T.C. o(n)
S.C. o(1)

Leetcode Daily Challenge

09/03/2022

problem

Remove Duplicates from Sorted List II

pre-requisites

linked list

difficulty
Medium

est. time
15-20 min

Let's build **Intuition**

can be asked in...

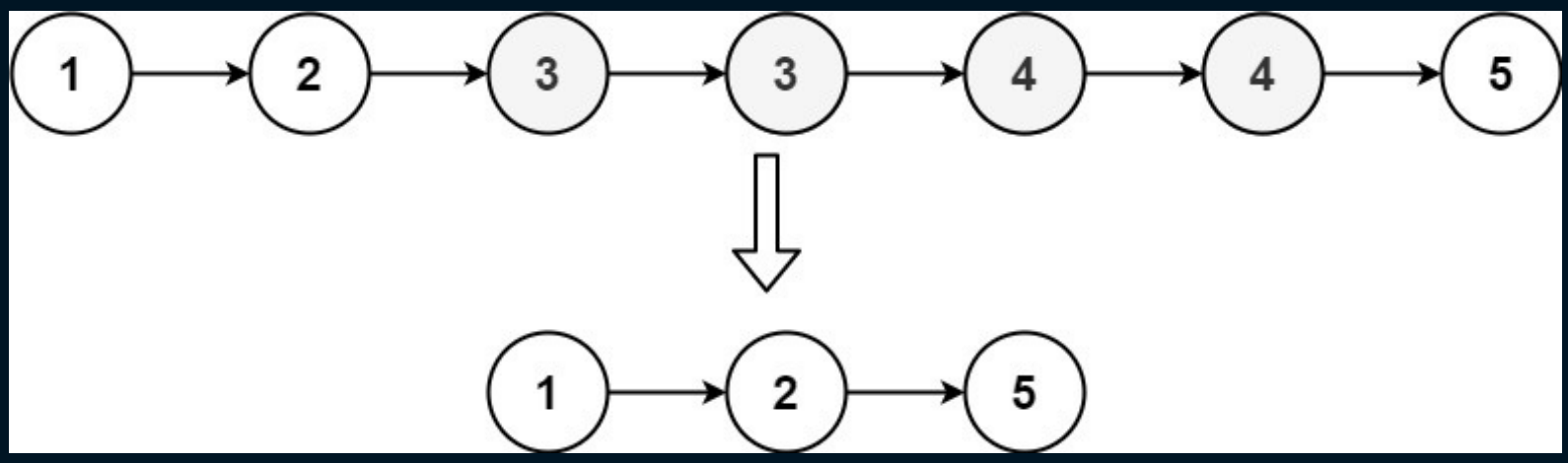
42% Accuracy

Statement

Description

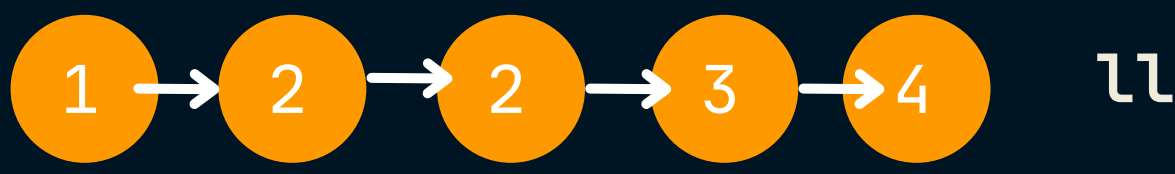
- Given the head of a sorted linked list, delete all nodes that have duplicate numbers, leaving only distinct numbers from the original list. Return the linked list sorted as well.

i/p	o/p
head = [1,2,3,3,4,4,5]	[1,2,5]



Intuition

- Let's make a function `deleteDuplicates()` which performs following tasks-
  1) Takes Linked List as input.
  2) Deletes the duplicate nodes
  3) Returns a list which has no 'duplicates'



