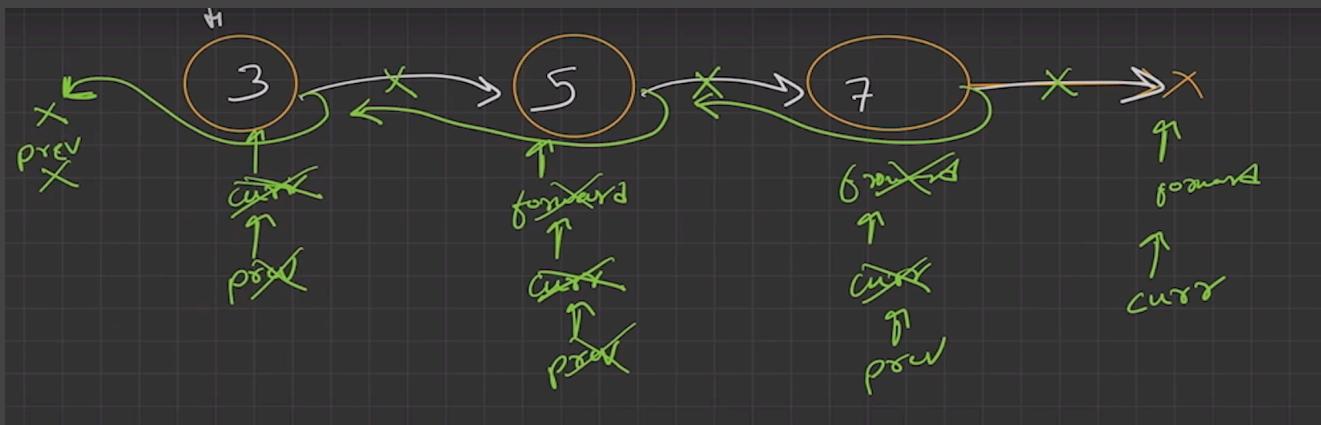
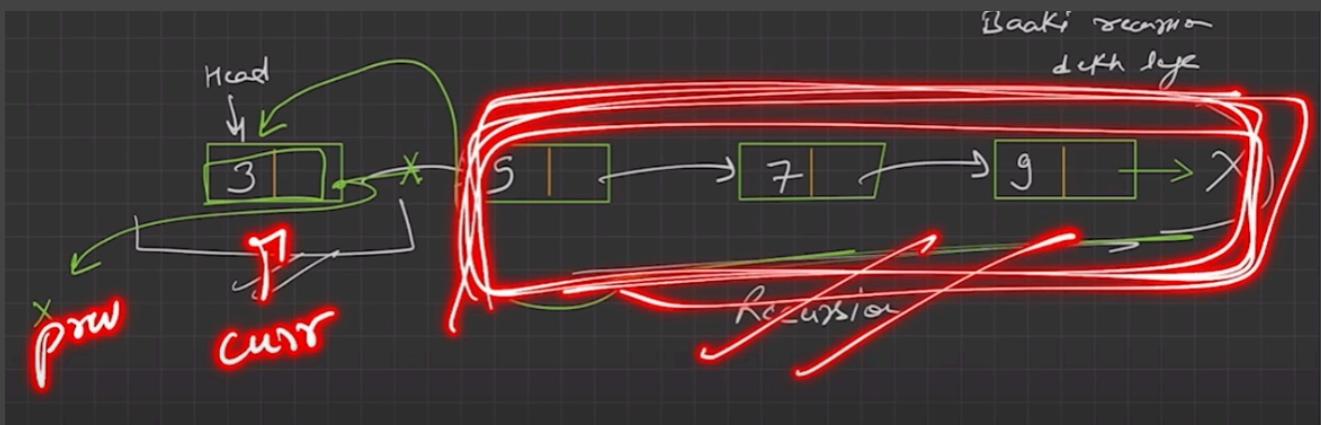


# Linked-List

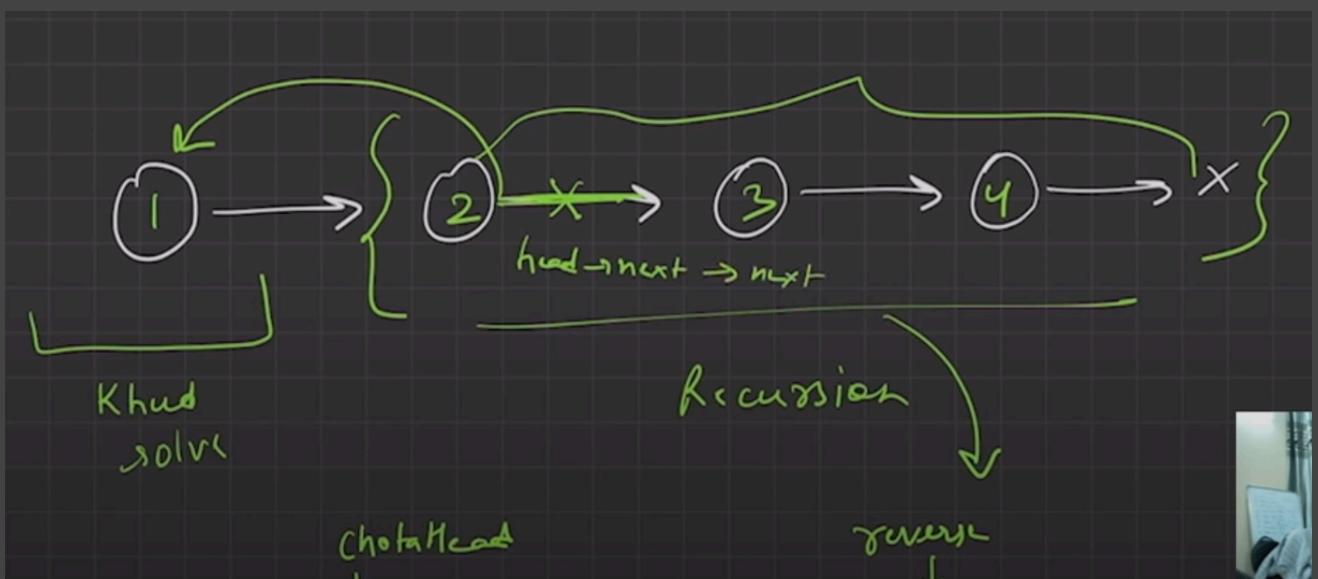
Reverse A Singly Linked List Approach-I:



