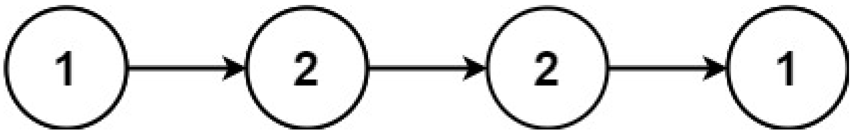
234. Palindrome Linked List

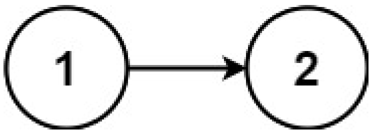
Given the `head` of a singly linked list, return `true` if it is a palindrome.

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, 10<sup>5</sup>]`.
- `0 <= Node.val <= 9`

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

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
class Solution:
    def isPalindrome(self, head: Optional[ListNode]) -> bool:
        if head is None or head.next is None:
            return head
        curr = head
        mid = self.findMid(head)

        tempHead = mid.next
        mid.next  = None
        newHead = self.reverseLL(tempHead)

        while curr is not None and newHead is not None:
            if curr.val!=newHead.val:
                return False
            curr = curr.next
            newHead = newHead.next
        return True

    def reverseLL(self,head):
        if head is None or head.next is None:
            return head

        curr = head
        prev = None

        while curr!=None:
            nxt = curr.next
            curr.next = prev
            prev = curr
            curr  = nxt
        return prev

    def findMid(self,head):
        if head is None or head.next is None:
            return head

        slow = head
        fast = head

        while fast.next is not None and fast.next.next is not None:
            slow = slow.next
            fast = fast.next.next
        return slow
```