**难度: 简单**

实现一种算法，找出单向链表中倒数第 k 个节点。返回该节点的值。

注意：本题相对原题稍作改动

示例：

输入： 1->2->3->4->5 和 k = 2

输出： 4

说明：

给定的 k 保证是有效的。

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/kth-node-from-end-of-list-lcci

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**思路1 (使用栈):**

1). 创建一个栈

2). 把head所以值全部放入到栈中

3). 因为栈是先进后出所以此时栈的数据和head的数据顺序相反

4). 循环把栈抛出k-1次>0

5). 下一次栈抛出就是所要的值

|  |
| --- |
| public int kthToLast(ListNode head, int k) {  Stack<Integer> stack = new Stack<>();  ListNode temp = head;  stack.push(head.val);  while(temp.next != null) {  stack.push(temp.next.val);  temp = temp.next;  }  while (k-1 > 0) {  stack.pop();  k--;  }  return stack.pop();  } |

**思路2:**

1). 先统计一下head总共有多少个

2). 使用 int count = nums - k; 就可以得到需要循环几次才能得到需要的值

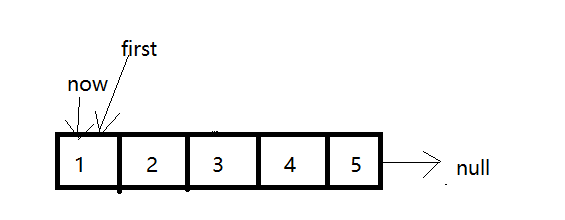
3). 把count循环count > 0 次每次转换结点

4). 直接返回结点的值就行了

|  |
| --- |
| public int kthToLast2(ListNode head, int k) {  ListNode temp = head;  int nums = 0;  while(temp != null) {  nums++;  temp = temp.next;  }  int count = nums - k;  while(count > 0) {  head = head.next;  count--;  }  return head.val;  } |

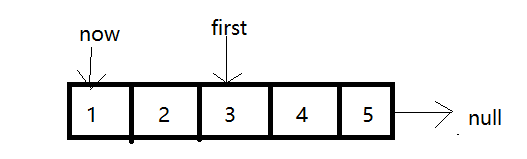
**思路3(使用快慢指针):**

1). 创建两个一个结点

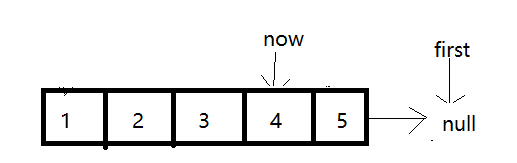


2). 将第一个结点(快指针)循环k次

假设k等于2,此时



3). 此时持续循环first知道first为null, 每次now也跟着下移



|  |
| --- |
| public int kthToLast3(ListNode head, int k) {  //快指针  ListNode first = head;  for (int i = 0; i < k; i++) {  first = first.next;  }  //当前指针  ListNode now = head;  while(first.next != null) {  first = first.next;  now = now.next;  }  return now.val;  } |