Reverse A Singly Linked List Approach-II:

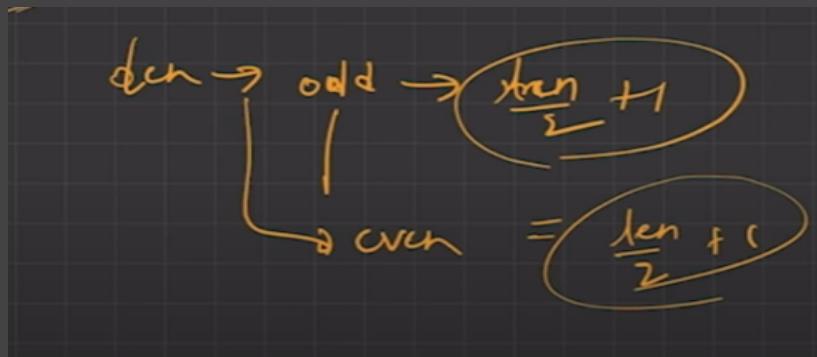
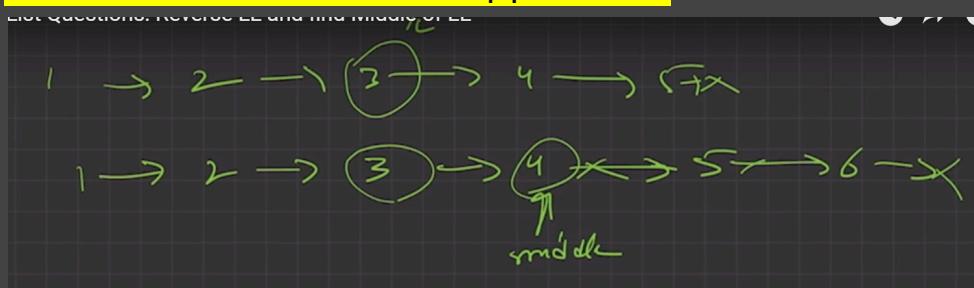


Reverse A Singly Linked List Approach-III:

