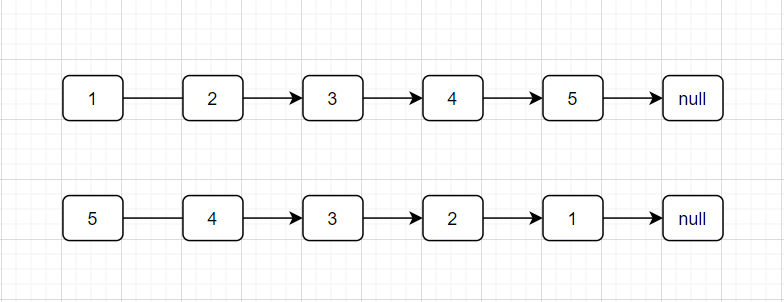
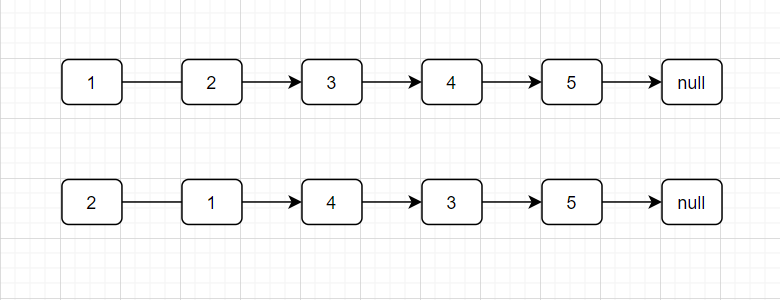
**【链表翻转】题目心得（Java解法）**

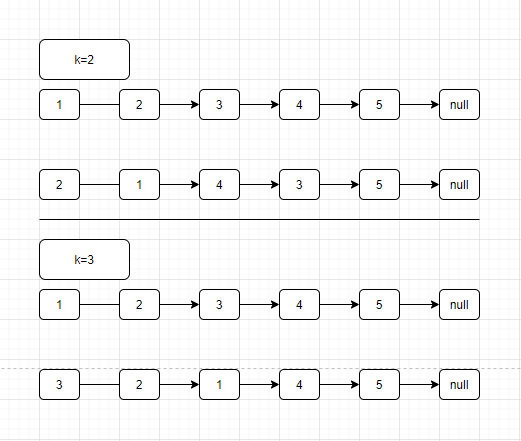
第一题：反转链表（leetcode-206）



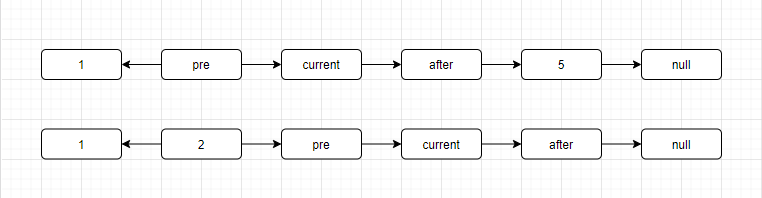
第二题：给定一个链表，两两交换其中相邻的节点，并返回交换后的链表（你不能只是单纯的改变节点内部的值，而是需要实际的进行节点交换）(leetcode-24)



第三题：给你一个链表，每 k 个节点一组进行翻转，请你返回翻转后的链表（k 是一个正整数，它的值小于或等于链表的长度；如果节点总数不是 k 的整数倍，那么请将最后剩余的节点保持原有顺序）。(leetcode-25)



对于链表翻转类的题目，核心点在于掌握连续三个节点间的关系。定义某一节点为当前节点（别名current），当前节点的前一节点（别名pre），当前节点的下一节点（别名after），暂不考虑当前节点在链表头尾的情况，如图：



**第一题解法：**

定义current.next = pre 即可完成current节点指向pre节点；现在只需考虑让current节点动起来，即如何让current节点分别代表链表中的每个节点，依次完成current.next = pre的执行计划，整个链表即完成翻转。

1. after = current.next，after节点保存current节点的原下一节点；
2. current.next = pre, 完成当前节点指向该节点前一节点的期望；

4，current = after（current节点依次向后遍历）；

1. 由于current节点已经移动after节点，pre节点也要相应移动，即pre = current（该操作在第四步之前）；

考虑current头结点的情况，此时pre节点为空即可，代码如下：

public ListNode reverseList(ListNode head) {

    if(head == null || head.next == null) {

        return null;

    }

    ListNode pre = null, current = head, after;

    while(current != null) {

        after = current.next;

        current.next = pre;

