DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

# Assignment 3.

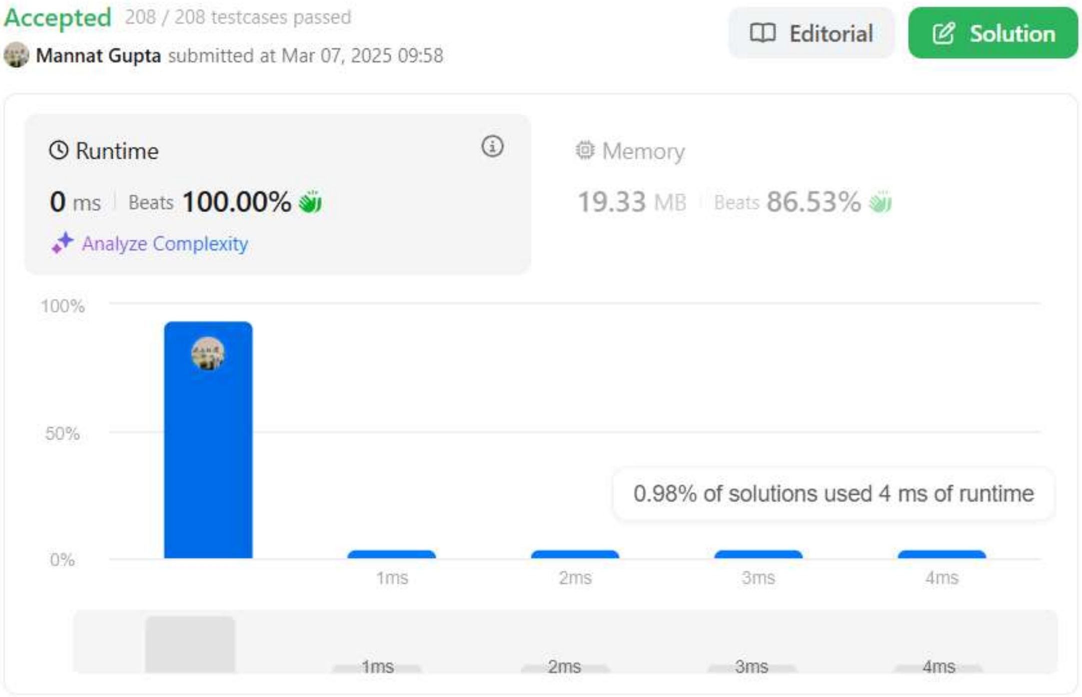
| | |
|---|---|
| **Student Name:** Mannat Gupta | **UID:** 22BCS15281 |
| **Branch:** BE-CSE | **Section/Group:** 608-B |
| **Semester:** 6th | **Date of Performance:** 07/03/25 |
| **Subject Name:** Advanced Programming Lab-2 | **Subject Code:** 22CSP-351 |

**Aim:** Linked Lists:

1. **Merge two sorted linked lists:**

   https://leetcode.com/problems/merge-two-sorted-lists/description/

```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
        if (!list1) return list2;
        if (!list2) return list1;

        ListNode* head = new ListNode(0); // Dummy node
        ListNode* tail = head;

        while (list1 && list2) {
            if (list1->val < list2->val) {
                tail->next = list1;
                list1 = list1->next;
            } else {
                tail->next = list2;
                list2 = list2->next;
            }
            tail = tail->next;
        }

        tail->next = list1 ? list1 : list2;
        return head->next;
    }
};
```



