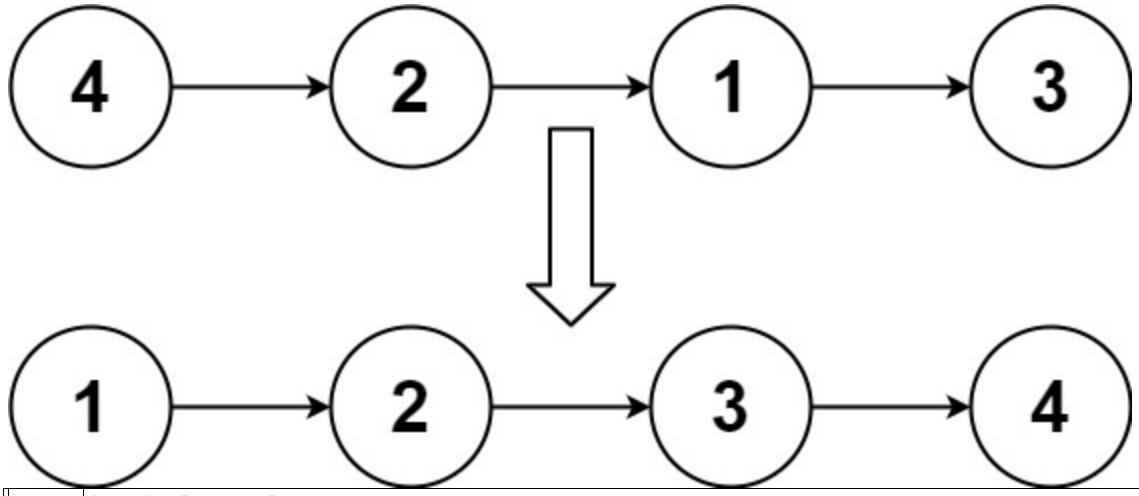
148. Sort List
Medium
9.6K
290
Companies

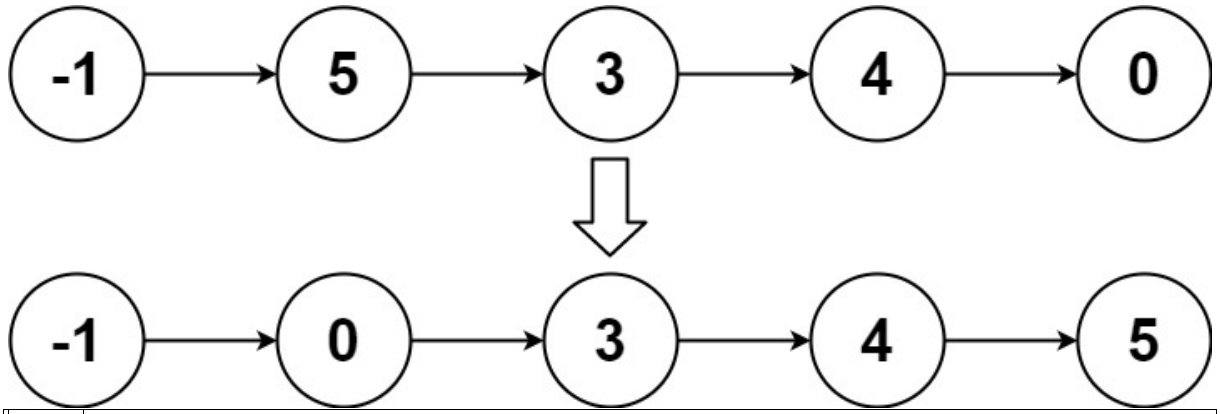
Given the head of a linked list, return *the list after sorting it in **ascending order***.

Example 1:



Input: head = [4,2,1,3]
Output: [1,2,3,4]

Example 2:



Input: head = [-1,5,3,4,0]
Output: [-1,0,3,4,5]

Example 3:

Input: head = []
Output: []

Constraints:

- The number of nodes in the list is in the range [0, 5 * 10⁴].
- -10⁵ <= Node.val <= 10⁵

Follow up: Can you sort the linked list in O(n logn) time and O(1) memory (i.e. constant space)?

Code link:- https://leetcode.com/problems/sort-list/

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x), next(nullptr) {}
 *     ListNode(int x, ListNode *next) : val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* findmid(ListNode* head){
        ListNode*slow =head;
        ListNode* fast=head;
        while(fast->next==NULL&&fast->next->next!=NULL){
            slow=slow->next;
            fast=fast->next->next;
        }
        return slow;
    }
    ListNode* mergsort(ListNode* h1,ListNode* h2)   {
        if(h1==NULL){
            return h2;
        }
        if(h2==NULL){
            return h1;
        }
         ListNode* ans=new ListNode(0);
        ListNode* cur=ans;
        while(h1!=NULL&&h2!=NULL)
        {
            if(h1->val<h2->val){
                cur->next=h1;
                h1=h1->next;
            }
            else{
                cur->next=h2;
                h2=h2->next;
            }
```

```
            cur=cur->next;
        }
        if(h1!=NULL){
            cur->next=h1;
        }
        if(h2!=NULL){
            cur->next=h2;
        }
        return ans->next;
    }
    ListNode* sortList(ListNode* head) {
        if(head==NULL||head->next==NULL){
            return head;
        }
        ListNode* mid=findmid(head);
        ListNode* newhead=mid->next;
        mid->next=NULL;
        ListNode* left=sortList(head);
        ListNode* right=sortList(newhead);
        return mergsort(left,right);
    }
};
```