Palindrome Linked list

BRUTE FORCE :

Traverse the linked list and store it in a string or array and check for plaindrome.

- Time Complexity : O(N)+O(N)

One for storing into string and other O(N) for checking for palindrome.

- Space Complexity : O(N) for string

Optimal Approach :

Find mid of the linked list then reverse mid+1,n part of linked list. To find mid elem there can be two condition when list is odd and when it is even. So for even we want the fast elem of list to satisfy this we must make sure that fast pointer is the last or second last elem.

Place a dummy pointer at linked list head and other pointer at mid+1 compare the elements wherever the cond of not equal does not satisy return false

```
class Solution {
public:
    void reverse(ListNode* end,ListNode* curr)
    {
        if(curr==NULL || curr->next==NULL)
            return;
        ListNode* prev=NULL;
        while(curr!=NULL)
        {
            ListNode* next=curr->next;
            curr->next=prev;
            prev=curr;
            curr=next;
        }
        end->next=prev;
    }
    bool isPalindrome(ListNode* head) {
        if(head==NULL||head->next==NULL)
            return true;
        ListNode* slow=head;
```

```
        ListNode* fast=head;
        while(fast->next!=NULL && fast->next->next!=NULL)
        {
            slow=slow->next;
            fast=fast->next->next;
        }
        reverse(slow,slow->next);
        ListNode* temp=head;
        while(temp!=NULL)
        {
            cout<<temp->val<<" ";
            temp=temp->next;
        }
        ListNode* d=head;
        slow=slow->next;
        while(slow!=NULL)
        {
            if(d->val!=slow->val)
            {
                return false;
            }
            d=d->next;
            slow=slow->next;
        }
        return true;
    }
};
```

- Time Complexity : O(n/2)+O(n/2) + O(n/2) finding mid , then one for reversing and last one for comparison.
- Space Complexity : O(1)