S

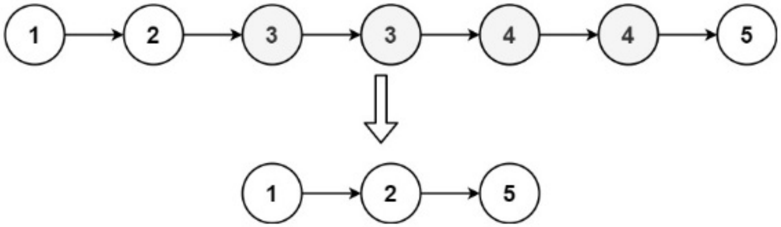
82. Remove Duplicates from Sorted List II

Medium ✓ 8.3K 214

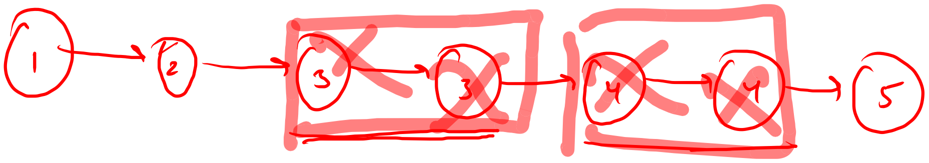
Companies

Given the `head` of a sorted linked list, *delete all nodes that have duplicate numbers, leaving only distinct numbers from the original list*. Return *the linked list **sorted** as well.*

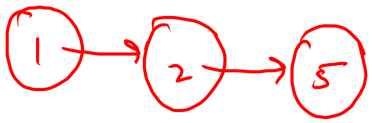
Example 1:

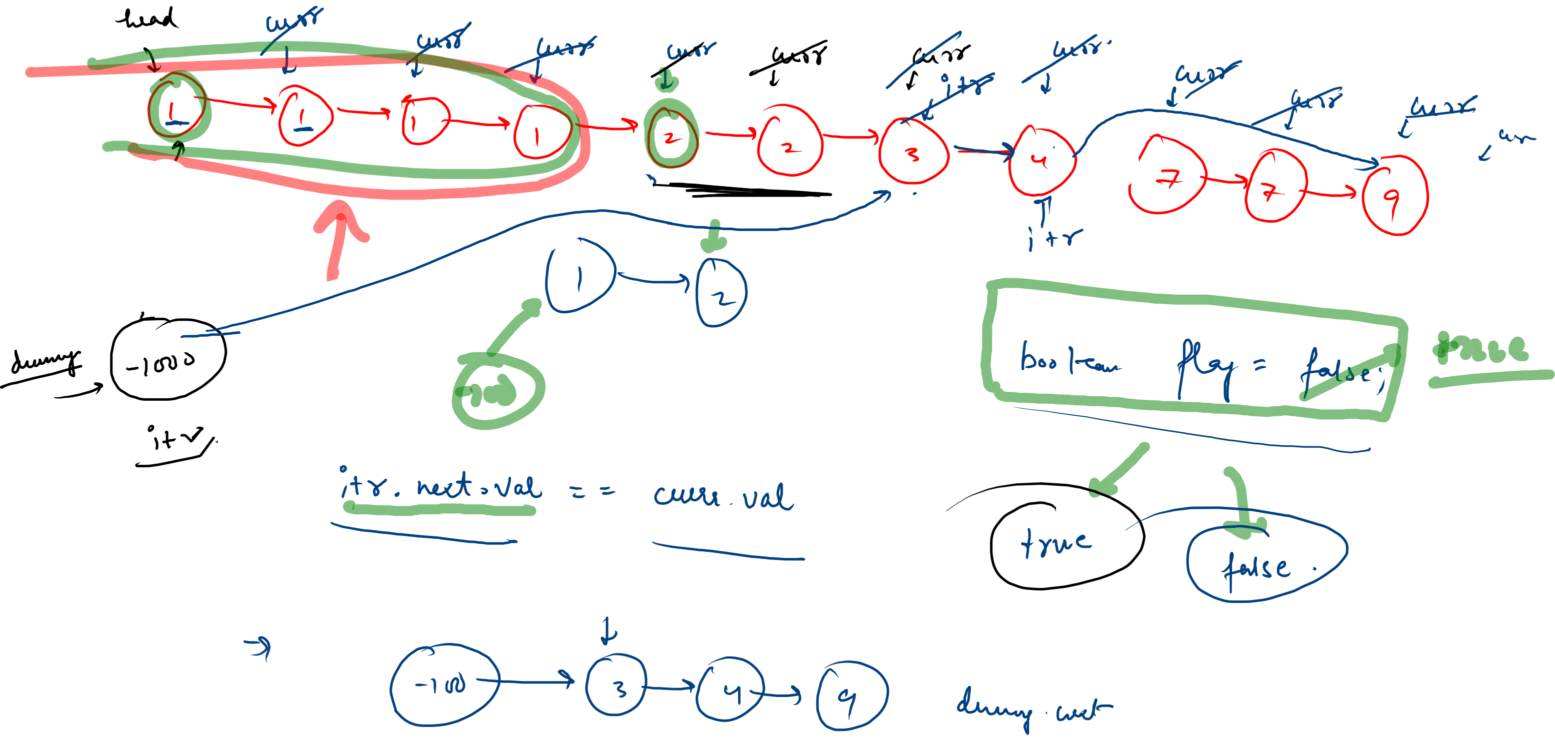


I.P.



