获取中间节点\_LeetCode\_876\_MiddleOfTheLinkedList\_Easy

# 获取中间节点\_LeetCode\_876\_MiddleOfTheLinkedList\_Easy

## 题目介绍

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LeetCode\_876\_MiddleOfTheLinkedList\_Easy\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

\* LeetCode\_876\_MiddleOfTheLinkedList\_Easy\_获取中间节点

\* https://leetcode.com/problems/middle-of-the-linked-list/description/

\* 难度：Easy

\* DateTime:2018-10-08

\*

\* 题目介绍：

\* Given a non-empty, singly linked list with head node head,

\* return a middle node of linked list.

\* If there are two middle nodes, return the second middle node.

\*

\* Example 1:

\* Input: [1,2,3,4,5]

\* Output: Node 3 from this list (Serialization: [3,4,5])

\* The returned node has value 3. (The judge's serialization of this node is [3,4,5]).

\* Note that we returned a ListNode object ans, such that:

\* ans.val = 3, ans.next.val = 4, ans.next.next.val = 5, and ans.next.next.next = NULL.

\*

\* Example 2:

\* Input: [1,2,3,4,5,6]

\* Output: Node 4 from this list (Serialization: [4,5,6])

\* Since the list has two middle nodes with values 3 and 4, we return the second one.

\*

\* Note:

\* The number of nodes in the given list will be between 1 and 100.

## 思路分析

\* 思路分析：快慢指针的方法。

\* 两个指针slow和fast：slow一次走一步，fast一次走两步；由于fast每次走两步即fast.next.next;

\* 因此需要判断fast!=null && fast.next != null;

\* 正好，当fast==null或fast.next==null的时候，此时slow对应就是中间Node或者中间两个Node的后一个。

\* 验证：1->2->3->4->5; slow = 3时候，fast=5，再一次循环判断fast.next==null,跳出循环，返回slow；、

\* 1->2->3->4->5->6; slow = 3时候，fast=5，再一次循环判断fast!=null && fast.next==null,

\* 再一次循环，slow = 4,fast = fast.next.next = null;再循环判断时，直接退出循环，返回slow。

## Java代码

//快慢指针方法

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;

}