Accepted 208 / 208 testcases passed

Mannat Gupta submitted at Mar 07, 2025 09:58

Runtime

0 ms | Beats 100.00%

Memory

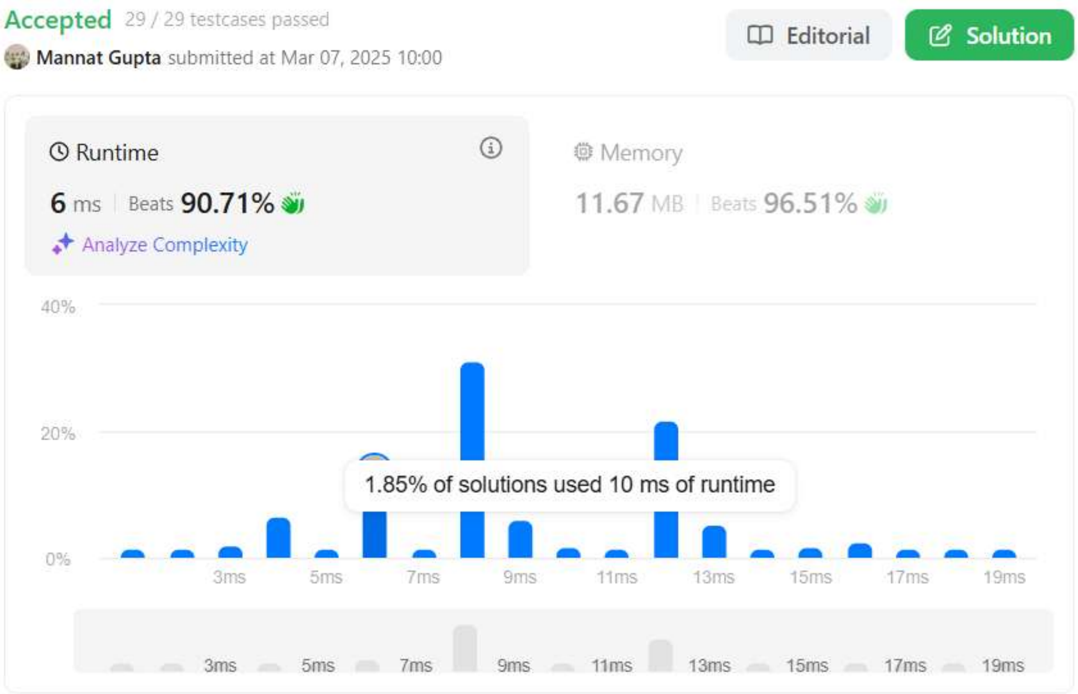
19.33 MB | Beats 86.53%

0.98% of solutions used 4 ms of runtime

2. **Detect a cycle in a linked list:**

   [https://leetcode.com/problems/linked-list-cycle/description/](https://leetcode.com/problems/linked-list-cycle/description/)

```
class Solution {
public:
    bool hasCycle(ListNode *head) {
        if (!head || !head->next) return false;

        ListNode *slow = head, *fast = head;

        while (fast && fast->next) {
            slow = slow->next;
            fast = fast->next->next;

            if (slow == fast) return true;
        }
        return false;
    }
};
```



