SlidesPractice_L2_Recursion_Diagram

4/4/17 9:40 AM

Notes for Recursion

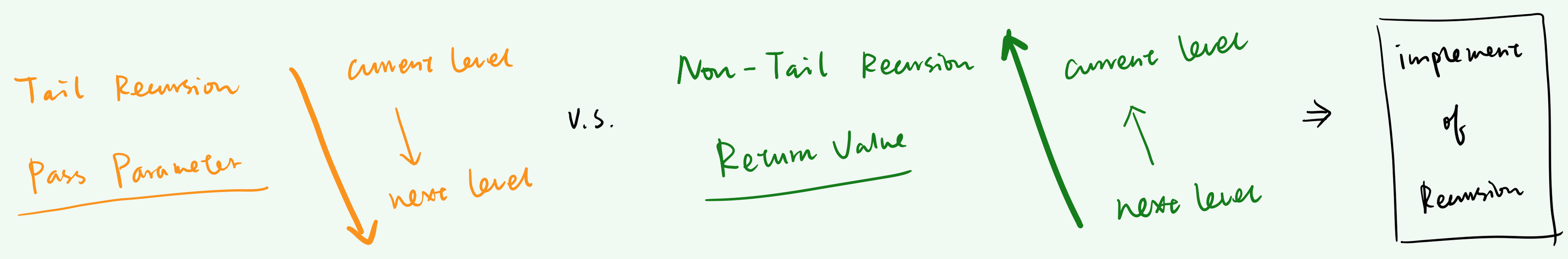
1. Why Recursion?

(1) Big problem —divided into→ Similar subproblems.

(2) Focus on solving the subproblems instead of the whole scope.

2. How Recursion?

(1) What's the subproblem? ⇒ Do what on current level?



(2) What represents next subproblem? ⇒ Pass what into next level?

Tail Recursion's main idea: Skip repeated operation, just call the method itself, pass new prameters at each call.

Non-Tail Recursion's main idea: Assuming all following levels are already handled, with return value.

3. 3-step:

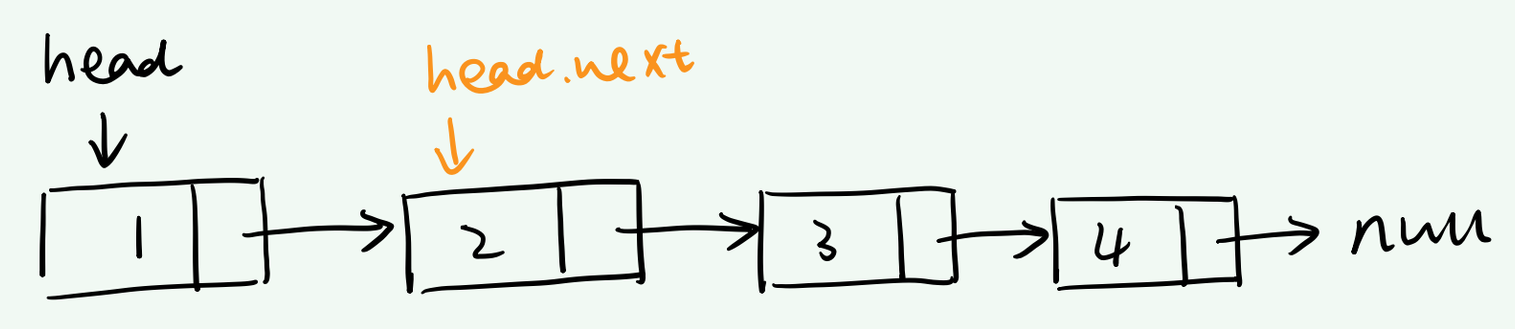
step 1: base case (exit, or return)

step 2: current layer (core logic)

