# Leetcode
# Daily Challenge

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

24/02/2022

problem

Sort List

pre-requisites

Sorting, Recursion

difficulty
**Medium**

est. time
**15-20 mins.**

Asked in...

**49%**
Accuracy

# Statement

## Description

Given head of a linked list, return head of sorted ll in asc. order

| I/P | O/P |
| --- | --- |
| head = [4,2,1,3] | [1,2,3,4] |

## Constraints

- The number of nodes in the list is in the range [0, 5 * 10^4].
- -10^5 <= Node.val <= 10^5

# Approaches

## Approach #1

-> Copy all values in an array.
-> Sort the array & paste values in LL.

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

## Approach #2

-> Apply merge sort on LL.

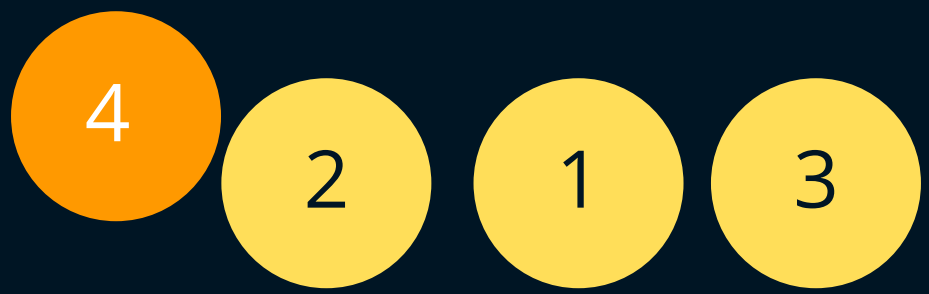
T.C. o(nlogn)  S.C. o(1)

-> There are more solutions, but here I'll use this problem to help you overcome fear of **Recursion**

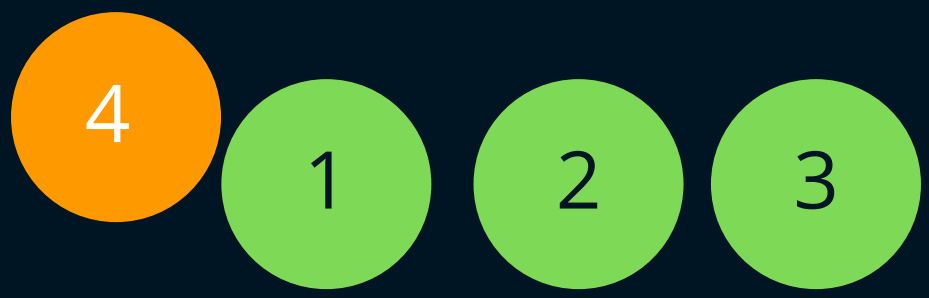
# Intuition


Given this LL

-> Let's start sorting...



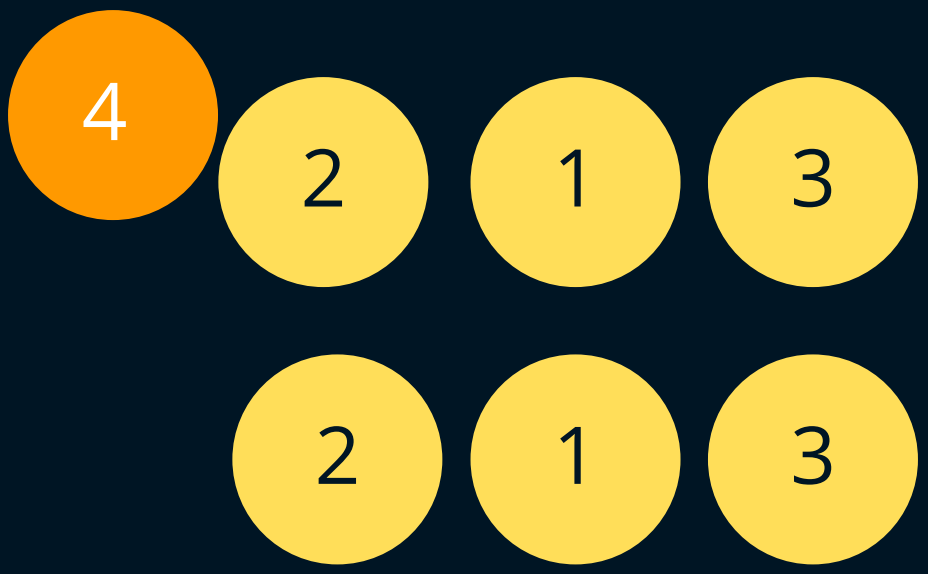

You are at 1st node,if nodes to right are sorted then we can just put (4) to its place.





Magic happened & node (2),(1),(3) are sorted, just put node (4) at its place

# Magic ?




Magic was required so that node 2,1,3 are sorted.

can we say node 2,1,3 are themselves forming a Linked List

Say, you are writing a function sort()
which sorts 

LL1

can we use same
sort() to sort 

LL2

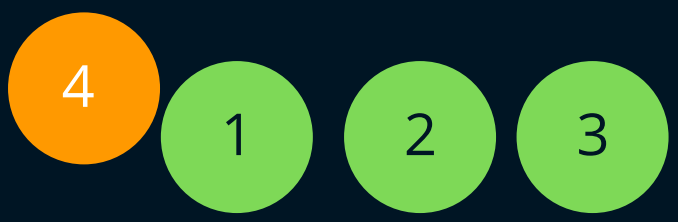
To sort LL1, we kept (4) aside & called sort() for LL2

RECURSION...

# IBH


Hypothesis

-> Keep (4) aside & call sort(), for remaining LL.
-> Believe in sort(), it will sort your remaining LL. (2),(1),(3)




After hypothesis step.


Induction

Now just keep (4), at it's place in sorted LL

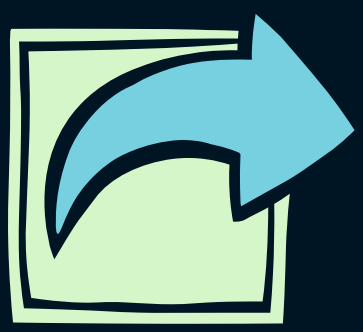
Base ?

You tell...

```
class Solution {
public:


ListNode* sortLL(ListNode* node)
{
    if(node == NULL) return NULL;
    ListNode* cur = node;
    ListNode* sortedListHead = sortLL(node->next);
    node = sortedListHead;
    ListNode* pre = NULL;
    while(node){
        if(cur->val <= node->val){
            break;
        }
        pre = node;
        node = node->next;
    }

    if(pre == NULL){
        cur->next = sortedListHead;
        return cur;
    }
    else{
        pre->next = cur;
        cur->next = node;
        return sortedListHead;
    }
}

    ListNode* sortList(ListNode* head) {
        return sortLL(head);
    }
```

Leave a Like

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



Share, with friends