# 0147. 对链表进行插入排序

ITCharge 大约 2 分钟

- 标签：链表、排序
- 难度：中等

## 题目链接

- 0147. 对链表进行插入排序 - 力扣

## 题目大意

**描述**：给定链表的头节点 `head` 。

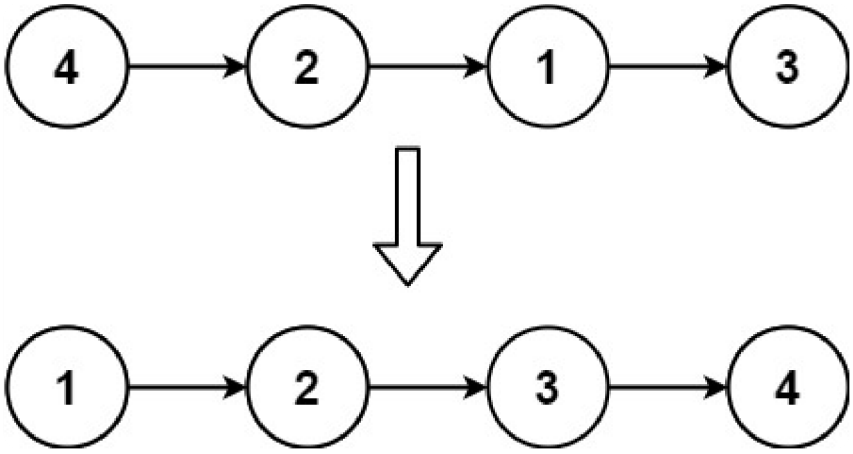
**要求**：对链表进行插入排序。

**说明**：

- 插入排序算法：
  - 插入排序是迭代的，每次只移动一个元素，直到所有元素可以形成一个有序的输出列表。
  - 每次迭代中，插入排序只从输入数据中移除一个待排序的元素，找到它在序列中适当的位置，并将其插入。
  - 重复直到所有输入数据插入完为止。
- 列表中的节点数在 $[1, 5000]$ 范围内。
- $-5000 \leq Node.val \leq 5000$。

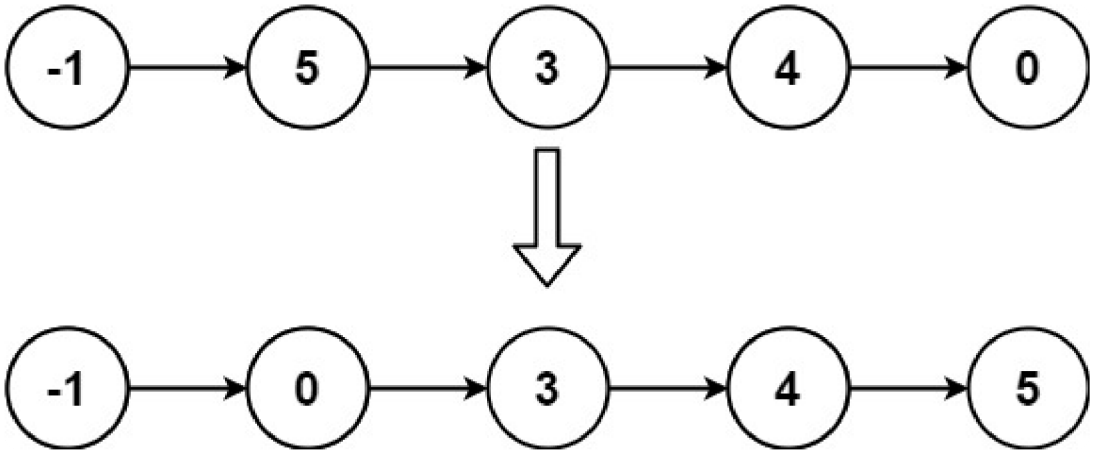
**示例**：

- 示例 1：




```py
输入：head = [4,2,1,3]
输出：[1,2,3,4]
```

- 示例 2：




```py
输入：head = [-1,5,3,4,0]
输出：[-1,0,3,4,5]
```

# 解题思路

## 思路 1：链表插入排序

1. 先使用哑节点 `dummy_head` 构造一个指向 `head` 的指针，使得可以从 `head` 开始遍历。
2. 维护 `sorted_list` 为链表的已排序部分的最后一个节点，初始时，`sorted_list = head` 。

3. 维护 `prev` 为插入元素位置的前一个节点，维护 `cur` 为待插入元素。初始时， `prev = head` ， `cur = head.next` 。
4. 比较 `sorted_list` 和 `cur` 的节点值。
    - 如果 `sorted_list.val <= cur.val` ，说明 `cur` 应该插入到 `sorted_list` 之后，则将 `sorted_list` 后移一位。
    - 如果 `sorted_list.val > cur.val` ，说明 `cur` 应该插入到 `head` 与 `sorted_list` 之间。则使用 `prev` 从 `head` 开始遍历，直到找到插入 `cur` 的位置的前一个节点位置。然后将 `cur` 插入。
5. 令 `cur = sorted_list.next` ，此时 `cur` 为下一个待插入元素。
6. 重复 4、5 步骤，直到 `cur` 遍历结束为空。返回 `dummy_head` 的下一个节点。

## 思路 1：代码

py

```
def insertionSortList(self, head: ListNode) -> ListNode:
    if not head or not head.next:
        return head

    dummy_head = ListNode(-1)
    dummy_head.next = head
    sorted_list = head
    cur = head.next

    while cur:
        if sorted_list.val <= cur.val:
            # 将 cur 插入到 sorted_list 之后
            sorted_list = sorted_list.next
        else:
            prev = dummy_head
            while prev.next.val <= cur.val:
                prev = prev.next
            # 将 cur 到链表中间
            sorted_list.next = cur.next
```

```
                cur.next = prev.next
                prev.next = cur
            cur = sorted_list.next

        return dummy_head.next
```

## 思路 1：复杂度分析

- **时间复杂度**：$O(n^2)$。
- **空间复杂度**：$O(1)$。