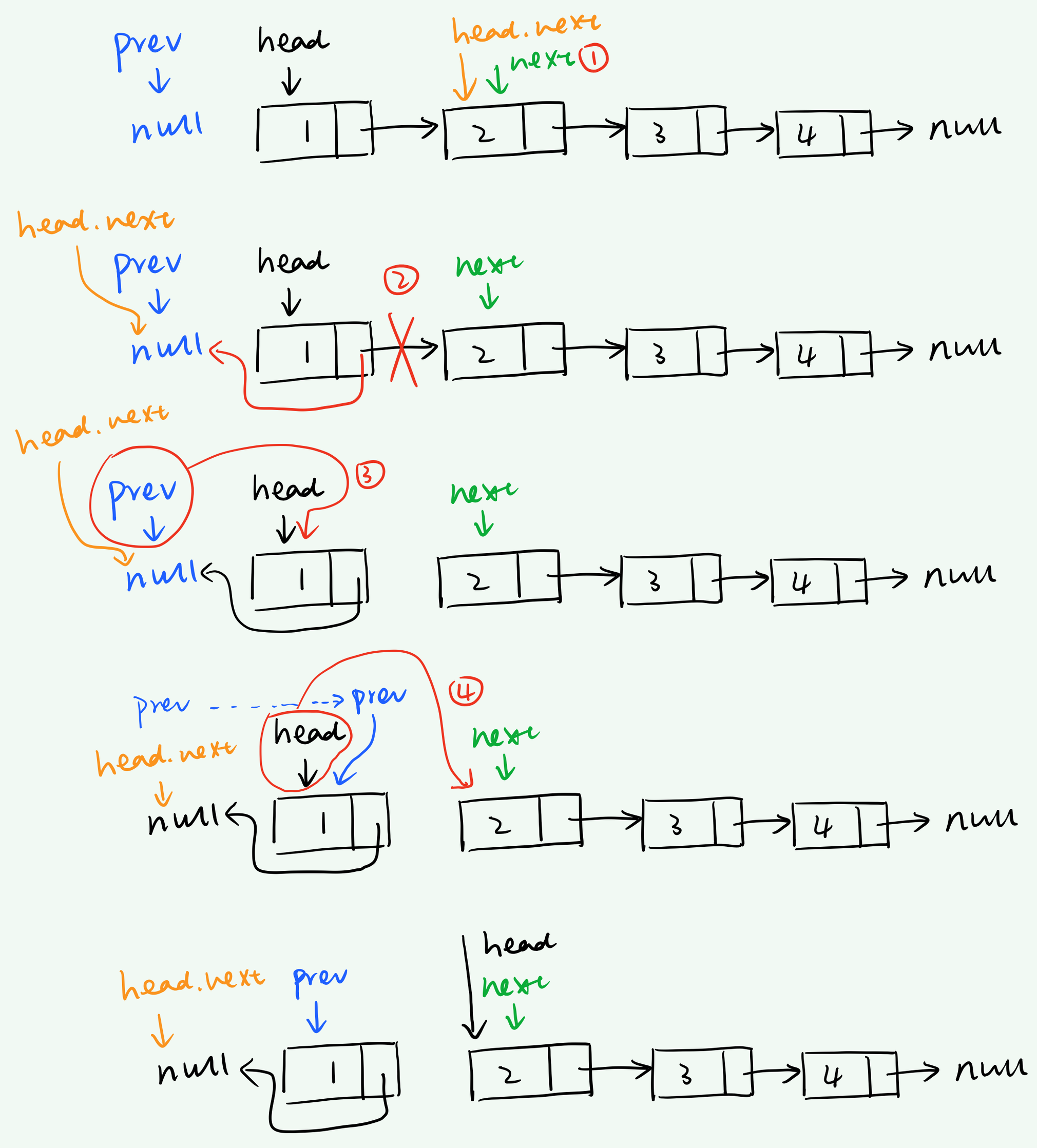
step 3: next layer

Examples of Recursion

Reverse LinkedList_Tail Recursion




```
ListNode prev = null;
ListNode next = null;
while (head != null) {
```



① 给 head.next 找替身

next = head.next;

② 断开原来的 head → head.next 的指向，令 head.next 指向 prev.

head.next = prev;

③ keep prev moving forward to do traversal of the LinkedList, prev's next node is "head"

prev = head;

④ keep head moving forward to do traversal of the LinkedList, head's next node is "next"

head = next;

