Problem 876: Middle of the Linked List

Problem Information

Difficulty: Easy
Acceptance Rate: 81.24%
Paid Only: No
Tags: Linked List, Two Pointers

Problem Description

Given the `head` of a singly linked list, return _the middle node of the linked list_.

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

Example 1:

![](https://assets.leetcode.com/uploads/2021/07/23/lc-midlist1.jpg)

Input: head = [1,2,3,4,5] **Output:** [3,4,5] **Explanation:** The middle node of the list is node 3.

Example 2:

![](https://assets.leetcode.com/uploads/2021/07/23/lc-midlist2.jpg)

Input: head = [1,2,3,4,5,6] **Output:** [4,5,6] **Explanation:** Since the list has two middle nodes with values 3 and 4, we return the second one.

Constraints:

* The number of nodes in the list is in the range `[1, 100]`. * `1 <= Node.val <= 100`

Code Snippets

C++:

```
/**
* Definition for singly-linked list.
* struct ListNode {
* int val;
* ListNode *next;
* ListNode() : val(0), next(nullptr) {}
* ListNode(int x) : val(x), next(nullptr) {}
* ListNode(int x, ListNode *next) : val(x), next(next) {}
* };
*/
class Solution {
public:
ListNode* middleNode(ListNode* head) {

}
};
```

Java:

```
/**
* Definition for singly-linked list.
* public class ListNode {
* int val;
* ListNode next;
* ListNode() {}
* ListNode(int val) { this.val = val; }
* ListNode(int val, ListNode next) { this.val = val; this.next = next; }
* }
*/
class Solution {
public ListNode middleNode(ListNode head) {

}
}
```

Python3:

```
# Definition for singly-linked list.
# class ListNode:
# def __init__(self, val=0, next=None):
# self.val = val
# self.next = next
class Solution:
def middleNode(self, head: Optional[ListNode]) -> Optional[ListNode]:
```