**难度: 简单**

编写代码，移除未排序链表中的重复节点。保留最开始出现的节点。

示例1:

输入：[1, 2, 3, 3, 2, 1]

输出：[1, 2, 3]

示例2:

输入：[1, 1, 1, 1, 2]

输出：[1, 2]

提示：

链表长度在[0, 20000]范围内。

链表元素在[0, 20000]范围内。

进阶：

如果不得使用临时缓冲区，该怎么解决？

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/remove-duplicate-node-lcci

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**思路1:**

1). 先判断当前的头结点是否为空,如果是,直接返回null就行了

2). 如果不是,创建一个临时变量用于操作ListNode,

3). 创建一个Set集合, 把head的值存储进去Set

4). 循环用set进行判断下一个结点的值是否包含在set里面

5) 如果已经存在直接temp的下一个结点指向temp的下一个结点的下一个结点

6). 如果不存在,就把temp的下一个结点的值存进行存储,然后把temp的下一个结点给到temp

7). 最后返回head就行了

|  |
| --- |
| public ListNode removeDuplicateNodes(ListNode head) {  if (head == null) {  return null;  }  ListNode temp = head;  HashSet<Integer> set =new HashSet<>();  set.add(head.val);  while (temp.next != null) {  if (set.contains(temp.next.val)) {  temp.next = temp.next.next;  }else {  set.add(temp.next.val);  temp = temp.next;  }  }  return head;  } |

**思路2(效率更快):**

1). 先判断当前的头结点是否为空,如果是,直接返回null就行了

2). 创建一个set集合

3). 先把head的值放入到set集合中

4). 创建一个临时结点(temp)指向head

5). 循环temp的下一个结点如果不等于null

6). 判断temp下一个结点的值添加进set是否true

1. true则把temp的下一个结点给到temp

2. false则把temp的下一个结点指向 temp的下一个结点的下一个结点

7). 最后返回head

|  |
| --- |
| public ListNode removeDuplicateNodes(ListNode head) {  if (head == null) {  return null;  }  HashSet<Integer> set = new HashSet<>();  set.add(head.val);  ListNode temp = head;  while(temp.next != null) {  if (set.add(temp.next.val)) {  temp = temp.next;  }else {  temp.next = temp.next.next;  }  }  return head;  } |