83 删除排序链表中的重复元素

Label：链表 双指针

给定一个排序链表，删除所有重复的元素，使得每个元素只出现一次。

```
输入: 1->1->2
输出: 1->2

输入: 1->1->2->3->3
输出: 1->2->3
```

- 遍历

```
/**
 * Definition for singly-linked list.
 * public class ListNode {
 *     int val;
 *     ListNode next;
 *     ListNode() {}
 *     ListNode(int val) { this.val = val; }
 *     ListNode(int val, ListNode next) { this.val = val; this.next = next; }
 * }
 */
class Solution {
    public ListNode deleteDuplicates(ListNode head) {

        if (head == null || head.next == null) {
            return head;
        }

        ListNode curr = head;

        while (curr != null && curr.next != null) {
            if (curr.val == curr.next.val) {
                curr.next = curr.next.next; // 重复元素，跳过
            } else {
                curr = curr.next;  // 放在else里，防止 出现连续三个以上的相同结点
            }
        }
        return head;
    }
}
```

- 用set判断是否重复

```
class Solution {
    public ListNode deleteDuplicates(ListNode head) {
        if(head == null || head.next == null)
            return head;

        Set<Integer> set = new HashSet<>();
        ListNode curr = head.next;
        ListNode pre  = head;
        set.add(head.val);

        while(curr != null ){
            if(set.contains(curr.val))
                pre.next = curr.next;   // pre 是不动的
            else {
                set.add(curr.val);  // 记录
                pre = curr;
            }
            curr = curr.next;
        }
        return head;
    }
}
```

- 递归

```
class Solution {
    public ListNode deleteDuplicates(ListNode head) {
        if (head == null || head.next == null) {
            return head;
        }

        head.next = deleteDuplicates(head.next);

        return head.val == head.next.val ? head.next : head;
    }
}
```