2->NULL

向右旋转 4 步: 2->0->1->NULL

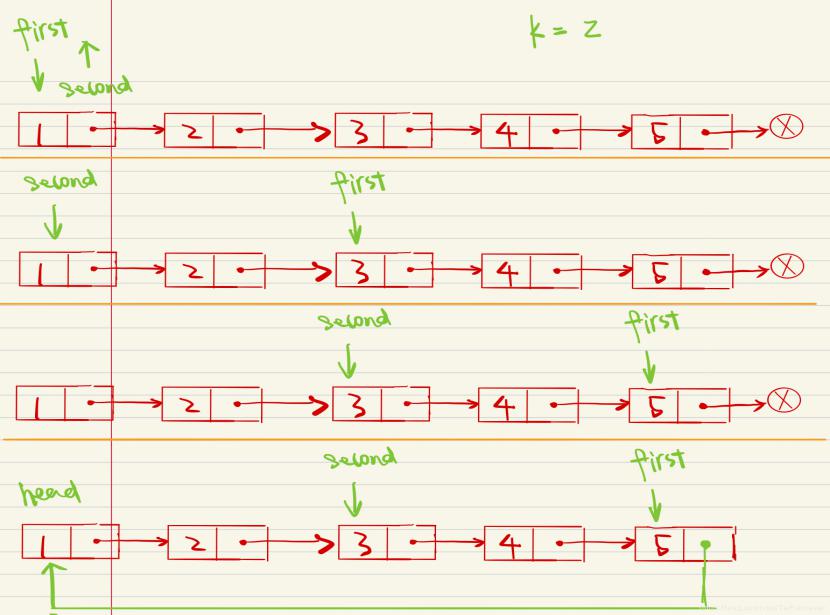
# 分析

## 方法一：双指针

思路：

这种有间隔的链表，大多数都是双指针。

先让快指针走k个位置，然后两个指针一起走完整个链表。

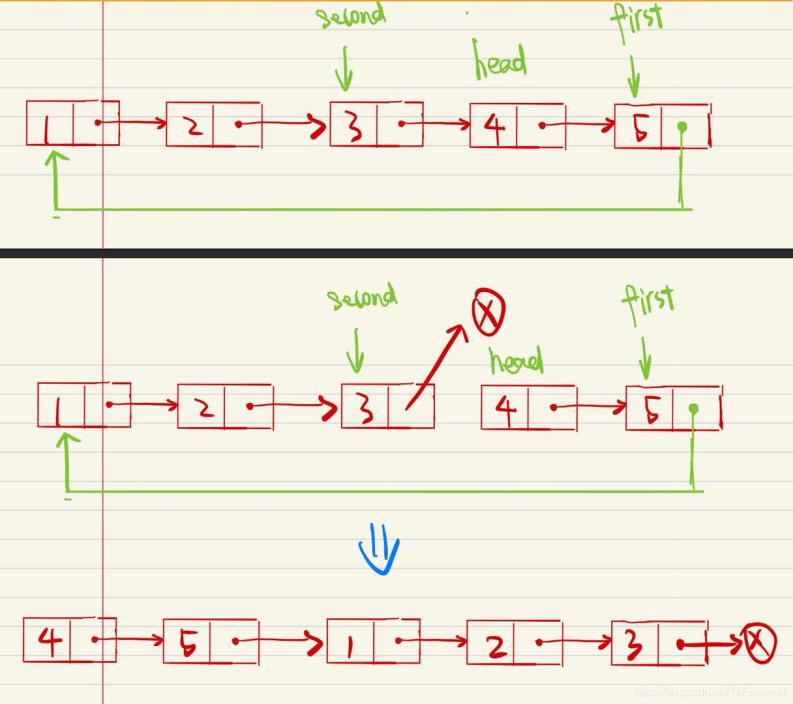


这时，两个指针之间的区域就是我们要移动的区域，只要更改指针指向，就完事了。

即，first->next指向head，完成旋转（当然还没完事）；

head指向second->next，头结点指向确认；

second->next指向空节点，尾结点指向确认。



记得返回头结点。

**代码：**

class Solution {

public:

ListNode\* rotateRight(ListNode\* head, int k) {

if(!head) return NULL;

int n=0;

for(auto p=head;p;p=p->next) n++;

k%=n; //如果k>n的时候需要取余

auto first=head,second=head;

//向后移动K位置

while(k--){

first=first->next;

}

while(first->next){

first=first->next;

second=second->next;

}

first->next=head; //链表尾部指针的next指向链表头部

head=second->next; //头指针移动k

second->next=NULL; //新的队尾

return head;

}

};

**另一种写法（推荐）：**

/\*\*

 \* 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\* rotateRight(ListNode\* head, int k) {

        if(nullptr ==  head)

            return nullptr;

        int n = 0;

        ListNode \*tmpNode = head;

        while(tmpNode)

        {

            n++;

            tmpNode = tmpNode->next;

        }

        k%=n;

        ListNode \*slow = head,\*fast = head;

        while(k--)

        {

            fast = fast->next;

        }

        while(slow && fast && fast->next)

        {

            slow = slow->next;

            fast = fast->next;

        }

        fast->next = head;

        head = slow->next;

        slow->next = nullptr;

        return head;

    }

};