

Palindrome linked list

class ListNode:

```
def __init__(self, val=0, next=None):  
    self.val = val  
    self.next = next
```

def isPalindrome(head: ListNode) -> bool:

```
    if not head or not head.next:  
        return True
```

Step 1: Find the middle of the linked list

```
slow, fast = head, head  
while fast and fast.next:  
    slow = slow.next  
    fast = fast.next.next
```

Step 2: Reverse the second half

```
prev = None  
while slow:  
    next_node = slow.next  
    slow.next = prev  
    prev = slow  
    slow = next_node
```

Step 3: Compare both halves

```
left, right = head, prev  
while right: # Compare until the end of the reversed second half  
    if left.val != right.val:  
        return False  
    left = left.next  
    right = right.next  
return True
```

Explanation:

Finding the Middle: The slow pointer moves one step at a time, while the fast pointer moves two steps. When fast reaches the end, slow will be at the middle.

Reversing the Second Half:

We reverse the linked list starting from slow until the end.

Comparison: We compare the values from the start and the reversed second half.