⭐🖌 234 Palindrome Linked List

2018年4月5日 17:39

Question:

Given a singly linked list, determine if it is a palindrome.

Follow up:

Could you do it in O(n) time and O(1) space?

来自 <https://leetcode.com/problems/palindrome-linked-list/description/>

请检查一个链表是否为回文链表。

进阶：

你能在 O(n) 的时间和 O(1) 的额外空间中做到吗？

Solution for Python3:

```python
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, x):
#         self.val = x
#         self.next = None
# abccba:两个指针slow和fast，两倍关系
# 当fast走到最后，slow刚好到一半
# 然后slow一边往前一边往后同时走并比较
class Solution1:
    def isPalindrome(self, head):
        """
        :type head: ListNode
        :rtype: bool
        """
        dummy = None
        slow = fast = head
        while fast and fast.next:
            fast = fast.next.next
            dummy, dummy.next, slow = slow, dummy, slow.next
        if fast:
            slow = slow.next
        while dummy and dummy.val == slow.val:
            slow = slow.next
            dummy = dummy.next
       return not dummy

```

```python
class Solution2:
    def isPalindrome(self, head):
        """
        :type head: ListNode
        :rtype: bool
        """
        rev = None
        fast = head
        while fast and fast.next:
           fast = fast.next.next
           rev, rev.next, head = head, rev, head.next
        tail = head.next if fast else head
        isPali = True
        while rev:
           isPali = isPali and rev.val == tail.val
           head, head.next, rev = rev, head, rev.next
           tail = tail.next
        return isPaliS
```

Solution for C++:

```cpp
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    bool isPalindrome(ListNode* head) {
        ListNode *rev = NULL, *pre = NULL;
        ListNode *slow = head,  *fast = head;
        while (fast && fast->next) {
            fast = fast->next->next;
            rev = slow;
            slow = slow->next;
            rev->next = pre;
            pre = rev;

        }
        if (fast) {
            slow = slow->next;
        }

        while (rev && (rev->val == slow->val)) {
            slow = slow->next;
            rev = rev->next;
        }
        return !rev;
    }
};
```