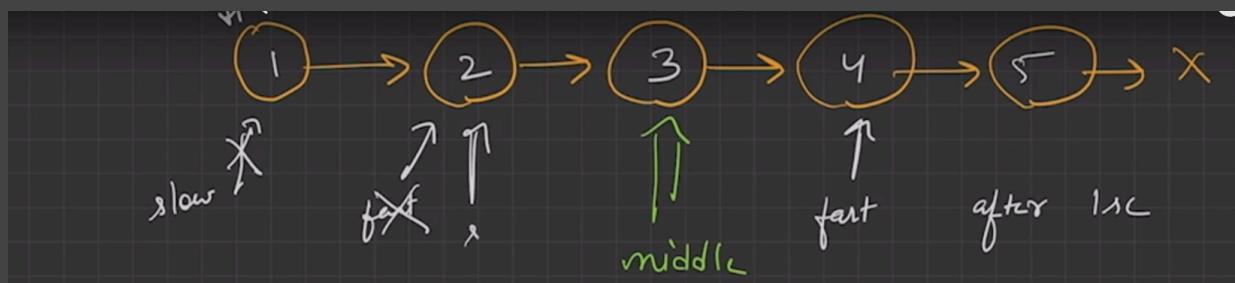


## Reverse A Doubly Linked List

### Find Middle In A Linked List Approach-I:



### Find Middle In A Linked List Approach-II:



① if  $\rightarrow$  empty list  $\rightarrow$  return `NULL`



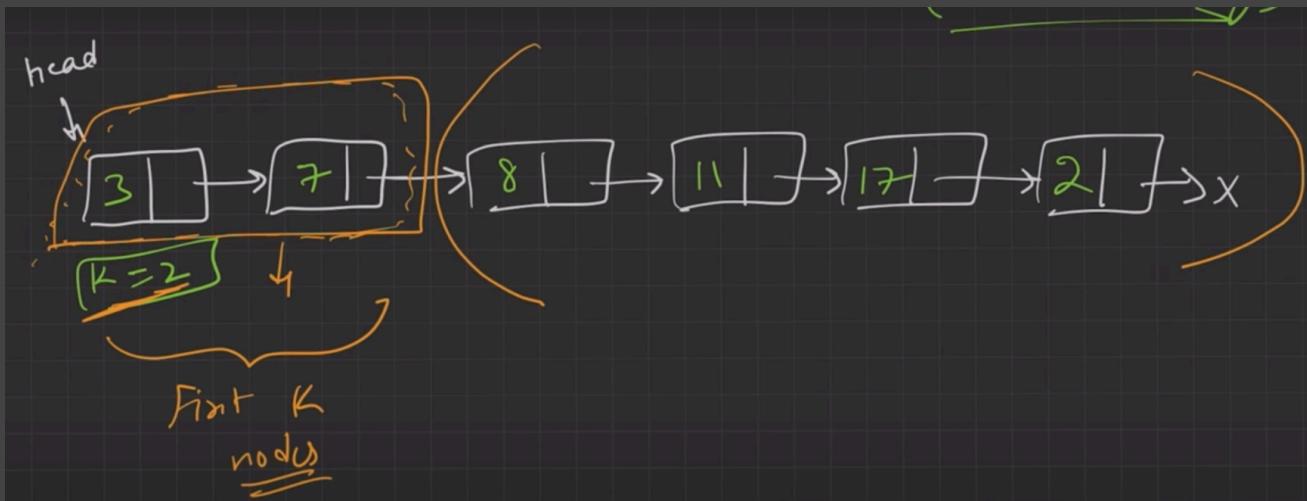
② 1 node  $\rightarrow$  head

③ 2 nodes  $\rightarrow$  head  $\rightarrow$  next

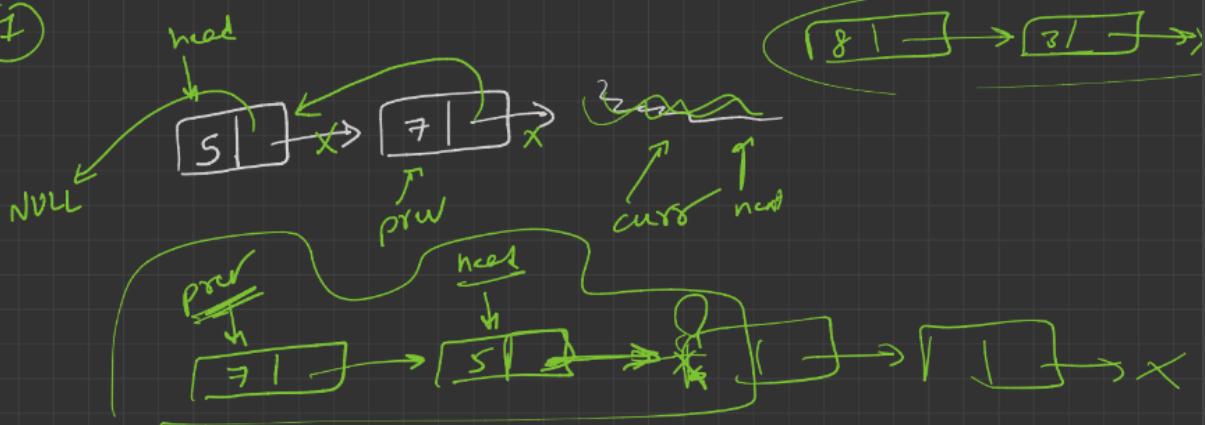
④ Algo



## Reverse Linked List In 'K' groups:



(i)



(ii)

$head \rightarrow next \rightarrow$  Recursion call

(iii)

