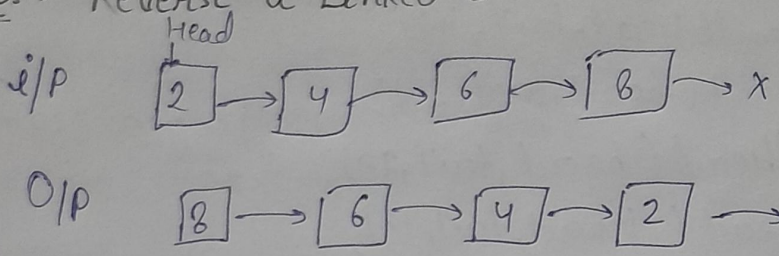
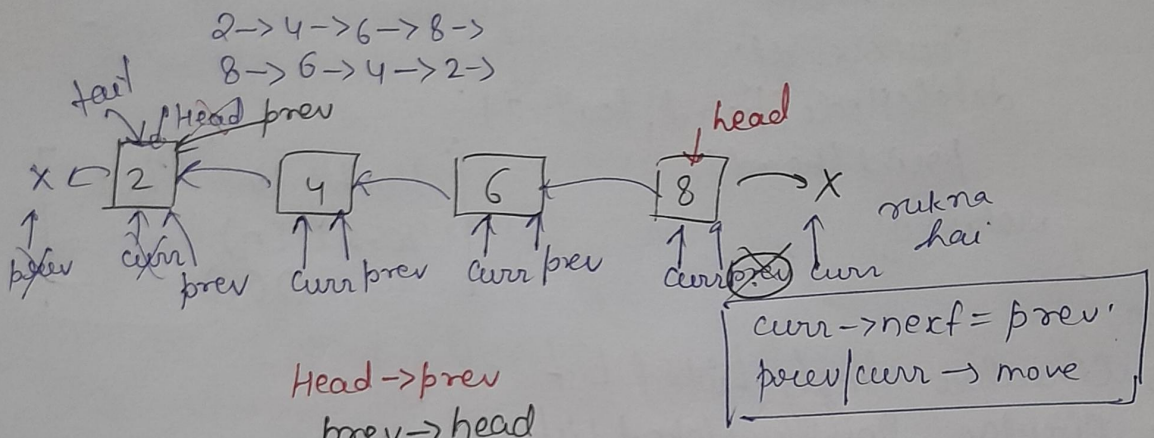
Lecture- Linked-List Class-3 Date-05/11/2023

Ques→ Reverse a Linked List

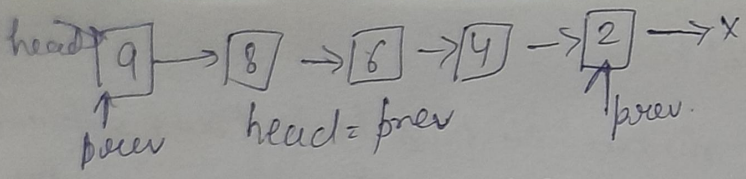




Track rakhne keliye hume ek curr pointer lagana hoga

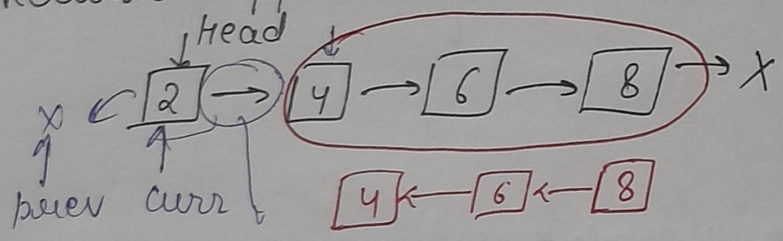
```
Node *prev=NULL
Node *curr = head;
While(curr!=NULL){
    Node * ~~head~~=nextNode=curr->next
        curr->next = prev
            prev = curr
            curr = nextNode
}
    head = prev;
```

```
class Solution {
public:
    ListNode * reverseList( ListNode * head) {
        ListNode * prev=NULL;
        ListNode curr = head;
    While (curr!=NULL){
        ListNode * nextNode = curr->next;
            curr->next = prev;
                prev = curr;
                curr = nextNode;
```



