연결 리스트가 팰린드롬 구조인지 판별하라

Input: 1->2
Output: false

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, val=0, next=None):
#         self.val = val
#         self.next = next
```

1.리스트로 변환

```
class Solution:
    def isPalindrome(self, head: ListNode) -> bool:
        nums = []
        node = head
        while node:
            nums.append(node.val)
            node = node.next
        while len(nums) > 1:
            if nums.pop(0) != nums.pop():
                return False
        return True
```

2.데크 활용

```
class Solution:
    def isPalindrome(self, head: ListNode) -> bool:
        nums = collections.deque()
        node = head
        while node:
            nums.append(node.val)
            node = node.next
        while len(nums) > 1:
            if nums.popleft() != nums.pop():
                return False
        return True
```

- 리스트(SLL)의 경우 pop(0)이 O(n)임,
- 데크의(DLL) 경우 popleft()가 O(1)

3.런너 활용

```
class Solution:
    def isPalindrome(self, head: ListNode) -> bool:
        rev = None
        slow = fast = head
        while fast and fast.next:
            fast = fast.next.next
            rev, rev.next, slow = slow, rev, slow.next
        if fast:
```

```
    slow = slow.next
while rev and rev.val == slow.val:
    slow, rev = slow.next, rev.next
return not rev
```

- 느린, 빠른 런너를 이용해서 중간 값부터 팰린드롬 여부 검사
- 홀수, 짝수를 다르게 처리해줌