876、链表的中间节点

一、题目描述

给定一个带有头结点 `head` 的非空单链表，返回链表的中间结点。

如果有两个中间结点，则返回第二个中间结点。

二、实现

方法一：数组

1、思路和算法

链表的缺点在于不能通过下标访问对应的元素。因此我们可以考虑对链表进行遍历，同时将遍历到的元素依次放入数组 `A` 中。如果我们遍历到了 `N` 个元素，那么链表以及数组的长度也为 `N`，对应的中间节点即为 `A[N/2]`。

2、代码实习

```
class Solution {
    public ListNode middleNode(ListNode head) {
        ListNode[] A = new ListNode[100];
        int t = 0;
        while (head != null) {
            A[t++] = head;
            head = head.next;
        }
        return A[t / 2];
    }
}
```

3、复杂度分析

- 时间复杂度：$O(N)$，其中 N 是给定链表中的结点数目。
- 空间复杂度：$O(N)$，即数组 `A` 用去的空间。

方法二：单指针法

1、思路

我们可以对方法一进行空间优化，省去数组 `A`。

我们可以对链表进行两次遍历。第一次遍历时，我们统计链表中的元素个数 `N`；第二次遍历时，我们遍历到第 `N/2` 个元素（链表的首节点为第 `0` 个元素）时，将该元素返回即可。

2、代码实现

```
class Solution {
    public ListNode middleNode(ListNode head) {
        int n = 0;
        ListNode cur = head;
        while (cur != null) {
            ++n;
            cur = cur.next;
        }
        int k = 0;
        cur = head;
        while (k < n / 2) {
            ++k;
            cur = cur.next;
        }
        return cur;
    }
}
```

3、复杂度分析

- 时间复杂度：$O(N)$，其中 N 是给定链表的结点数目。
- 空间复杂度：$O(1)$，只需要常数空间存放变量和指针。

方法三：快慢指针法

1、思路和算法

我们可以继续优化方法二，用两个指针 `slow` 与 `fast` 一起遍历链表。`slow` 一次走一步，`fast` 一次走两步。那么当 `fast` 到达链表的末尾时，`slow` 必然位于中间。

2、代码实现

```
class Solution {
    public ListNode middleNode(ListNode head) {
        ListNode slow = head, fast = head;
        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
```

```
        }
        return slow;
    }
}
```

3、复杂度分析

- 时间复杂度：$O(N)$，其中 N 是给定链表的结点数目。
- 空间复杂度：$O(1)$，只需要常数空间存放 `slow` 和 `fast` 两个指针。