O.P.





head
curr
dummy
-1000
itr
itr.next.val == curr.val
boolean flag = false;
true
true
false
-100
dummy.next

→ 1 Create a dummy node.

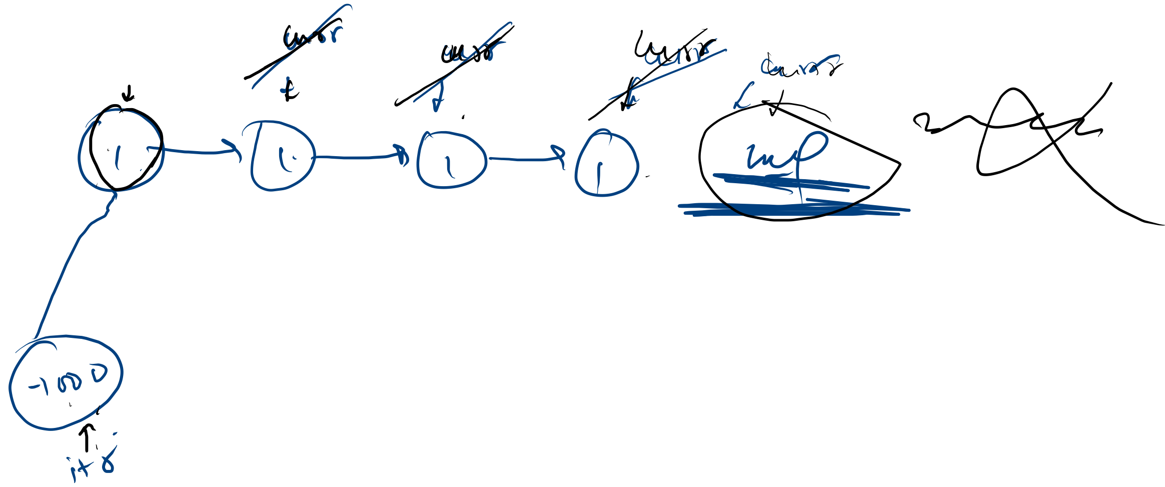
→ 2 Make a pointer which points to dummy node. (itr)

→ 3 itr.next = head;

→ 4 curr = head.next

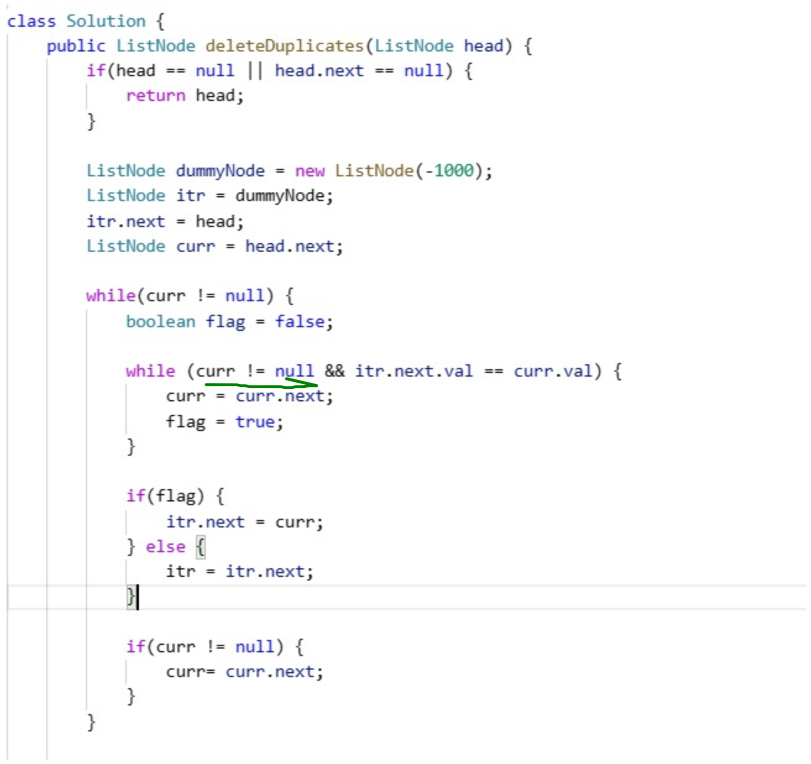
→ 5

```
while (curr != null)
{
    flag = false.
    while( itr.next.val == curr.val)
        curr = curr.next.
        flag = true;
    }

    if (flag)
        itr.next = curr;
    else
        itr = itr.next;
}
return dummy.next;
```



