LeetCode 234

Description

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?

Thought

The first thought is to create a new linkedList that's reverse of the given one and compare the two, but it will take extra space. A better idea is to find the middle point and reversing the second half and compare with the first half.

To find the middle point, we can use the two points(fast & slow) method

Solution

```
public boolean isPalindrome(ListNode head) {
    ListNode fast = head, slow = head;
    while (fast != null && fast.next != null) {
        fast = fast.next.next;
        slow = slow.next;
    }
    if (fast != null) { // odd nodes: let right half smaller
        slow = slow.next;
    }
    slow = reverse(slow);
    fast = head;

    while (slow != null) {
        if (fast.val != slow.val) {
            return false;
        }
        fast = fast.next;
        slow = slow.next;
    }
    return true;
}

public ListNode reverse(ListNode head) {
    ListNode prev = null;
    while (head != null) {
        ListNode next = head.next;
        head.next = prev;
```

```
            prev = head;
            head = next;
        }
        return prev;
    }
```

Takeaways

- Tow pointers to equally divide a list
- Three Pointer … multiple pointers
- Reverse List is often used