Author: Xiaoqiang Yan
Date: 08/06/2018

LeetCode 234 Palindrome Linked List

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

Example 1:

```
Input: 1->2
Output: false
```

Example 2:

```
Input: 1->2->2->1
Output: true
```

Follow up:

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

Method

If there's no space limit, we get a new linked list by reversing the original linked list, and compare each nodes from head to tail.

When we have space limit, we need to use the original list to save reverse list. So we can divide the original list two parts: keep the front half unchanged and latter half reversed.

We need to get the middle nodes firstly.

Code

```
struct ListNode {
        int val;
        ListNode* next;
        ListNode(int x): val(x), next(NULL) {}
};

bool isPalindrome( ListNode* head) {
        if(head == NULL || head->next == NULL) return true;
        //first we need to get the middle node of the list
        ListNode  *tail = head, *mid = head;
        while(tail && tail->next) {  // first check tail, then check tail->next
                tail = tail->next->next;
                mid = mid->next;
        }
        //reverse the latter half list
        ListNode *pre = mid, *cur = mid->next;
        pre->next = NULL;
        while(cur) {
                ListNode* temp = cur->next;
                cur->next = pre;
                pre = cur;
                cur = temp;
        }
        ListNode *p1 = head, *p2 = pre;
        while(p1!=p2 && p2!=NULL){
                if(p1->val != p2->val)
                        return false;
                else{
                        p1 = p1->next;
                        p2 = p2->next;
                }
        }
        return true;
}
```

Note

1. when we have to save space, try to use the original list, like divide into two parts;
2. pay more attention to the condition (when use && or ||, which one should be checked firstly)
3. this question should find the middle pointer firstly, if not, we can’t solve it in space O(1).

Summary

Two pointers model