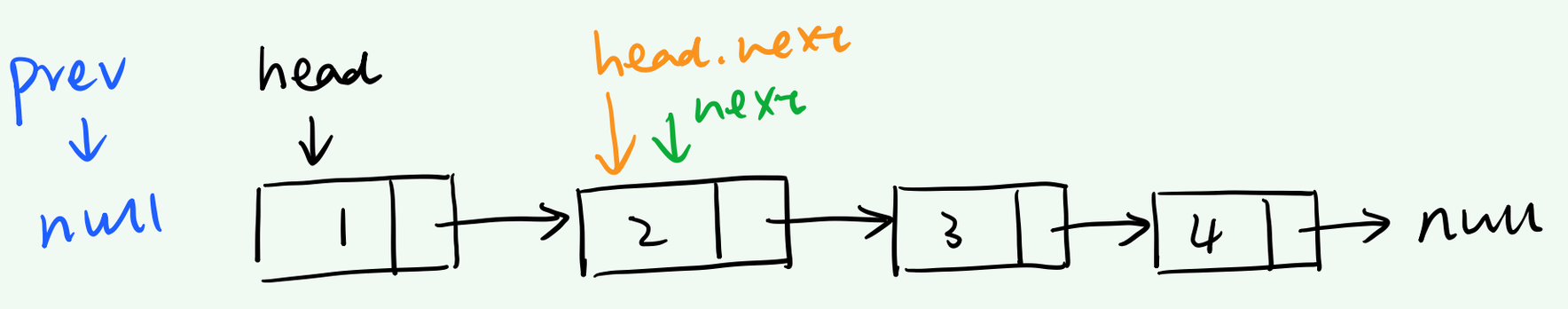
这4步"环环相扣"

```
}
return prev;
```

Iteration

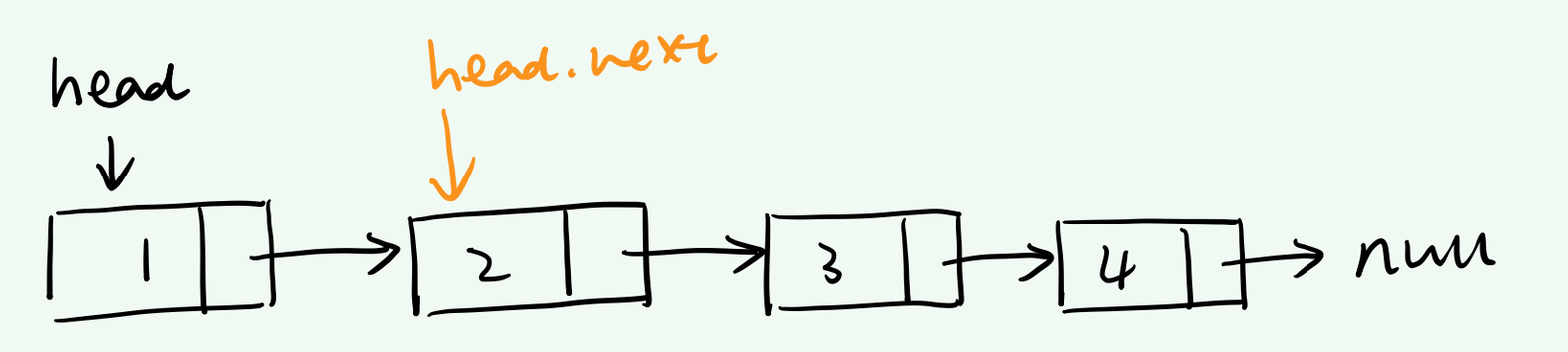
```
if (head == null || head.next == null) {
    return head;
}
ListNode prev = null;
ListNode next = null;
while (head != null) {
    // Reverse
    next = head.next;
    head.next = prev;
    // Prepare for next iteration
    prev = head;
    head = next;
}
return prev;
```

Tail Recursion

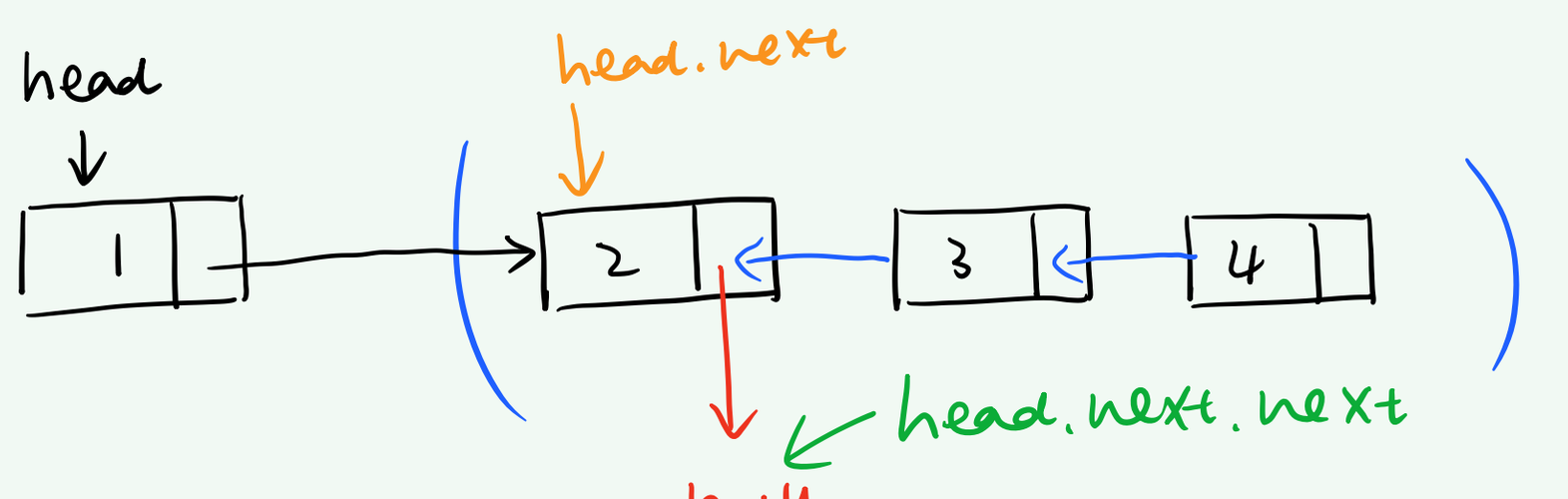



```
private Node reverseHelper(Node head, Node prev) {
    if (head == null) {
        return head;
    }
    Node next = head.next;
    head.next = prev;
    return reverseHelper(next, head);
}
```