~~curr~~
~~curr~~
~~curr~~
curr
1
1
1
1
inf
-1000
itr

$$\begin{tikzcd}
1 \arrow[r] & 1 \arrow[r] & 1 \arrow[r] & 2 \arrow[r] & 3 \arrow[d] \\
 & & \boxed{1 \rightarrow 2 \rightarrow 3} & &
\end{tikzcd}$$



```java
class Solution {
    public ListNode deleteDuplicates(ListNode head) {
        if(head == null || head.next == null) {
            return head;
        }

        ListNode dummyNode = new ListNode(-1000);
        ListNode itr = dummyNode;
        itr.next = head;
        ListNode curr = head.next;

        while(curr != null) {
            boolean flag = false;

            while (curr != null && itr.next.val == curr.val) {
                curr = curr.next;
                flag = true;
            }

            if(flag) {
                itr.next = curr;
            } else {
                itr = itr.next;
            }

            if(curr != null) {
                curr= curr.next;
            }
        }
```

25. Reverse Nodes in k-Group

Hard 12.6K 624

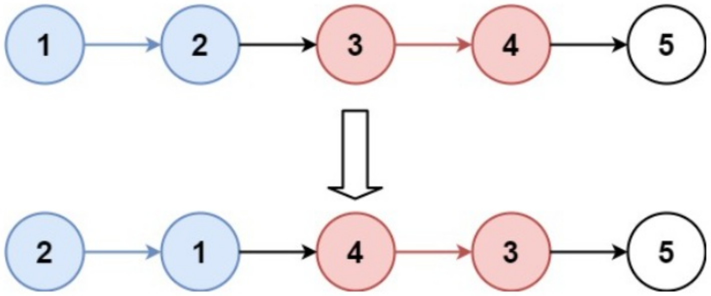
Companies

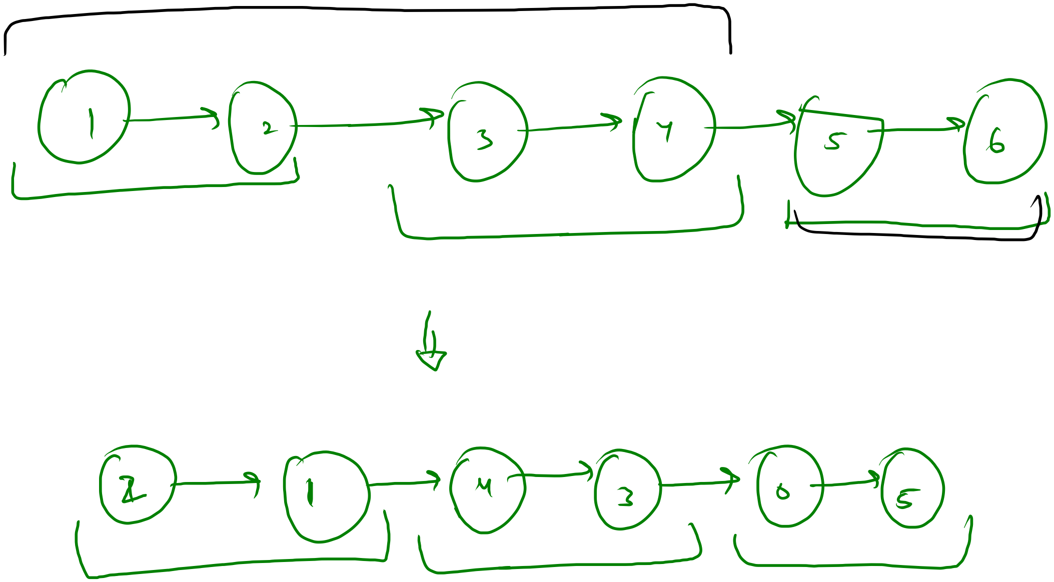
Given the `head` of a linked list, reverse the nodes of the list `k` at a time, and return *the modified list*.

`k` is a positive integer and is less than or equal to the length of the linked list. If the number of nodes is not a multiple of `k` then left-out nodes, in the end, should remain as it is.

You may not alter the values in the list's nodes, only nodes themselves may be changed.

