0141. 环形链表

ITCharge 大约 2 分钟

- 标签：哈希表、链表、双指针
- 难度：简单

题目链接

- 0141. 环形链表 - 力扣

题目大意

描述：给定一个链表的头节点 `head` 。

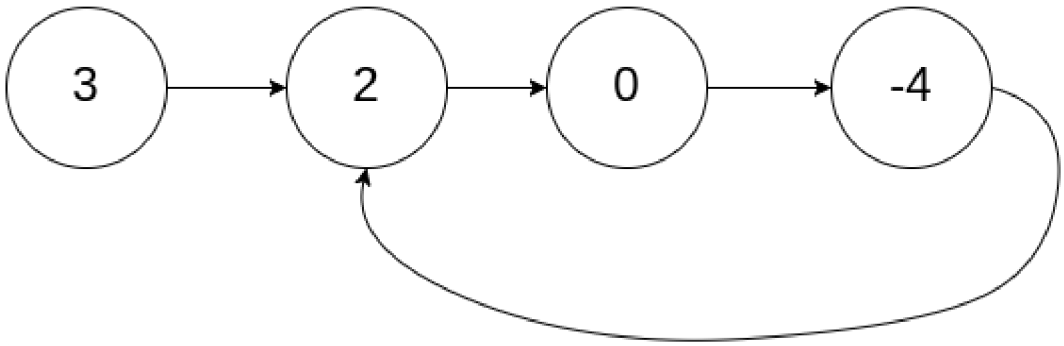
要求：判断链表中是否有环。如果有环则返回 `True` ，否则返回 `False` 。

说明：

- 链表中节点的数目范围是 $[0, 10^4]$。
- $-10^5 \leq Node.val \leq 10^5$。
- `pos` 为 `-1` 或者链表中的一个有效索引。

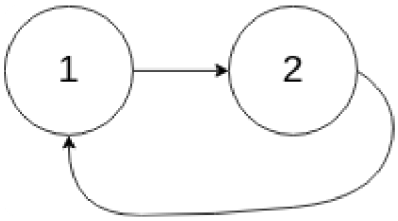
示例：

- 示例 1：




```py
输入：head = [3,2,0,-4], pos = 1
输出：True
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：True
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：哈希表

最简单的思路是遍历所有节点，每次遍历节点之前，使用哈希表判断该节点是否被访问过。如果访问过就说明存在环，如果没访问过则将该节点添加到哈希表中，继续遍历判断。

思路 1：代码

```py
class Solution:
    def hasCycle(self, head: ListNode) -> bool:
        nodeset = set()

        while head:
            if head in nodeset:
                return True
            nodeset.add(head)
            head = head.next
        return False
```

思路 1：复杂度分析

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

思路 2：快慢指针（Floyd 判圈算法）

这种方法类似于在操场跑道跑步。两个人从同一位置同时出发，如果跑道有环（环形跑道），那么快的一方总能追上慢的一方。

基于上边的想法，Floyd 用两个指针，一个慢指针（龟）每次前进一步，快指针（兔）指针每次前进两步（两步或多步效果是等价的）。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，否则，如果（快指针）到达了某个没有后继指针的节点时，那么说明没环。

思路 2：代码

```py
class Solution:
    def hasCycle(self, head: ListNode) -> bool:
        if head == None or head.next == None:
            return False

        slow = head
        fast = head.next

        while slow != fast:
            if fast == None or fast.next == None:
                return False
            slow = slow.next
            fast = fast.next.next

        return True
```

思路 2：复杂度分析

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