Reverse LinkedList_Non-Tail Recursion



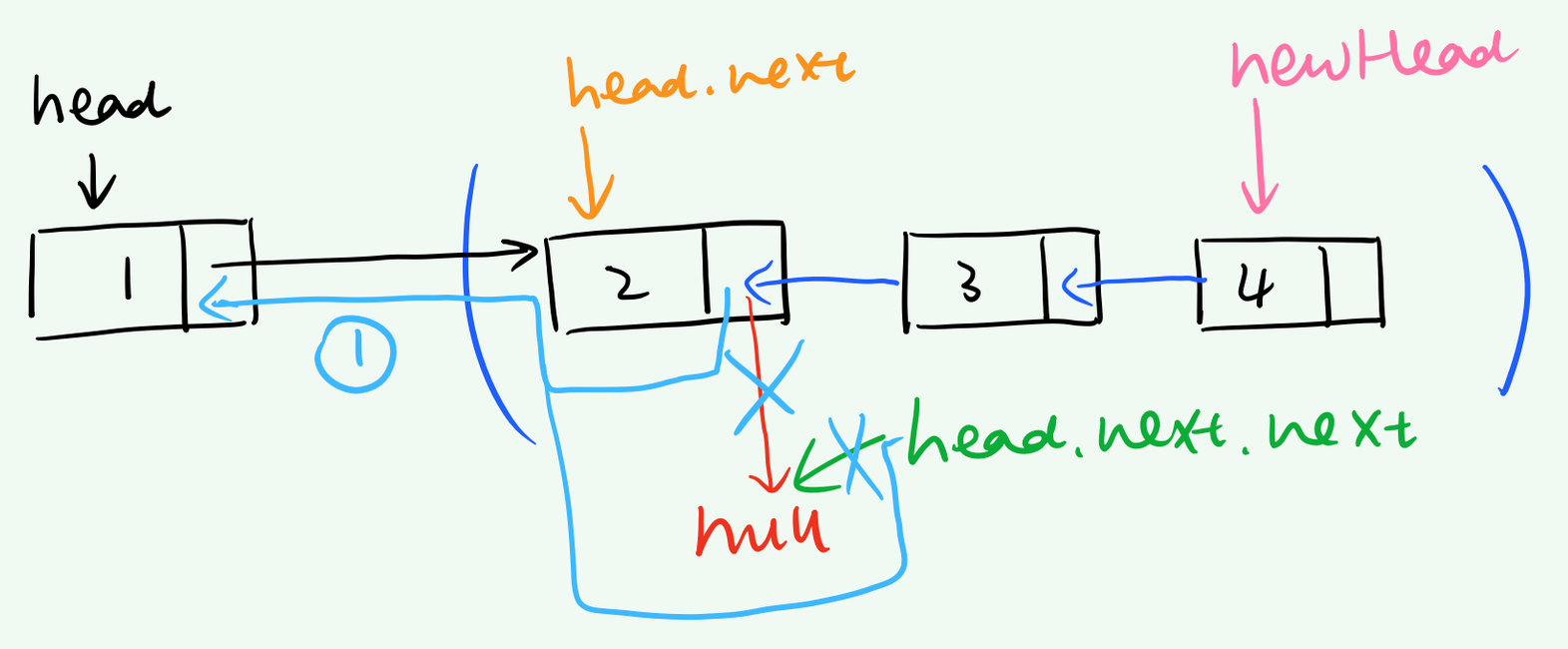
假设 head 后面的部分都已经 reverse 好了.