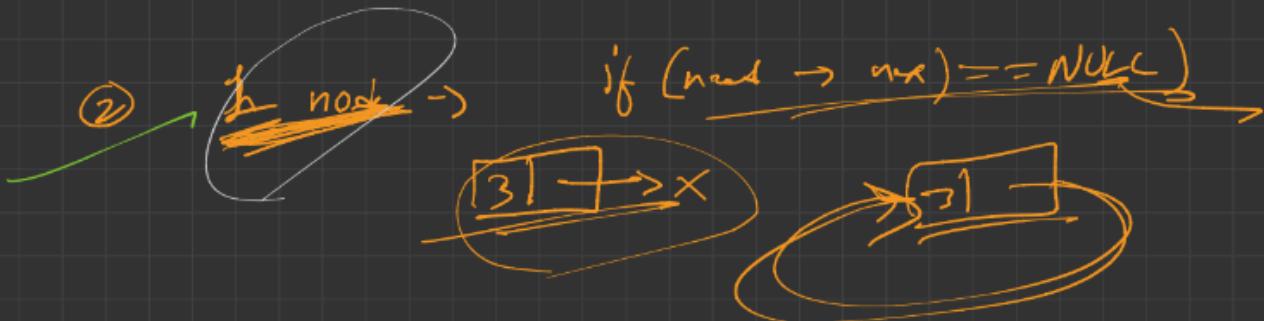
return  $prev$

## Check Whether A Linked List is Circular or Not Approach-I:

Approach 1 :-



(1) Empty List  $\rightarrow$  if (head == NULL)  
return True



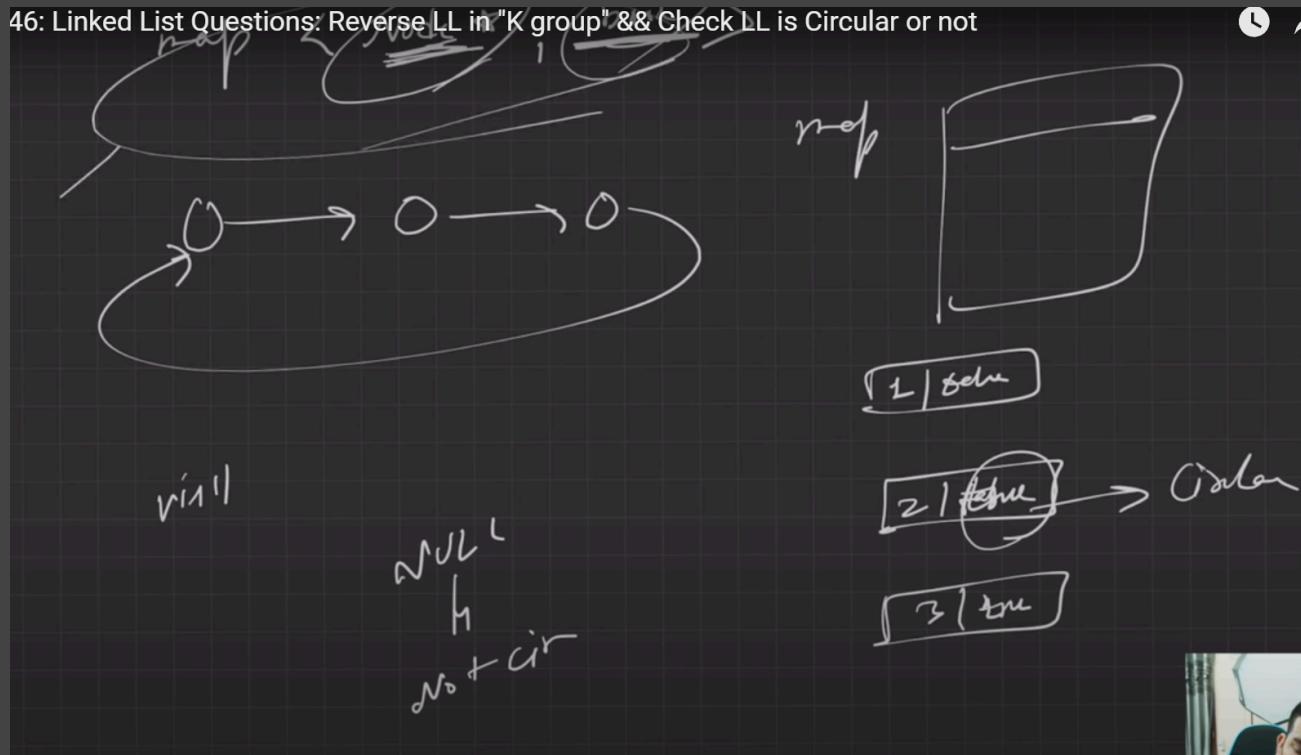
(2) Non-empty List  $\rightarrow$  if (head->next == head)  
return True



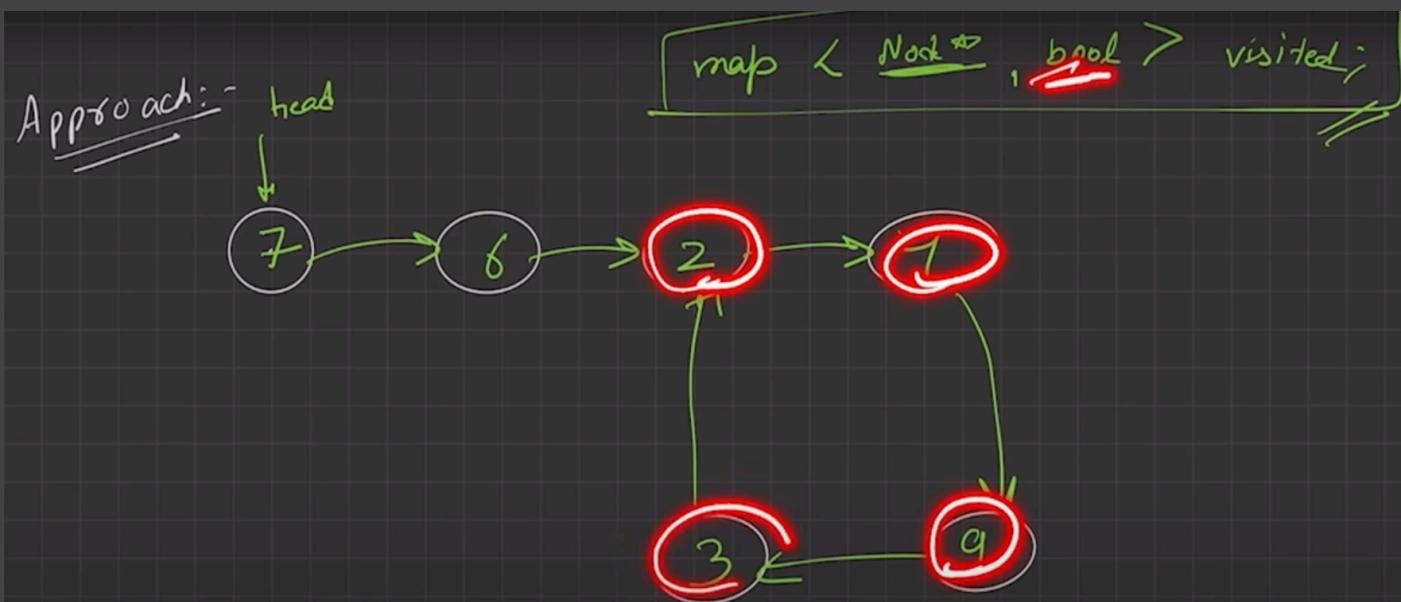
L  
 $\rightarrow$  n

## Check Whether A Linked List is Circular or Not Approach-II:

46: Linked List Questions: Reverse LL in "K group" && Check LL is Circular or not

