206. Reverse Linked List (need to review)

92. Reverse Linked List II

206.link solution youtube

92.link

我的错误写法

```
while(head != null){
    next = head.next;
    head = previous;
//previous.next造成闭环, 相当于previous.next = previous
    previous.next = head;
    head = next;
}
```

正确写法, 理解 previous是纯指针, 不参与next的作用. 所有操作发生发head

```
while(head != null){
    next = head.next;
    head.next = previous;//这里
    previous = head;
    head = next;
}
```

递归版本的理解:

传入这个节点和他之前的节点, 如果这个节点后面还有节点, 然后让他本身指向他前面的节点, 如果他后面还有节点, 就继续传入他的next和他自己, 否则就返回.

```
public ListNode reverseList(ListNode head) {
    /* recursive solution */
    return reverseListInt(head, null);
}
```

```
private ListNode reverseListInt(ListNode head, ListNode previous) {
    if (head == null)
        return previous;
    ListNode next = head.next;
    head.next = previous;
    return reverseListInt(next, head);
}
```

还有一种写法更难理解

思路：找到最后的节点，然后把他反转.
1 -> 2 -> 3 -> 4
首先是一个递归，最后确定return的Head是4.
然后处理此时node是3
//node.next.next = node
1 -> 2 -> 3 -> 4 变成 1 -> 2 -> 3 <-> 4.
//node.next = null
1 -> 2 - > 3 -> null, 4 -> 3 -> null.
继续，此时return的值仍然是4，但是node变成了2，
//2.next.next = 2 也就是 3.next = 2
变成 1 -> 2 -> 3 <-> 2, 4 -> 3 <-> 2
// 2.next = null
1 -> 2 ->null, 4 -> 3 -> 2 -> null;
最后 //1.next.next = 1
1 <-> 2, 4 -> 3 -> 2 <-> 1
//1.next = null
1 -> null, 4-> 3 -> 2 -> 1 -> null.

代码:

```
public ListNode reverseList(ListNode node){
    //这个node==null是为了解决一开始node就是null.
    if(node == null || node.next == null) return node;
    ListNode reversedListHead = reverseList(node.next);
    node.next.next = node;
    node.next = null;
    return reversedListHead;
}
```

92

需要注意的点: 我是从fake开始循环的, 用了beforeReverse标记开始循环之前, reversedTail是i = m也就是开始reverse的头.

之前的疑惑

```
beofreReverse = current;
current = current.next;
```
这个时候beforeReverse不会因为current变了而改变.
但是如果是
```
beforeReverse = current;
current.val = 123;
```
这个时候beforeReverse.val 就会因为current.val的改变改变.

code:

```
    public ListNode reverseBetween(ListNode head, int m, int n) {
        if(m == n) return head;
        ListNode fake = new ListNode(0, head);
        ListNode current = fake;
        ListNode beforeReverse = fake;
        ListNode reversedTail = new ListNode();
        ListNode reversedHead = null, next = null;
        for(int i = 0; i <= n; ++i){
            if(i <= m - 1){
                if(i == m - 1) beforeReverse = current;
                current = current.next;
            }else if(i >= m){
                if(i == m) reversedTail = current;
                next = current.next;
                current.next = reversedHead;
                reversedHead = current;
                current = next;
            }
        }
        beforeReverse.next = reversedHead;
        reversedTail.next = current;
        return fake.next;
    }
```