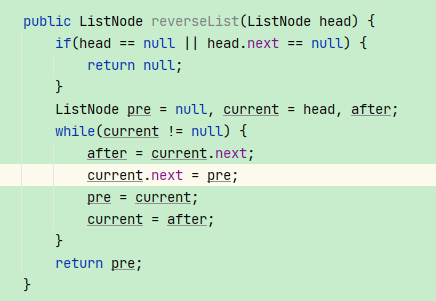
        pre = current;

        current = after;

    }

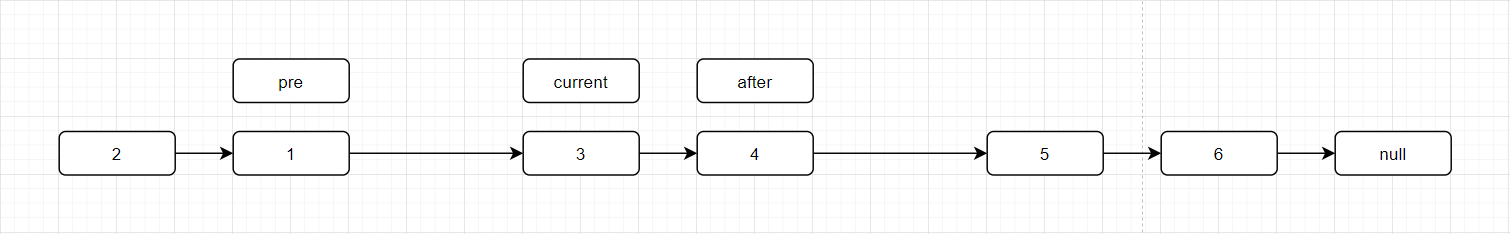
    return pre;

}



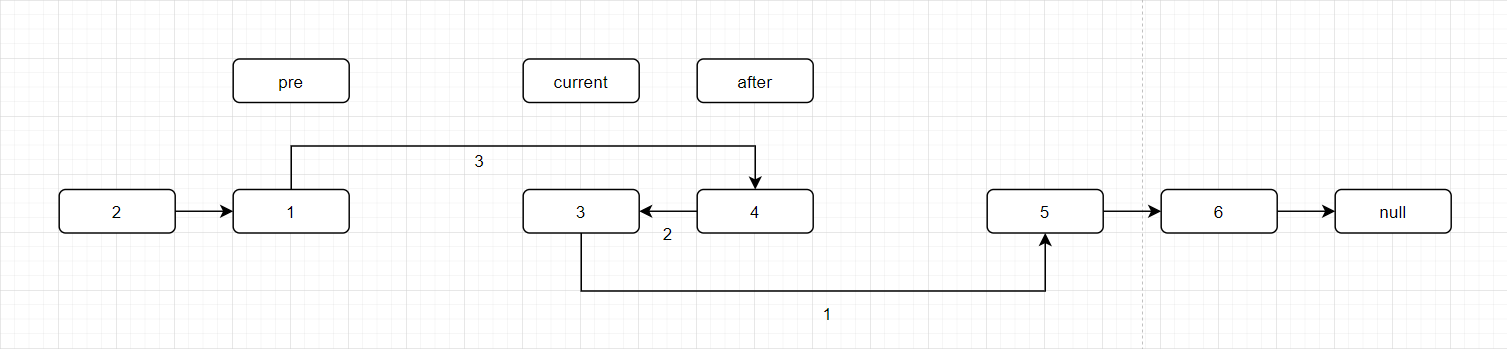
**第二题解法：**

同样利用pre,current,after之间的密切关系；首先忽略头结点，思考中间节点两两交换的规则。



首先忽略头结点，思考中间节点两两交换的规则。假设第一对节点已经交换完毕，此时pre，current，after节点如图所示，考虑如何将current，after节点交换。如上图所示。

1. current.next = after.next
2. after.next = current
3. pre.next = after



经历1,2,3步，链表第三，第四个节点已经完成转换。此时考虑和第一题一样的问题，如何让current节点动起来。

1. current = current.next
2. after = current.next

循环遍历上述1-5步即可（考虑链表个数为奇数的情况，需要适当的判空）。

public ListNode reverseList(ListNode head) {

    if(head == null || head.next == null) {

        return null;

    }

    ListNode pre = null, current = head, after;

    while(current != null) {

        after = current.next;

        current.next = pre;

