https://leetcode.com/problems/insertion-sort-list/

Problem List | Submit | 0 | 00:17:57 | Premium

Description | Editorial | Solutions | Submissions

147. Insertion Sort List

Medium | Topics | Companies

Given the `head` of a singly linked list, sort the list using **insertion sort**, and return *the sorted list's head*.

The steps of the **insertion sort** algorithm:

1. Insertion sort iterates, consuming one input element each repetition and growing a sorted output list.
2. At each iteration, insertion sort removes one element from the input data, finds the location it belongs within the sorted list and inserts it there.
3. It repeats until no input elements remain.

The following is a graphical example of the insertion sort algorithm. The partially sorted list (black) initially contains only the first element in the list. One element (red) is removed from the input data and inserted in-place into the sorted list with each iteration.

3.4K | 66 | 31 Online

Code | Note

C++ | Auto

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

head =
[4,2,1,3]

Output

[1,2,3,4]

Expected

[1,2,3,4]

Contribute a testcase

leetcode - Search | Insertion Sort List - LeetCode

https://leetcode.com/problems/insertion-sort-list/submissions/1892122266/

Problem List | Submit | 00:17:57 | Premium

Description | Accepted | Editorial | Solutions | Submissions

All Submissions

Accepted 19 / 19 testcases passed

BhavneetKaur_19 submitted at Jan 21, 2026 18:11

Editorial | Solution

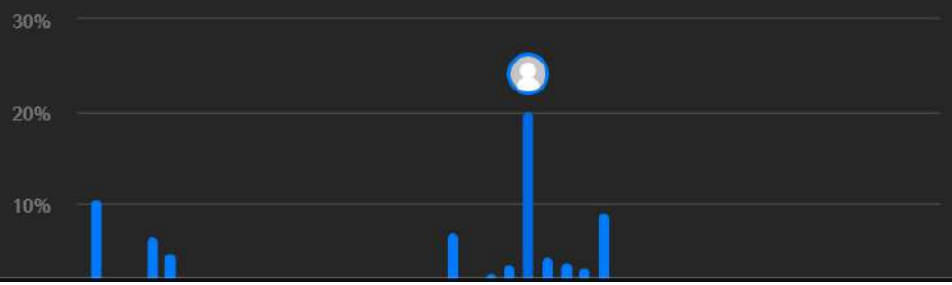
Runtime

23 ms | Beats 54.13%

Analyze Complexity

Memory

14.72 MB | Beats 22.51%



Code | Note

C++ | Auto

```cpp
class Solution {
public:
    ListNode* insertionSortList(ListNode* head) {
        if (!head || !head->next) return head;

        ListNode* dummy = new ListNode(0);
        ListNode* curr = head;

        while (curr) {
            ListNode* nextNode = curr->next;
            ListNode* prev = dummy;

            while (prev->next && prev->next->val < curr->val) {
                prev = prev->next;
            }

            curr->next = prev->next;
            prev->next = curr;
            curr = nextNode;
        }

```

Saved | Ln 35, Col 1

Testcase | Test Result

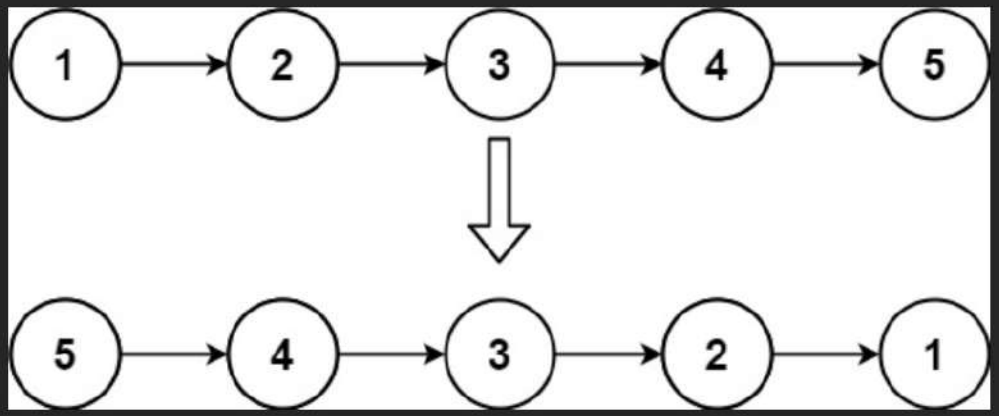
Description | Editorial | Solutions | Submissions

206. Reverse Linked List

Easy | Topics | Companies

Given the `head` of a singly linked list, reverse the list, and return *the reversed list*.

Example 1:



24.1K | 384 | 379 Online

Code | Note

C++ | Auto

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

head =
[1,2,3,4,5]

Output

[5,4,3,2,1]

Expected

[5,4,3,2,1]

Contribute a testcase

Problem List | Submit | 00:21:47 | Premium

Description | Editorial | Solutions | Accepted | Submissions

All Submissions

Accepted 28 / 28 testcases passed **Time taken: 21 m 47 s**

BhavneetKaur_19 submitted at Jan 21, 2026 18:18

Editorial | Solution

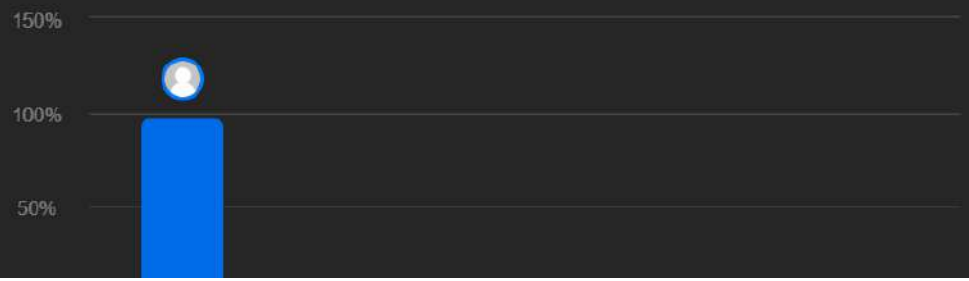
Runtime

0 ms | Beats **100.00%**

Analyze Complexity

Memory

13.27 MB | Beats **91.11%**



Code | Note

C++ | Auto

```cpp
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode* prev = nullptr;
        ListNode* curr = head;
        while (curr != nullptr) {
            ListNode* temp = curr->next;
            curr->next = prev;
            prev = curr;
            curr = temp;
        }
        return prev;
    }
};
```

Saved

Ln 8, Col 43

Testcase | Test Result