Accepted 29 / 29 testcases passed

Mannat Gupta submitted at Mar 07, 2025 10:00

Runtime

6 ms | Beats 90.71%

Memory

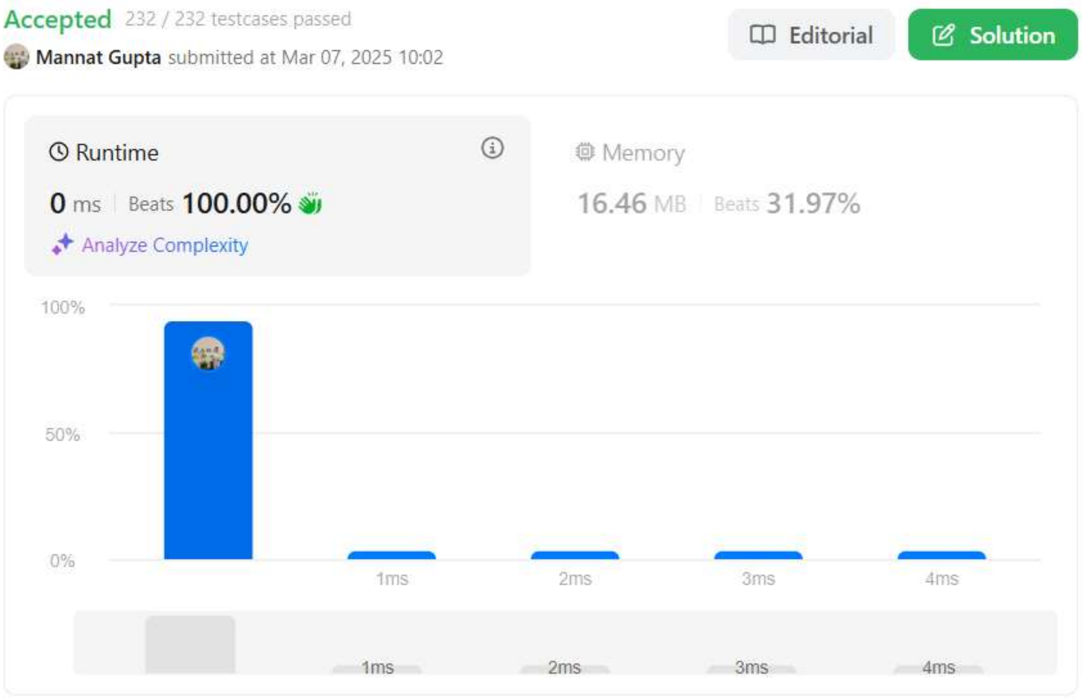
11.67 MB | Beats 96.51%

1.85% of solutions used 10 ms of runtime

### 3. Rotate a list:

[https://leetcode.com/problems/rotate-list/description/](https://leetcode.com/problems/rotate-list/description/)

```
class Solution {
public:
    ListNode* rotateRight(ListNode* head, int k) {
        if (!head || !head->next || k == 0) return head;

        int length = 1;
        ListNode* tail = head;
        while (tail->next) {
            tail = tail->next;
            length++;
        }
        tail->next = head;

        k = k % length;
        int stepsToNewHead = length - k;
        ListNode* newTail = head;
        for (int i = 1; i < stepsToNewHead; i++) {
            newTail = newTail->next;
        }

        head = newTail->next;
        newTail->next = nullptr;

        return head;
    }
};
```