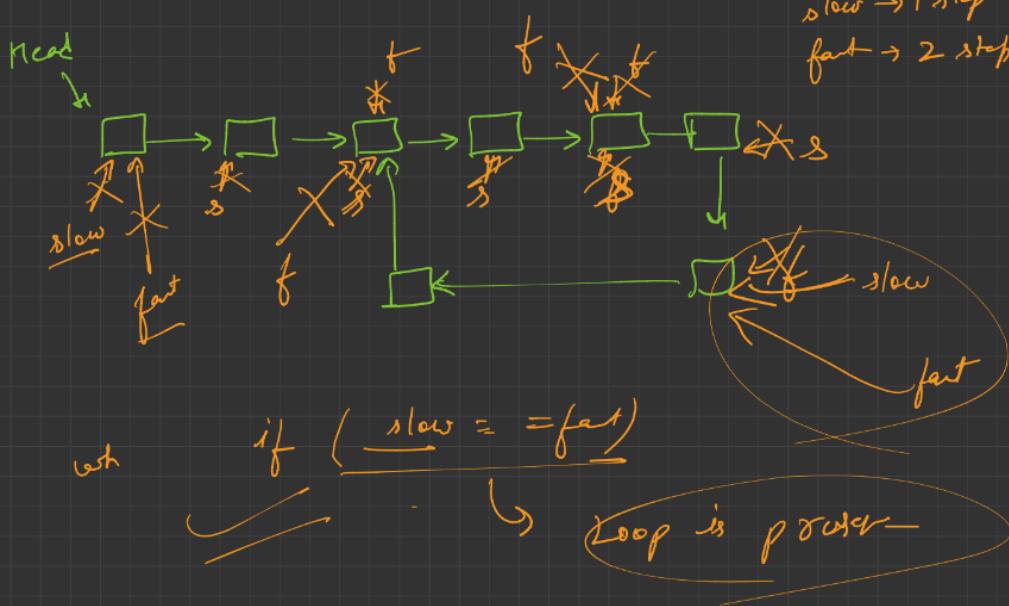


## Detect Whether A Linked List Forms a Loop or Not:

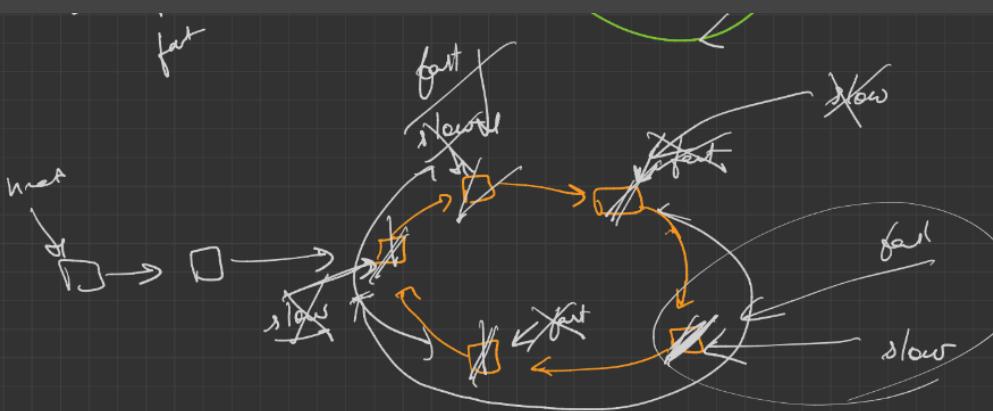


## Floyd's Cycle Detection:

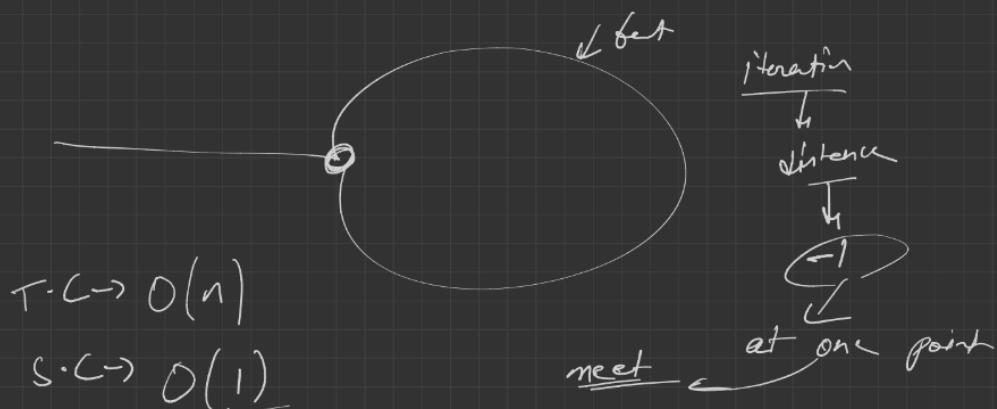
① Floyd's cycle detection algo.



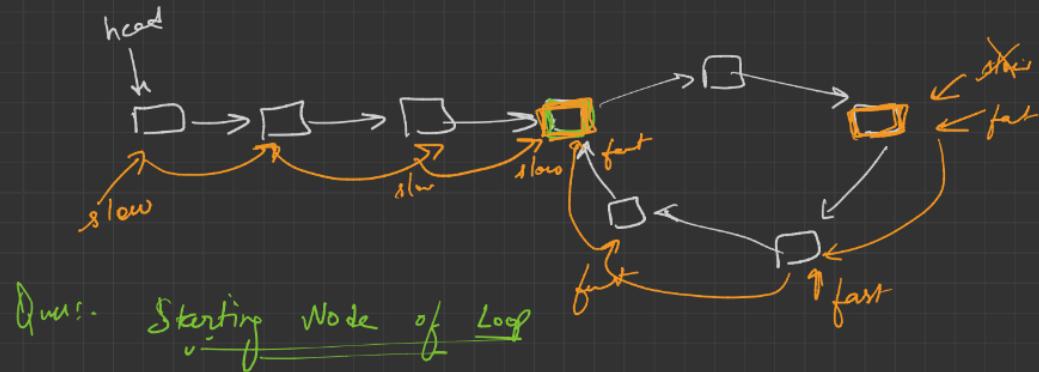
②  $\text{fast} = \text{NULL}$   $\rightarrow$  No loop



