Java Palindrome Linked List

Which response do you prefer? Responses may take a moment to load.

Response 1

Here is the Java code to determine if a singly linked list is a palindrome:

```java
class ListNode {
    int val;
    ListNode next;

    ListNode(int val) {
        this.val = val;
        this.next = null;
    }
}

public class Solution {

    public boolean isPalindrome(ListNode head) {
        if (head == null || head.next == null) {
            return true;
        }

        // Find the middle of the linked list
        ListNode slow = head, fast = head;
        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
        }

        // Reverse the second half of the linked list
        ListNode reversed = reverseList(slow);

        // Compare the two halves
        ListNode pointer1 = head, pointer2 = reversed;
        while (pointer2 != null) {
            if (pointer1.val != pointer2.val) {
                return false;
            }
            pointer1 = pointer1.next;
            pointer2 = pointer2.next;
        }

        return true;
    }

    private ListNode reverseList(ListNode head) {
        ListNode prev = null, next = null;
        while (head != null) {
            next = head.next;
            head.next = prev;
            prev = head;
            head = next;
        }
        return prev;
    }

    public static void main(String[] args) {
        // Test case 1: [1,2,2,1]
        ListNode head1 = new ListNode(1);
        head1.next = new ListNode(2);
        head1.next.next = new ListNode(2);
```

```
        head1.next.next.next = new ListNode(1);
        Solution solution = new Solution();
        System.out.println(solution.isPalindrome(head1)); // Output: true

        // Test case 2: [1,2]
        ListNode head2 = new ListNode(1);
        head2.next = new ListNode(2);
        System.out.println(solution.isPalindrome(head2)); // Output: false
    }
}
```

Explanation:

1. **Find the Middle**: Using a slow and fast pointer technique, we find the middle of the linked list.
2. **Reverse the Second Half**: We reverse the second half of the list.
3. **Compare the Two Halves**: Finally, we compare the values from both halves. If they match, the list is a palindrome.

I prefer this response

Response 2

Here is the Java code to determine if a singly linked list is a palindrome:

```java
class ListNode {
    int val;
    ListNode next;

    ListNode(int val) {
        this.val = val;
        this.next = null;
    }
}

public class Solution {
    public boolean isPalindrome(ListNode head) {
        if (head == null || head.next == null) {
            return true;
        }

        // Step 1: Find the middle of the linked list
        ListNode slow = head, fast = head;
        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
        }

        // Step 2: Reverse the second half of the linked list
        ListNode prev = null, curr = slow;
        while (curr != null) {
            ListNode nextTemp = curr.next;
            curr.next = prev;
            prev = curr;
            curr = nextTemp;
        }

        // Step 3: Compare the two halves
        ListNode firstHalf = head, secondHalf = prev;
        while (secondHalf != null) {
            if (firstHalf.val != secondHalf.val) {
                return false;
            }
            firstHalf = firstHalf.next;
            secondHalf = secondHalf.next;
```

```
            }

            return true;
        }
    }
```

Explanation:

1. **Step 1**: Use the slow and fast pointer approach to find the middle of the linked list. When `fast` pointer reaches the end, `slow` pointer will be at the middle.
2. **Step 2**: Reverse the second half of the linked list starting from the middle.
3. **Step 3**: Compare the values of nodes from the start of the list and the reversed second half. If all values match, the list is a palindrome; otherwise, it is not.

Example:

- Input: `[1,2,2,1]`
  - Output: `true`
- Input: `[1,2]`
  - Output: `false`

I prefer this response