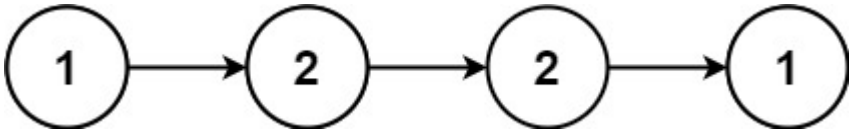
204. Palindrome Linked List

Given the `head` of a singly linked list, return `true` *if it is a*

palindrome

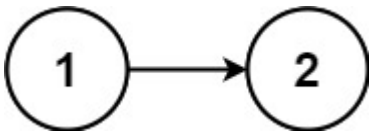
or `false` *otherwise*.

Example 1:



Input: head = [1,2,2,1]
Output: true

Example 2:



Input: head = [1,2]
Output: false

Constraints:

- The number of nodes in the list is in the range `[1, 105]`.
- `0 ≤ Node.val ≤ 9`

Follow up: Could you do it in `O(n)` time and `O(1)` space?

```
/**
 * 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 {
```

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

        ListNode slow = head;
        ListNode fast = head;
        ListNode prev = null;
        // Find the middle of the linked list and reverse the first half
        while (fast ≠ null && fast.next ≠ null) {
            fast = fast.next.next;
            ListNode temp = slow.next;
            slow.next = prev;
            prev = slow;
            slow = temp;
        }

        if (fast ≠ null)
            slow = slow.next;
          while (prev ≠ null && slow ≠ null) {
            if (prev.val ≠ slow.val)
                return false;
            prev = prev.next;
            slow = slow.next;
        }
        return true;
    }
}
```