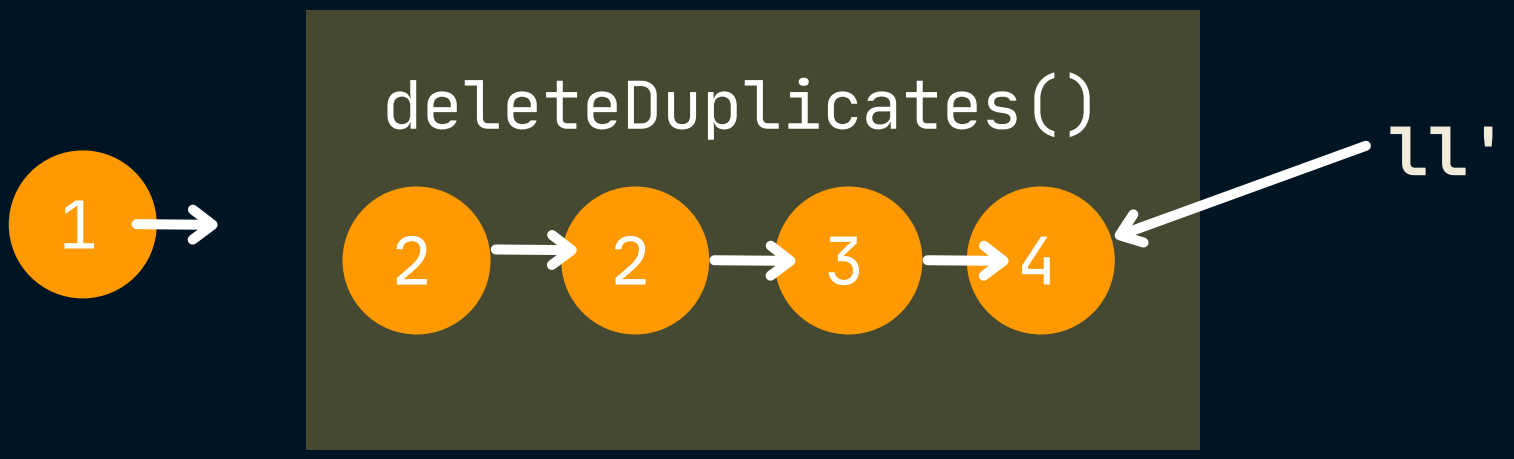
ll

Let's consider above list

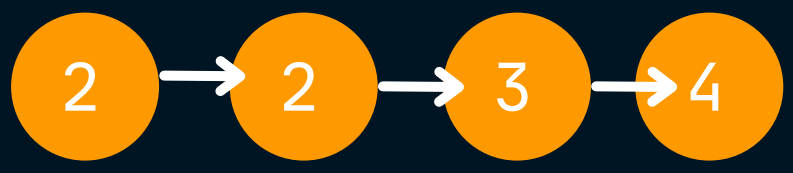
deleteDuplicates()

- we start with 1st node, traverse the list to see if there exists any duplicate of 1
- we don't find any so we keep 1 & ask `deleteDuplicates()` to take the remaining list, delete the duplicates & return remaining list with no dupl.

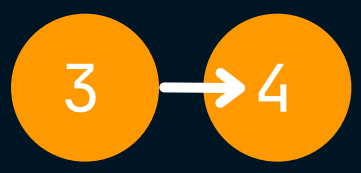
Intuition



- Now deleteDuplicates() take ll', deletes duplicates & return remaing list which gets attached to node 1's right.
  - let's see ll' in action-

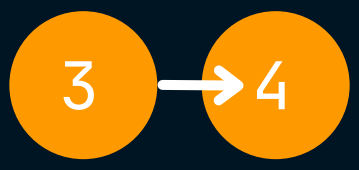


- since 2 is duplicate, deleteDuplicates() will delete these nodes & again check for remaining list
- Now after deleteDuplicates() deleted 2, our list looks like



ll''