Example 1:



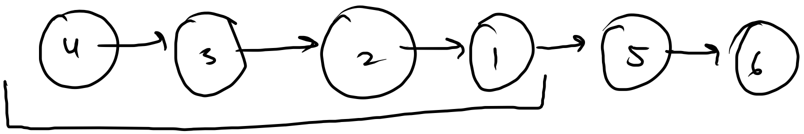


k=2

$\rightarrow$ $\boxed{\text{5 mins}}$

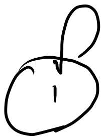
K

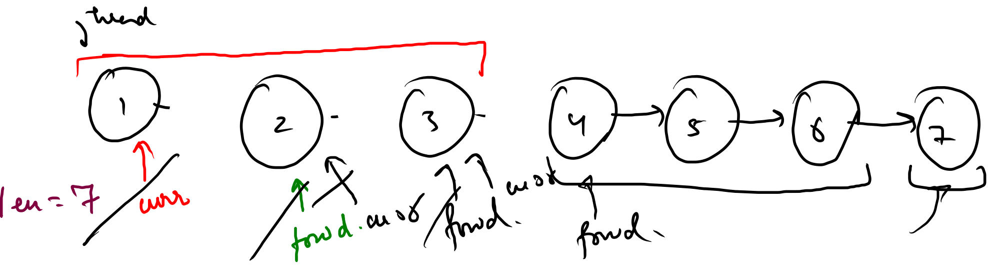
k=4



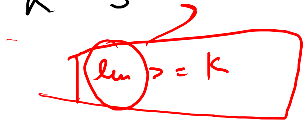
```
(1) if ( head == null || head.next == null || k == 0 || k == -1 )
            return head
```

(2) Find the length of LL;





k = 3



```
{ oh = null
  ot = null }
curr = head.
```

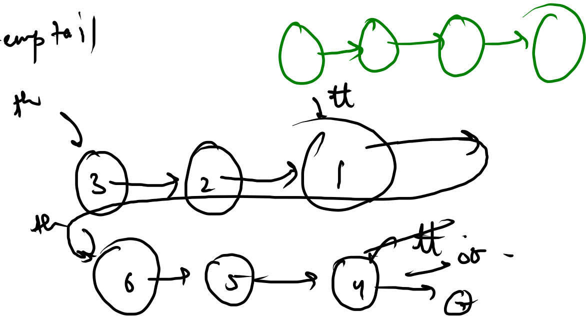
temp = K

```
while (temp--) {
    fowd = curr.next
    curr.next = null;
```

add first

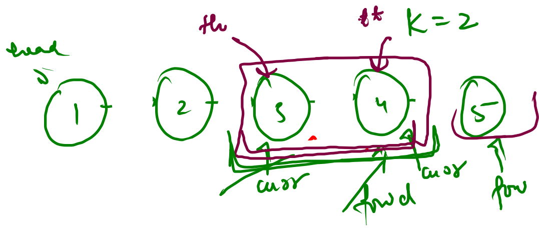
```
th = temph.
tt = temptail
```

```
addfirst (curr)
if (oh == null)          else
{ oh = th                 ot.next = th.
  ot = tt }               ot = tt.
```



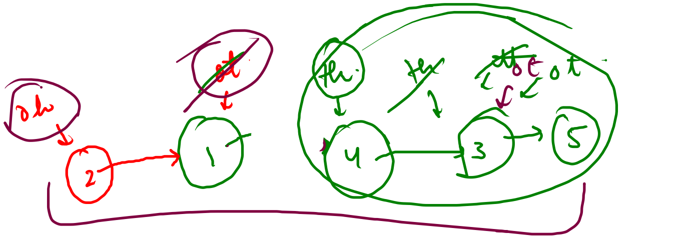
```
public ListNode reverseKGroup(ListNode head, int k) {
    if(head == null || head.next == null || k == 0 || k == 1) {
        return head;
    }

    int len = lengthOfLL(head);

    ListNode oh = null;
    ListNode ot = null;
    ListNode curr = head;

    while(len>=k) {
        int temp = k;
        while (temp > 0) {
            ListNode fowd = curr.next;
            curr.next = null;
            addFirstNode(curr);
            curr = fowd;
            temp--;
        }

        if (oh == null) {
            oh = th;
            ot = tt;
        } else {
            ot.next = th;
            ot = tt;
        }

        tt = null;
        th = null;
        len -=k;
    }

    ot.next = curr;
    return oh;
}
```

```
public int lengthOfLL(ListNode head) {
    int count = 0;
    ListNode temp = head;

    while (temp != null) {
        count++;
        temp = temp.next;
    }

    return count;
}

ListNode tt = null;
ListNode th = null;

public void addFirstNode(ListNode curr) {
    if(th == null){
        th = curr;
        tt = curr;
    } else {
        curr.next = th;
        th = curr;
    }
}
```



th

tt



th = null
tt = null

oh = null

ot = null

len = 5/ 3>=~~2~~ 1

5>=2

temp = ~~2~~ ~~1~~ 0

1>=2