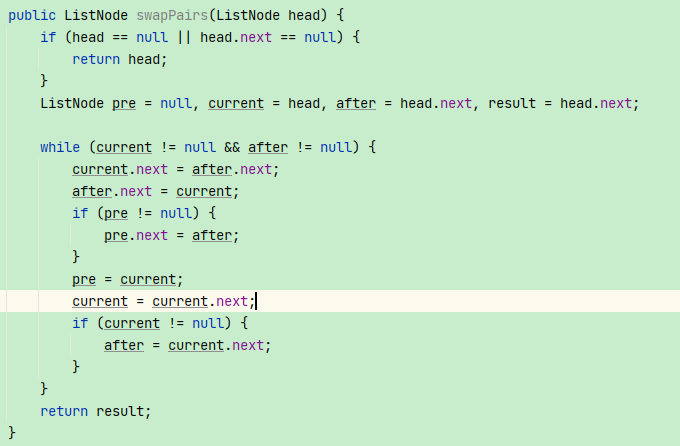
        pre = current;

        current = after;

    }

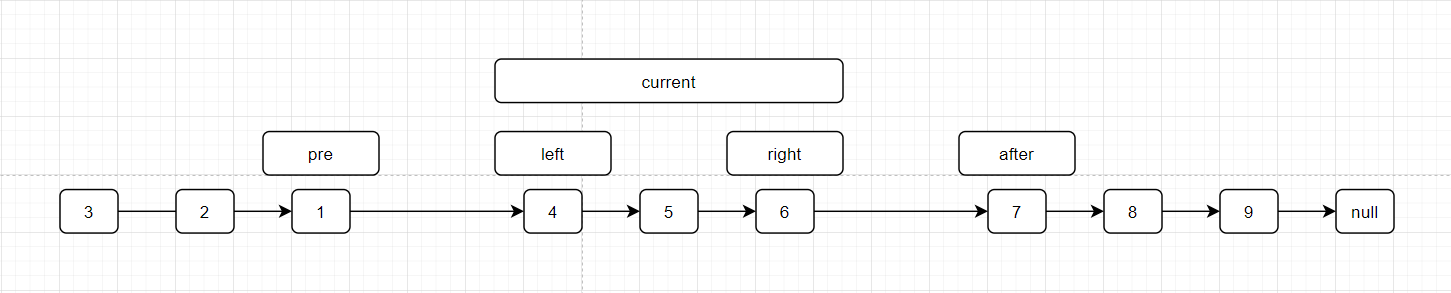
    return pre;