Intuition

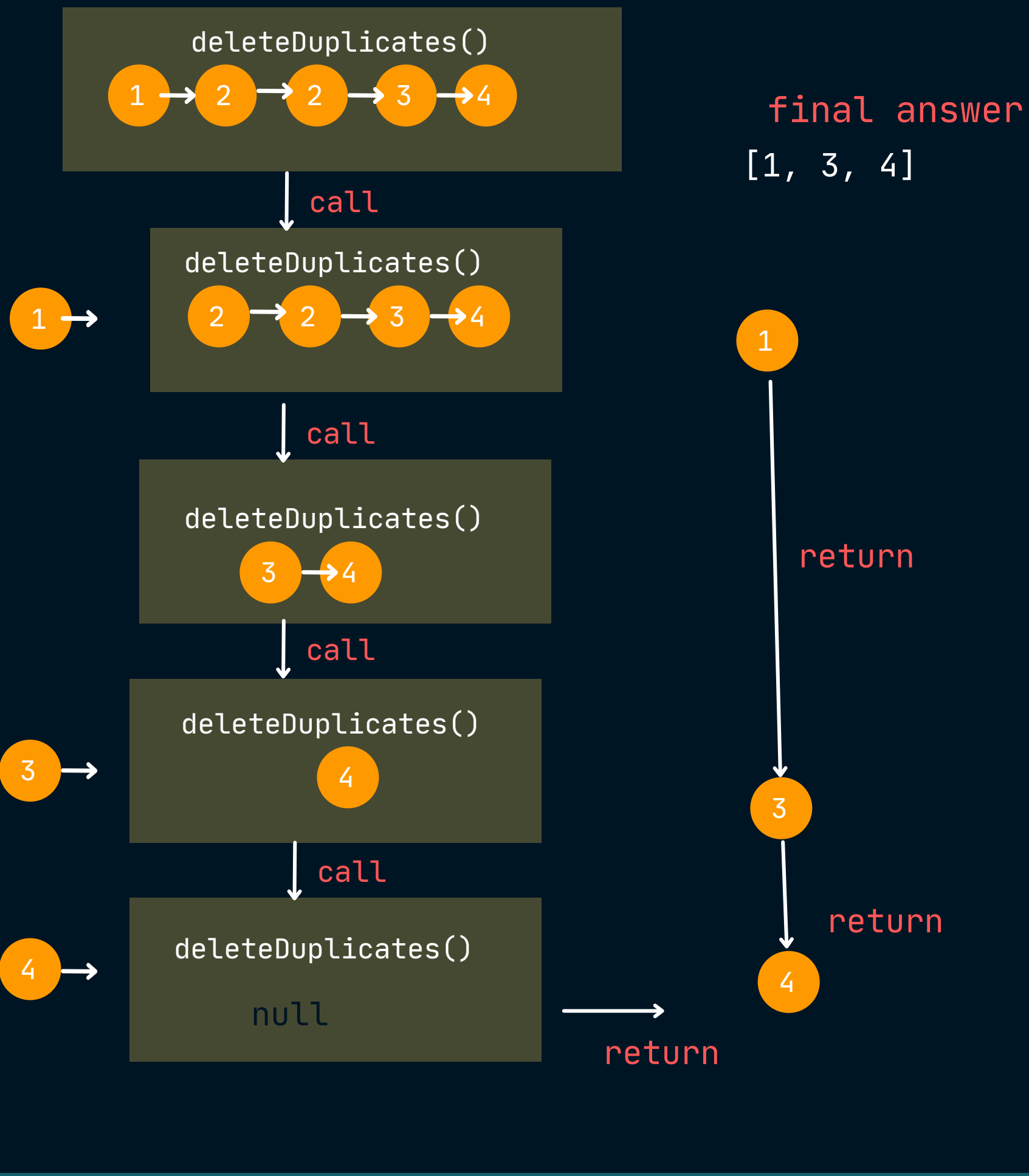


ll''

- Now deleteDuplicates() will work on ll''
- as 3 is not duplicated, so 3 won't be deleted & remaining list will be checked.

- now we are remaining with a single node, which is not repeated, so our last function call ends & we return the lists at end of each call.

Intuition


deleteDuplicates()
1 → 2 → 2 → 3 → 4
call
deleteDuplicates()
1 →
2 → 2 → 3 → 4
call
deleteDuplicates()
3 → 4
call
deleteDuplicates()
3 →
4
call
deleteDuplicates()
4 →
null
return
return
return
1
3
4
final answer
[1, 3, 4]

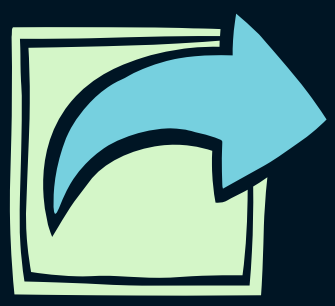
Algorithm

- Can you visualize how 'deleteDuplicates()' is doing-
  1) take list as input.
  2) iterate till you you get a node whose value is not equal to head value.
  3) While iterating if duplicate found, delete duplicates
  4) recur for remaining list.

```cpp
class Solution {
public:


    ListNode* deleteDuplicates(ListNode* head) {


        if(!head || !head->next) {
            return head;
        }

        int val = head->val;
        ListNode* currNode = head->next;

        if(currNode->val != val) {
            head->next = deleteDuplicates(currNode);
            return head;
        } else {
            while(currNode && currNode->val == val) {
            ListNode* dummy = currNode;
            currNode = currNode->next;
            delete dummy;
        }
            delete head;
            return deleteDuplicates(currNode);
        }
    }
};
```

Leave a Like

Comment if you love posts like this, will motivate me to make posts like these



Share, with friends