Reverse LL ka head prev pe hota hai

* Recursion approach :-



ye link todne se pehle

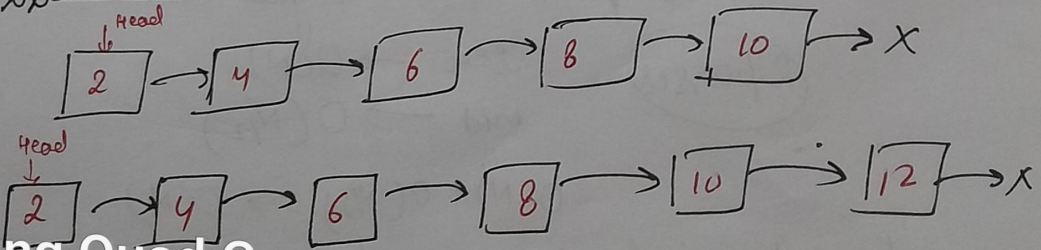
~~prev~~ →
Node * prev = NULL
Node * curr = head
Node * nextNode = curr->ne
curr->next = prev

base case →
if (curr == NULL) {
return prev;

```
ListNode* reverseUsingRecursion(ListNode* prev,
                                ListNode* curr) {
    //base case
    if(curr==NULL){
        return prev;
    }
    ListNode* nextNode = curr->next;
    curr->next = prev;
    ListNode* recursionKaAns = reverseUsingRecursion(curr, nextNode);
    return recursionKaAns;
}
ListNode* reverseList(ListNode *head) {
        ListNode* prev = NULL;
        ListNode* curr = head;
    return reverseListRecursion(prev, curr);
    }
};
```

prev = curr
curr = nextNode

* Middle of a Linked List



Algo n→?

$(\frac{n}{2}+1)$ → traverse point kardo.

Two baar passes lag rhi hai ek pass lag rha hai $(\frac{n}{2}+1)$ point karne ke liye and fir ek pass lag rha hai $(\frac{n}{2}+1)$ point karne ke liye.

```
class Solution {
public:
    int getLength(ListNode* head){
        int len=0;
        ListNode* temp=head;
        while(temp!=NULL){
            len++;
            temp=temp->next;
        }
        return len;
    }
ListNode* middleNode(ListNode* head){
    int n= getLength(head);
    int position= n/2+1;
    int curPos=1;
    ListNode* temp=head;
    while(curPos!=position){
        curPos++;
        temp=temp->next;
    }
    return temp;
}
};
```

kya toh fir aise kar sakte hai

```
while(position!=1){
    position--;
    temp=temp->next;
}
return temp;
}
```

is code me thodi si galti hai.

2 passes

len → O(N)

mid → O(N/2)

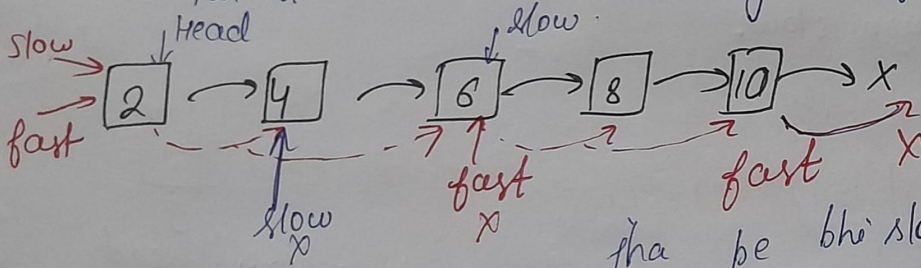
O(N) + O(N/2) ⇒ O(N)