Distance: 4, 3, 2, 1



Find The Starting Node Of The Loop:



Approach:-

I  $\rightarrow$  FCD Algo  $\rightarrow$  Point of Intersection

II  $\rightarrow$   $slow = head$   
 $slow, fast \rightarrow$  same pace  
 / / /

when ( $slow == fast$ )

$\hookrightarrow$  [start] point of Loop

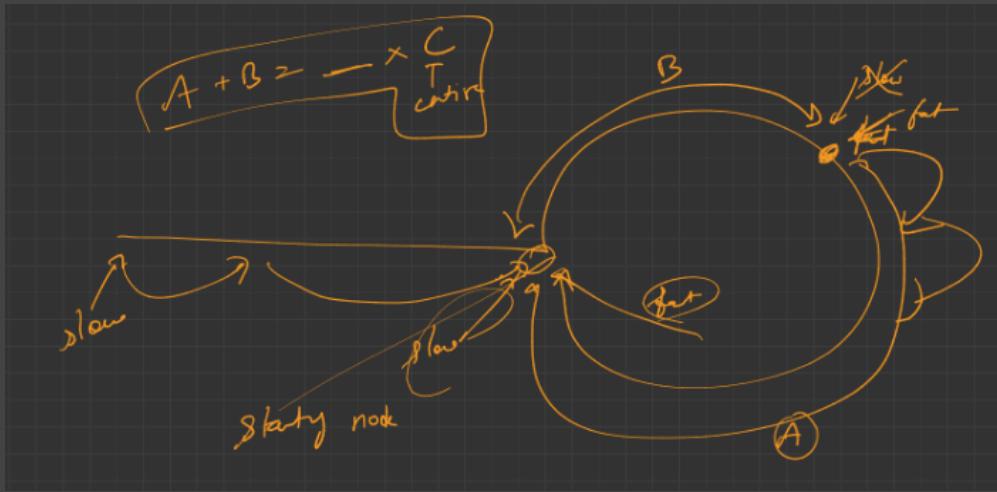
Distance by  $= 2 \star$  Distance by slow pointed  
 fast pointer