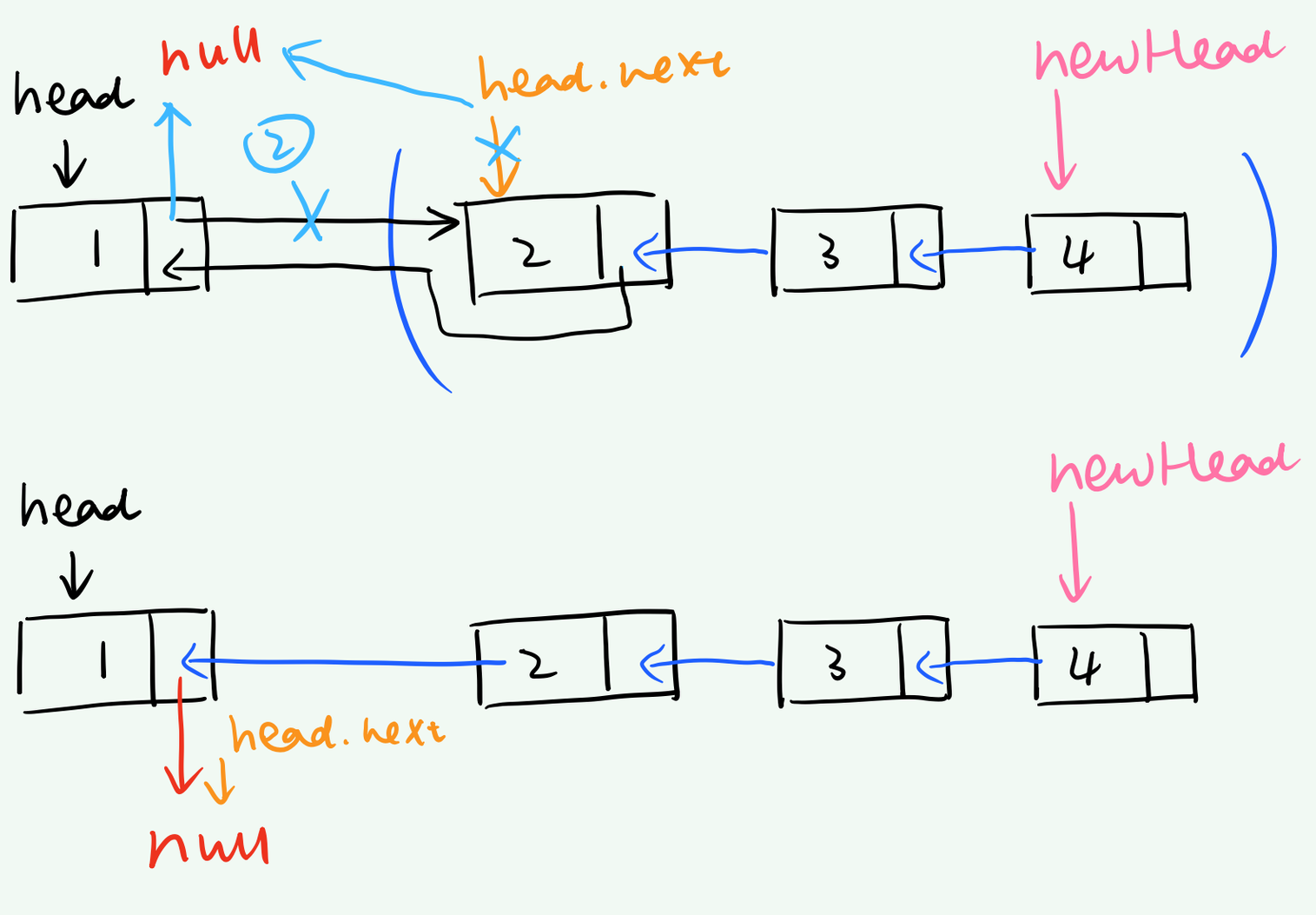
此时, head 指向"1", 且 head.next 指向 2, 但 head.next.next 指向 null

要做的事:

① Let head.next.next 指向 head



② Let head.next 指向 null



③ return newHead

```
public Node reverseRecursive(ListNode head) {
    // Base Case
    if (head = null || head.next == null) {
        return head;
    }
    ListNode newHead = reverseRecursive(head.next);
    // Reverse
    head.next.next = head;
    head.next = null;
    return newHead;
}
```