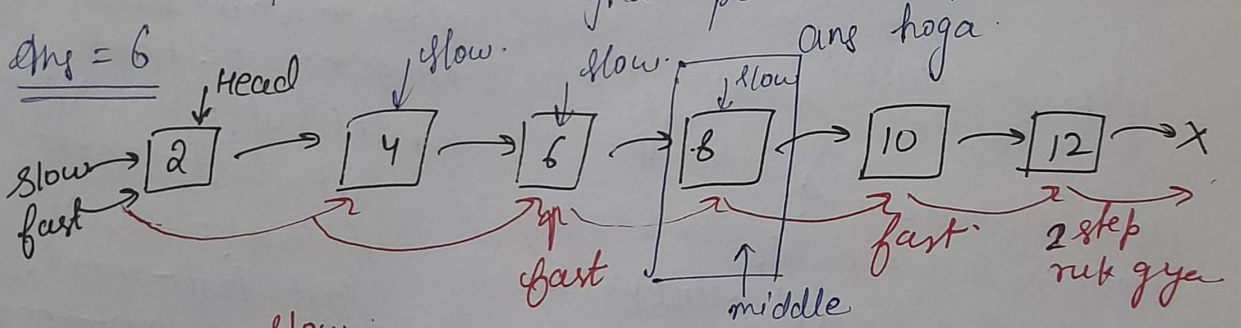
kya ise single pass me kar sakte hai

Single Pass

1) Slow & fast approach
or
Two pointer approach
or
Tortoise Algorithm

Slow wala pointer ek step se aage badhega and fast wala 2 step se aage badhega lekin condition ye hai ki slow wala pointer tabhi aage badhega jab fast wala pointer 2 step aage badh jayega.
Agar fast wala pointer 2 step aage nhi badh paa rha toh slow wala nhi bhi badhega aage ek step.

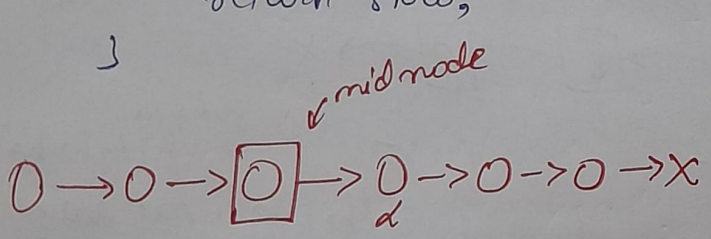


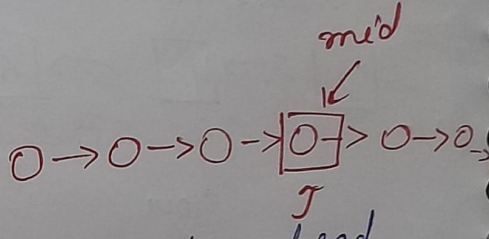


Slow

```
class Solution {
public:
    int getLength(ListNode* head) {
        int len=0;
        ListNode* temp=head;
        while(temp!=NULL) {
            len++;
            temp=temp->next;
        }
        return len;
    }
```

```
ListNode * middleNode(ListNode * head){
    ListNode * slow = head;
    ListNode * fast = head;
    while(fast!=NULL){
        fast = fast->next;
        if(fast!=NULL){
            fast = fast->next;
            // main yha pe keh rakha hu k fast ne 2 step
            //                  chal liya hai
            //ab slow ko bhi ek ek step chlwa do.
            slow = slow->next;
        }
    }
    return slow;
}
```






```
if(head==NULL){
    return;
}
Node * slow = head;
Node * fast = head;
while(fast->next!=NULL){
    fast = fast->next;
    if(fast->next!=NULL){
        fast = fast->next
        slow = slow->next
    }
}
return slow;
}
```

assume LL is empty.

```
Node * slow = head
Node * fast = head
while(fast!=NULL){
    fast = fast->next;
    if(fast!=NULL){
        fast = fast->next;
        slow = slow->next;
    }
}
return slow;
}
```

(3) Palindrome

1 → 2 → 1 → ∝

1 → 5 → 5 → 1 → ∝

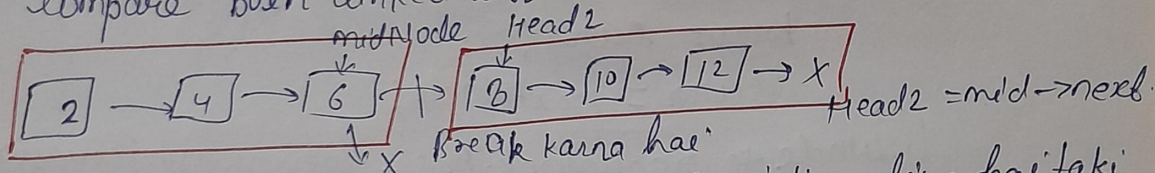
Odd: head → 1 → 2 → 3 → 2 → 1 → X (mid = 3)

Even: 1 → 2 → 3 → 1 → 2 → 3 → X

(A) Break into 2 halfes

(B) Second list → reverse

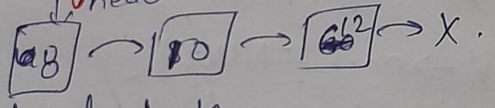
(C) compare both linked list.