$$(A + x * C + B) = 2 \star (A + y * C + B)$$

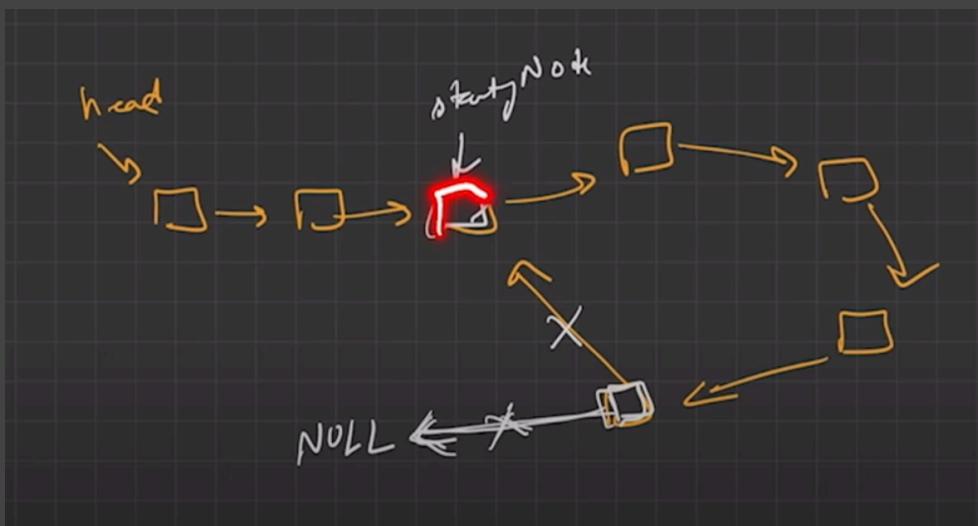
$$A + nC + B = 2A + 2yC + 2B$$

$$C(n - 2y) = A + B$$

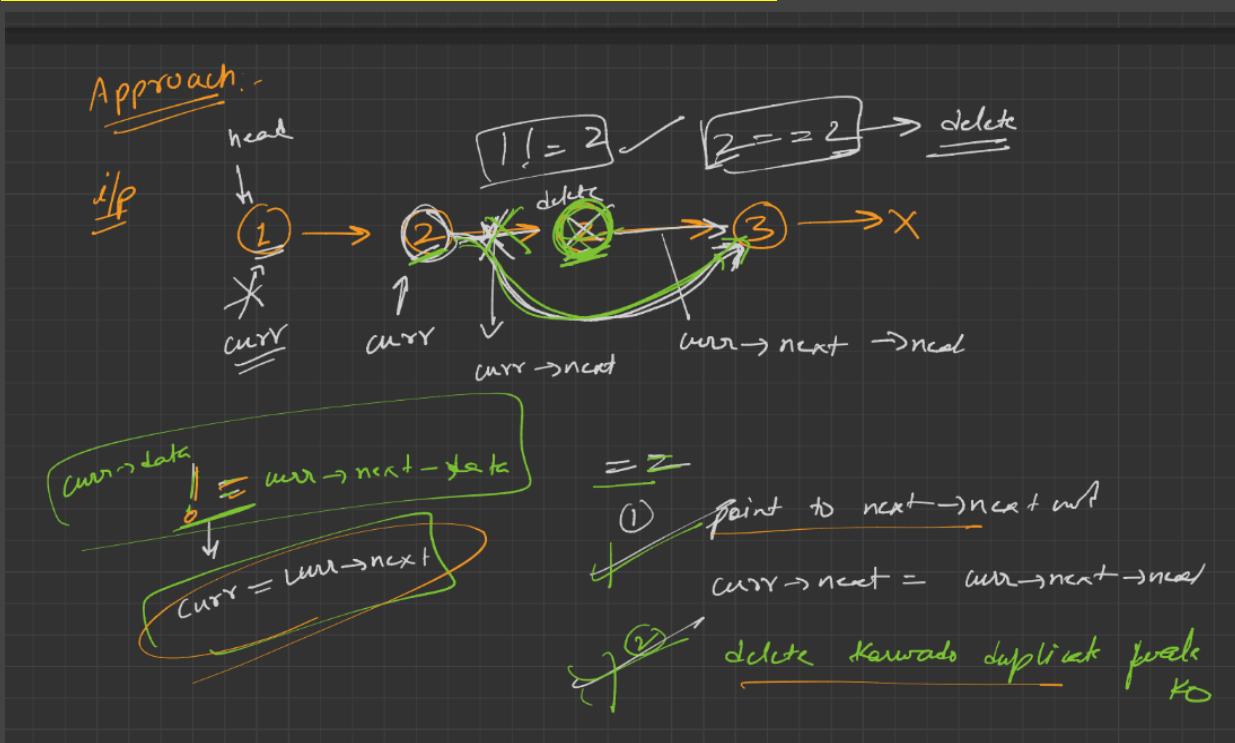
$$\frac{A + B}{K} = \frac{K \text{ times } C}{K}$$



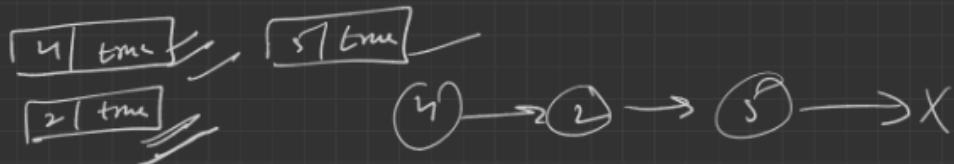
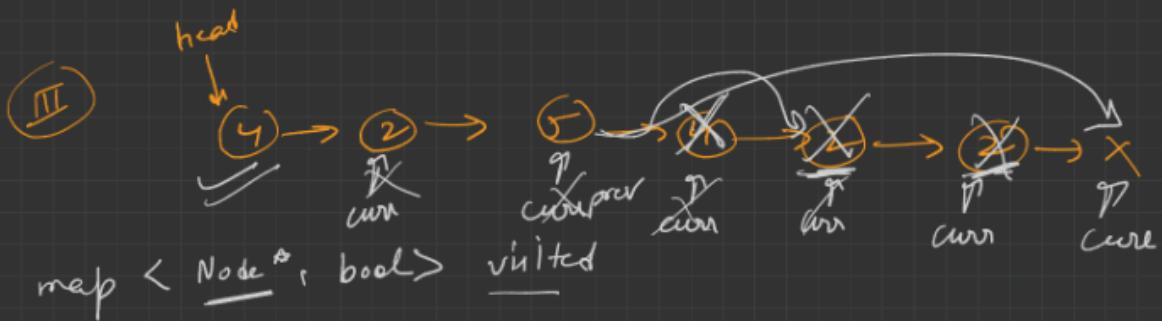
### Remove The Loop:



### Remove Duplicates From A Sorted Linked List:

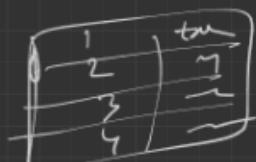


## Remove Duplicates From A Unsorted Linked List:



$T \cdot C \rightarrow O(n)$   
 $S \cdot C \rightarrow O(n)$

① → ② → ③ → ④ → X



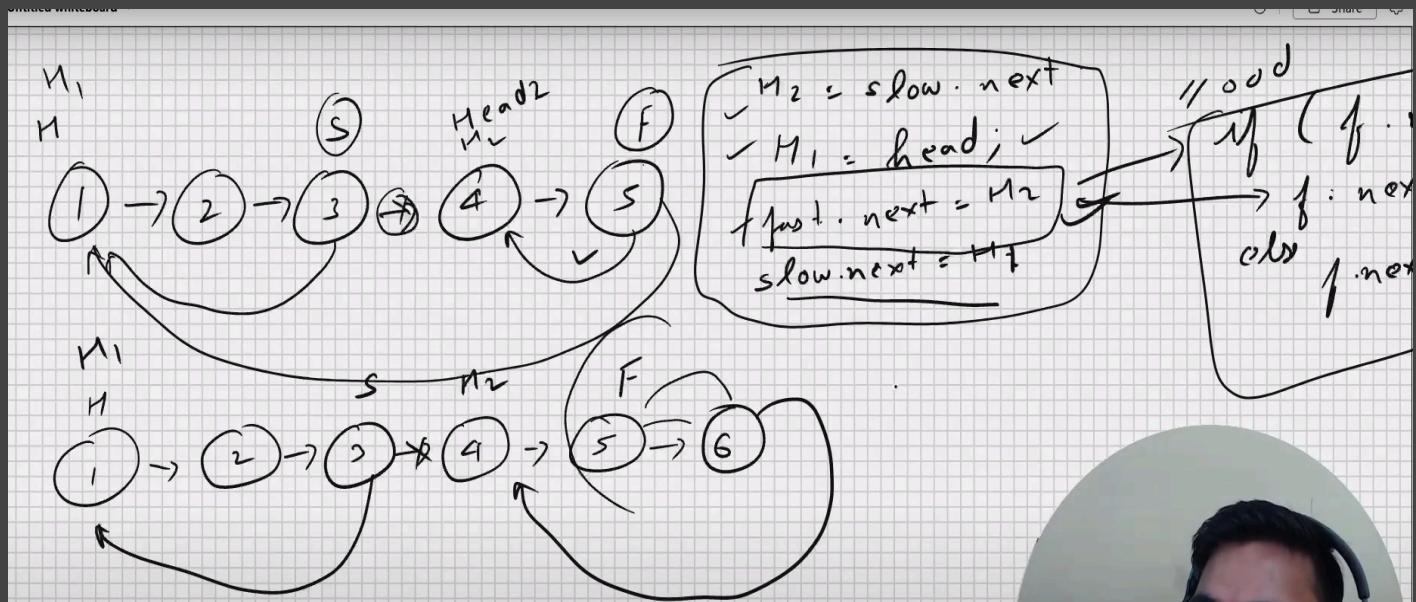
Remove Duplicate from Unsorted L

## Homework

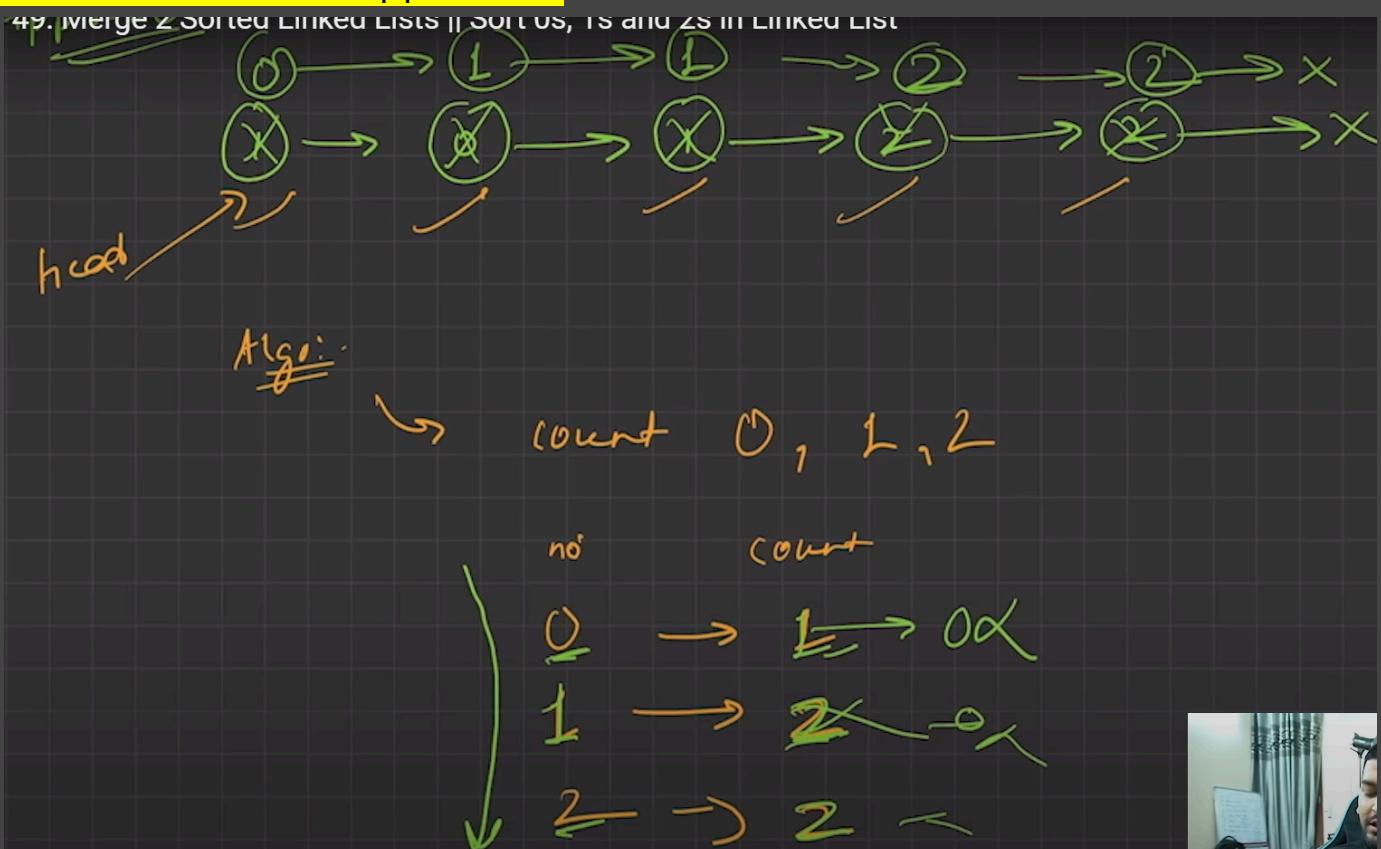
$$O(n) \rightarrow 2 \text{ loops}$$

$\rightarrow O(n \log n) \rightarrow \text{sort} \rightarrow O(n)$  suchen  
 $\Rightarrow \boxed{\text{mehr} \rightarrow O(n)}$

## Split A Circular LL Into Two Halves:



## Sort 0s, 1s & 2s In A LL Approach-I:



## Sort 0s, 1s & 2s In A LL Approach-II:

