0142. 环形链表 II

ITCharge 大约 2 分钟

- 标签：哈希表、链表、双指针
- 难度：中等

题目链接

- 0142. 环形链表 II - 力扣

题目大意

描述：给定一个链表的头节点 `head` 。

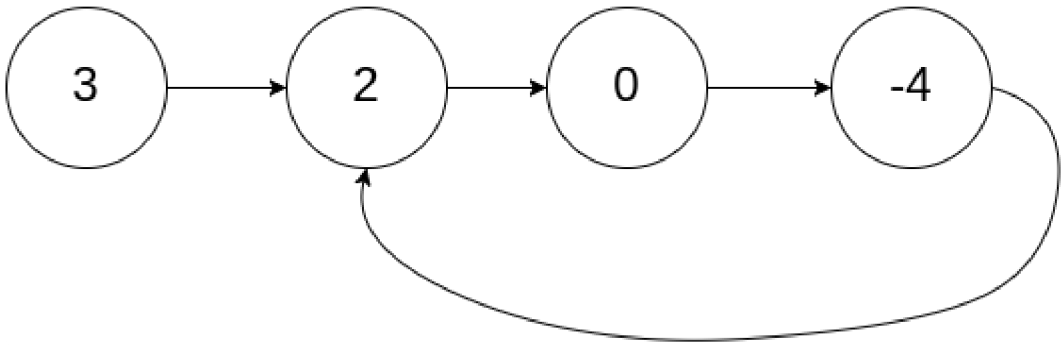
要求：判断链表中是否有环，如果有环则返回入环的第一个节点，无环则返回 `None` 。

说明：

- 链表中节点的数目范围在范围 $[0,]$ 为。
- $-10^5 \leq Node.val \leq 10^5$。
- `pos` 的值为 `-1` 或者链表中的一个有效索引。

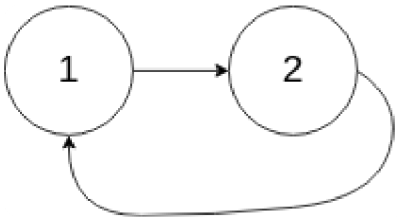
示例：

- 示例 1：




```py
输入：head = [3,2,0,-4], pos = 1
输出：返回索引为 1 的链表节点
解释：链表中有一个环，其尾部连接到第二个节点。
```

- 示例 2：




```py
输入：head = [1,2], pos = 0
输出：返回索引为 0 的链表节点
解释：链表中有一个环，其尾部连接到第一个节点。
```

解题思路

思路 1：快慢指针（Floyd 判圈算法）

1. 利用两个指针，一个慢指针 `slow` 每次前进一步，快指针 `fast` 每次前进两步（两步或多步效果是等价的）。
2. 如果两个指针在链表头节点以外的某一节点相遇（即相等）了，那么说明链表有环。
3. 否则，如果（快指针）到达了某个 `None` 后继指针的节点时，那么说明没环。
4. 如果有环，则再定义一个指针 `ans`，和慢指针一起每次移动一步，两个指针相遇的位置即为入口节点。

这是因为：假设入环位置为 `A`，快慢指针在 `B` 点相遇，则相遇时慢指针走了 $a+b$ 步，快指针走了 $a+n(b+c)+b$ 步。

因为快指针总共走的步数是慢指针走的步数的两倍，即 $2(a+b)=a+n(b+c)+b$，所以可以推出：$a=c+(n-1)(b+c)$。

我们可以发现：从相遇点到入环点的距离 c 加上 $n-1$ 圈的环长 $b+c$ 刚好等于从链表头部到入环点的距离。

思路 1：代码

```py
class Solution:
    def detectCycle(self, head: ListNode) -> ListNode:
        fast, slow = head, head
        while True:
            if not fast or not fast.next:
                return None
            fast = fast.next.next
            slow = slow.next
            if fast == slow:
                break

        ans = head
        while ans != slow:
            ans, slow = ans.next, slow.next
        return ans
```

思路 1：复杂度分析

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