(A)
1. midNode find
2. head2 set
3. midNode → next = NULL

is node ko midNode isliye liya hai taki ye wali link break ho jaaye.

(B) Reverse second half :-



Reverse ke baad. head2

12 → 10 → 8 → X

(C) Phele linked list me

12 → 4 → 6

12 → 10 → 8

comparison → match (head = head → next, head2 = head2 → next.)

↳ Not match

can we only check while (S2 != NULL)

Break mid → O(N)

Reverse → O(N)

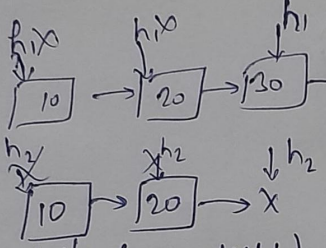
Comparison → O(N/2)

odd ke case me S1 only wali list badi hogi humesa S2 wala list null hoga phele

```
class Solution {
public:
ListNode* reverseUsingRecursion(ListNode *prev, ListNode* curr) {
    //base case
    if(curr==NULL){
        return prev;
    }
ListNode* nextNode = curr->next;
    curr->next=prev;
     prev= curr;
     curr = nextNode;
  ListNode* recursionkaAns = reverseUsingRecursion(prev, cu
  return recursionkaAns;
ListNode* middleNode(ListNode* head){
     ListNode* slow=head;
     ListNode* fast=head;
    while(fast->next!=NULL){
        fast=fast->next;
        if(fast->next!=NULL){
            fast= fast->next;
            slow=slow->next;
        }
    }
    return slow;
}
bool compareList(ListNode* head1, ListNode* head2){
    while(head2!=NULL){
        if(head1->val!=head2->val){
            return false;
        }
        else{
            head1 = head1->next;
            head2 = head2->next;
    }
}
```

```
return true;
}
bool isPalindrome(ListNode* head){
    // break into two halves
ListNode* midNode = middleNode(head);
ListNode* head2 = midNode->next;
    midNode->next = NULL;
// reverse second half
ListNode* prev = NULL;
ListNode* curr = head2;
head2 = reverseUsingRecursion(prev, curr);
// compare both linked list
    bool ans = compareList(head, head2);
    return ans;
}
}
```

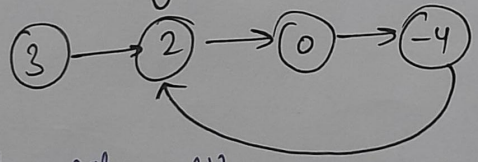
(A) Break → mid → O(N)
(B) Reverse → O(N)
(C) compare → O(N)
↓
Total → O(N)



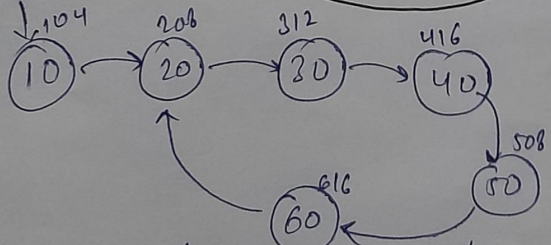
Odd list me mid wala apne aap se hi compare hota hai toh usko check karne ki kine need nhi hai

* can we only check while (s2 != NULL) Hmm kar sakte hai → out of loop → return true.

Ques → Check cycle in a linked list



I/P → head = [3, 2, 0, -4]



Jab koi address repeat ka jaye traversal karte h toh loop present hai

Ek map bna liya map ek table

address	status
104	True
208	True
312	T
416	T
508	T
616	T

loop present

address repeat → traverse ↓ loop

```
class solution {
public:
    bool hasCycle(ListNode *head) {
        map<ListNode*, bool> table;
        ListNode* temp = head;
        while(temp != NULL){
            if(table[temp] == false){
                table[temp] = true;
            }else{
                return true;
            }
            return false temp = temp->next;
        }
        return false;
    }
};
```