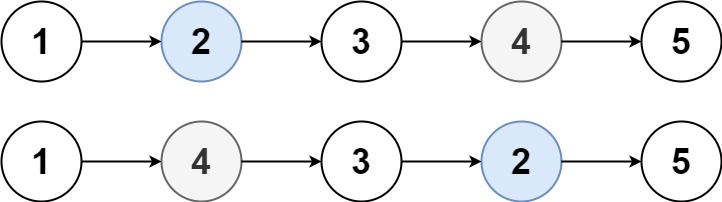
# 题目

给你链表的头节点 head 和一个整数 k 。

交换 链表正数第 k 个节点和倒数第 k 个节点的值后，返回链表的头节点（链表 从 1 开始索引）。

示例 1：



输入：head = [1,2,3,4,5], k = 2

输出：[1,4,3,2,5]

示例 2：

输入：head = [7,9,6,6,7,8,3,0,9,5], k = 5

输出：[7,9,6,6,8,7,3,0,9,5]

示例 3：

输入：head = [1], k = 1

输出：[1]

示例 4：

输入：head = [1,2], k = 1

输出：[2,1]

示例 5：

输入：head = [1,2,3], k = 2

输出：[1,2,3]

提示：

链表中节点的数目是 n

1 <= k <= n <= 105

0 <= Node.val <= 100

# 分析

## 方法一：快慢指针

**思路：**

**代码：**

/\*\*

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

int num=0;

for(auto p = head; p; p = p->next) num+=1; //确定链表长度

auto a=head;

for(int i=1; i<k; i+=1) a=a->next; //定位到第k个数

auto b=head;

for(int i=1; i<num-k+1; i+=1) b=b->next; //定位到倒数第k个数

swap(a->val,b->val); //交换值

return head;

}

};

另一种写法：

/\*\*

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

        int num = 0;

        ListNode \*tmpNode = head;

        for(;tmpNode;tmpNode=tmpNode->next)

        {

            num += 1;

        }

        ListNode \*slow = head;

        for(int i=1;i<k;i++)

        {

            slow = slow->next;

        }

        ListNode \*fast = head;

        for(int i=1;i<num-k+1;i++)

        {

            fast = fast->next;

        }

        swap(slow->val,fast->val);

        return head;

    }

};

错误写法：

/\*\*

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

        int num = 0;

        ListNode \*tmpNode = head;

        while(tmpNode->next)

        {

            num += 1;

            tmpNode = tmpNode->next;

        }

        ListNode \*slow = head;

        for(int i=1;i<k;i++)

        {

            slow = slow->next;

        }

        ListNode \*fast = head;

        for(int i=1;i<num-k+1;i++)

        {

            fast = fast->next;

        }

        swap(slow->val,fast->val);

        return head;

    }

};

## 方法二：数组

**思路：**

对于这种链表重排的操作，都可以采用数组存储其中的数据，然后通过数组的下标注方便访问各个节点的数据。这种方法特点就是非常容易理解，将链表操作转换为数组的操作，缺点是空间复杂度较大。

**代码：**

/\*\*

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

        if(nullptr == head)

            return nullptr;

        vector<ListNode\*> vec;

        ListNode \*dummpyNode = new ListNode(-1);

        dummpyNode->next = head;

        ListNode \*cur = dummpyNode;

        while(cur->next)

        {

            vec.push\_back(cur);

            cur = cur->next;

        }

        int vec\_size = vec.size();

        // 暂存第k,倒数k个节点

        ListNode \*kthNode = vec[k];

        ListNode \*reverseKthNode = vec[vec\_size-k+1];

        ListNode \*kthNextNode = vec[k+1];

        ListNode \*reverseKthNextNode  = vec[vec\_size-k+2];

        vec[k-1]->next = reverseKthNode;

        reverseKthNode->next = kthNode->next;

        vec[vec\_size-k]->next = kthNode;

        kthNode->next = reverseKthNextNode;

        return dummpyNode->next;

    }

};