# 题目

给定一个链表，两两交换其中相邻的节点，并返回交换后的链表。

你不能只是单纯的改变节点内部的值，而是需要实际的进行节点交换。

**示例:**

给定 1->2->3->4, 你应该返回 2->1->4->3.

# 分析

## 方法一：递归法

**思路：**

**代码：**

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

\* ListNode(int x) : val(x), next(NULL) {}

\* };

\*/

class Solution

{

public:

ListNode\* swapPairs(ListNode\* head)

{

///终止条件为所有都交换完了的链表

if(head == NULL || head->next == NULL)

return head;

///递归单元内两个节点做交换

//定义一个p结点为head->next(该递归单元内第二个节点)

ListNode \*p = head->next;

//记录第三个节点同时也是下一次递归的head

ListNode \*temp = p->next;

//使此单元内第二个节点指向第一个节点，即交换位置

p->next = head;

//交换完成后的head应指向下一次单元返回的头节点

head->next = swapPairs(temp);

//返回该单元的头节点

return p;

}

};

## 方法二：迭代法

**思路：**

**代码：**

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

\* ListNode(int x) : val(x), next(NULL) {}

\* };

\*/

class Solution

{

public:

ListNode\* swapPairs(ListNode\* head)

{

//新建一个空结点，用来指向头节点

ListNode\* p = new ListNode(0);

p->next = head;

//新建和p相同一个curr节点，两个相同的节点一个是当前做改变的节点，一个是保持不动用来返回的节点

ListNode\* curr = p;

//循环条件为当前节点为NULL或当前的下一个节点为NULL，分别对应偶数和奇数个节点的终止标志

while(head != NULL && head->next != NULL)

{

//为了清晰明了，我们新建两个节点，一节点和二节点

ListNode\* firstNode = head;

ListNode\* secondNode = head->next;

///把一和二进行交换，并连接前后

//当前curr节点指向二节点

curr->next = secondNode;

//一节点指向二节点此时的下一节点

firstNode->next = secondNode->next;

//二节点指向一节点，即交换位置成功

secondNode->next = firstNode;

//由于每次循环curr节点都指向每次循环的一节点，所以要再次把curr节点指向一节点

curr = firstNode;

//每次移动都是由head节点来赋值操作，所以head应向右移动两格，即新循环的一节点

head = firstNode->next;

}

//返回p的下一个节点即对应整个操作后的链表

return p->next;

}

};