}



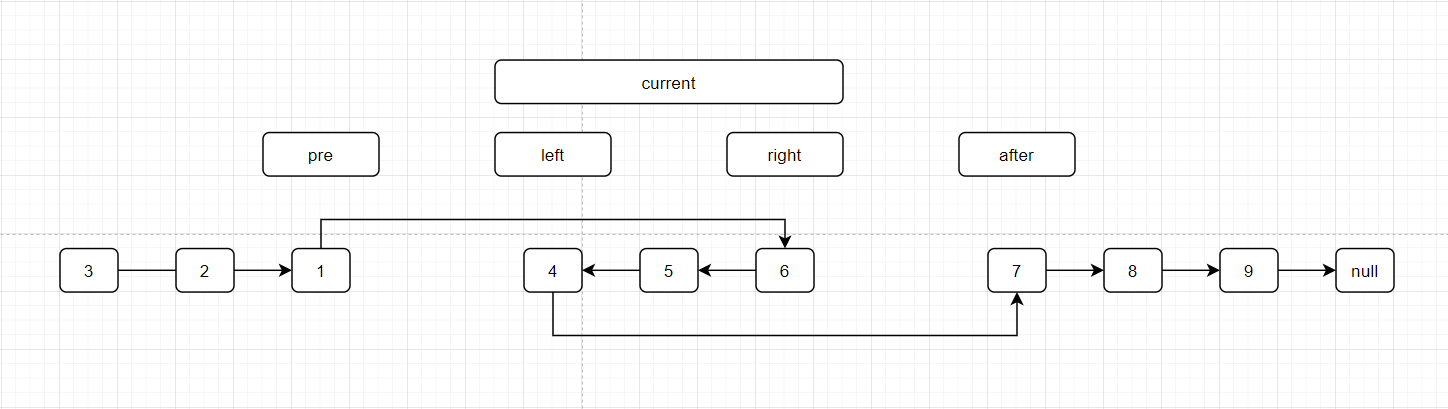
**第三题解法**

第三题属于第一题与第二题的结合。也是充分理解pre,current,after节点之间的关系，此时current不再是一个单独的节点，而是一串节点，如图。

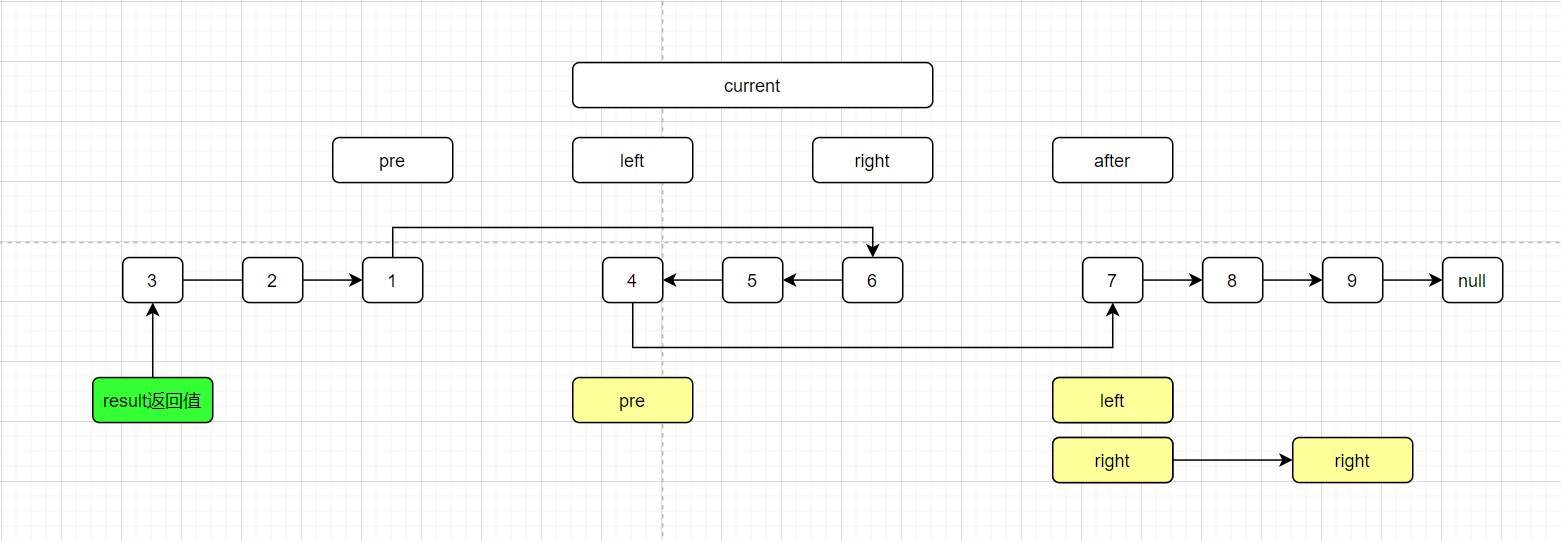


当k=3时，假设前三个节点已经完成翻转，考虑第3,4,5（索引由0开始）个节点如何翻转。

1. left,right节点一开始均指向值为4的节点，根据k的值向后遍历确定right的值（right值需判空）
2. after = right.next
3. 根据第一题思想将left节点至right节点完成翻转
4. 调整pre，left节点的指向，完成第二次翻转，如图



1. 考虑如何让current动起来



代码如下：

public ListNode reverseKGroup(ListNode head, int k) {

    if (head == null || head.next == null || k < 2) {

        return head;

    }

    ListNode left = head, right = head, pre = null, after, result = null;

//isHead是为了保证第一次翻转后，保存返回值的位置。

    boolean isHead = true;

    while (right != null) {

        for (int i = 1; i < k && right != null; i++) {

            right = right.next;

        }

        if (right == null) {

            return result;

        }

        after = right.next;

        reverseLeftRight(left, right.next);

        if (isHead) {

            result = right;

            isHead = false;

        }

        if (pre != null) {

            pre.next = right;

        }

        pre = left;

        left.next = after;

        left = after;

        right = after;

    }

    return result;

}

private void reverseLeftRight(ListNode left, ListNode right) {

    ListNode pre = null, current = left, after;

    while (current != right) {

        after = current.next;

        current.next = pre;

        pre = current;

        current = after;

    }

}

好吧，我承认第三题我讲的晦涩难懂，也是第一次写类似题解的文章，最近在专攻链表的题目，有感而发。希望读者务必按照顺序一一理解。

**充分理解pre,current,after节点之间的关系。**

**充分理解pre,current,after节点之间的关系。**

**充分理解pre,current,after节点之间的关系。**

纸笔配合，理解其中的关系。同时记得判空，以及在代码开始阶段考虑各个题目的特殊情况。

来自一个自以为很可爱，很帅气，但又是个菜菜的程序员。