Accepted 232 / 232 testcases passed

Mannat Gupta submitted at Mar 07, 2025 10:02

Runtime

0 ms | Beats 100.00%

Memory

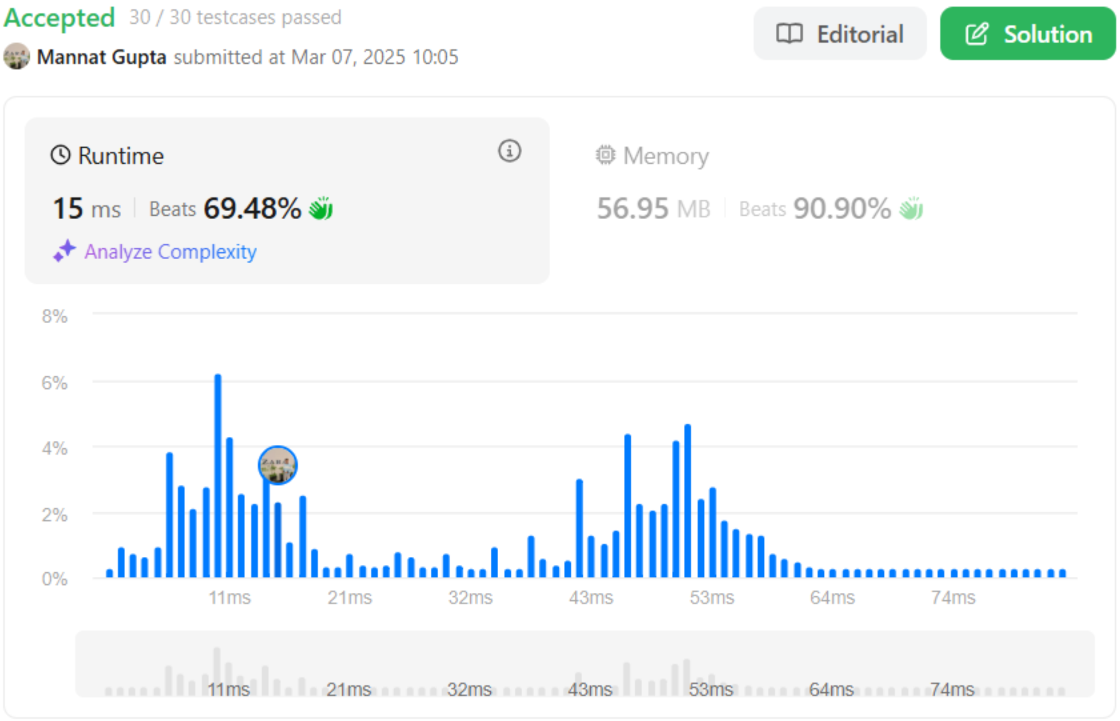
16.46 MB | Beats 31.97%

### 4. Sort List:

[https://leetcode.com/problems/sort-list/description/](https://leetcode.com/problems/sort-list/description/)

```
class Solution {
```

```
public:
    ListNode* sortList(ListNode* head) {
        if (!head || !head->next) return head;

        ListNode* mid = getMiddle(head);
        ListNode* left = head;
        ListNode* right = mid->next;
        mid->next = nullptr;

        left = sortList(left);
        right = sortList(right);

        return merge(left, right);
    }

private:
    ListNode* getMiddle(ListNode* head) {
        ListNode* slow = head, *fast = head->next;
        while (fast && fast->next) {
            slow = slow->next;
            fast = fast->next->next;
        }
        return slow;
    }

    ListNode* merge(ListNode* l1, ListNode* l2) {
        ListNode dummy(0);
        ListNode* tail = &dummy;

        while (l1 && l2) {
            if (l1->val < l2->val) {
                tail->next = l1;
                l1 = l1->next;
            } else {
                tail->next = l2;
                l2 = l2->next;
            }
            tail = tail->next;
        }

        tail->next = l1 ? l1 : l2;
        return dummy.next;
    }
};
```



Accepted 30 / 30 testcases passed

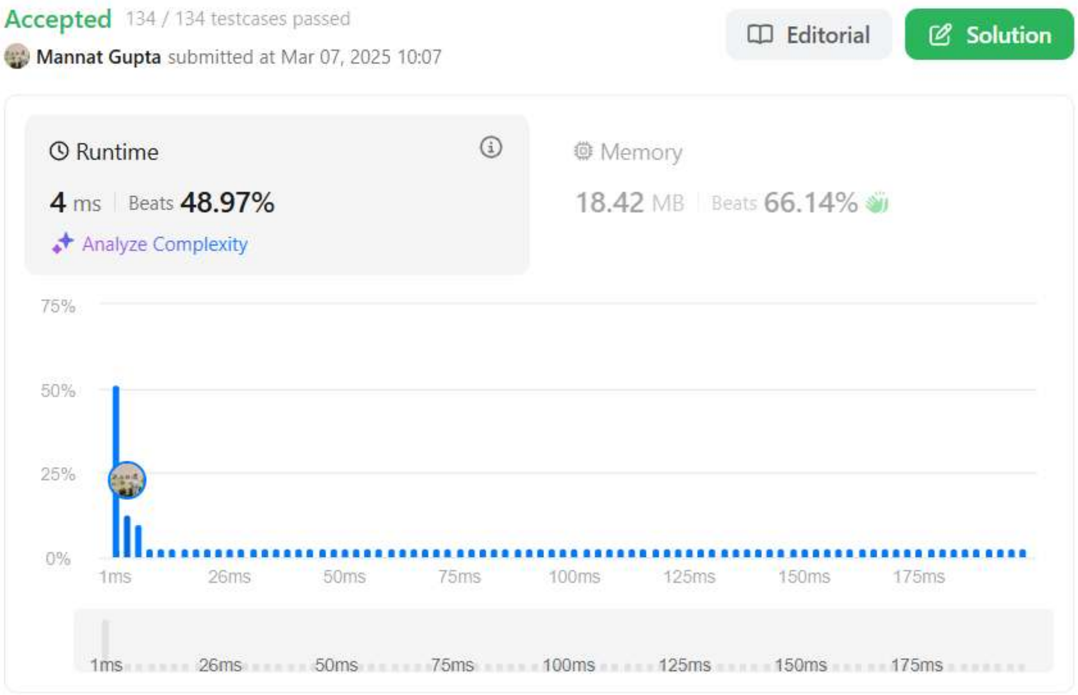
Mannat Gupta submitted at Mar 07, 2025 10:05

## 5. Merge k sorted lists:

[https://leetcode.com/problems/merge-k-sorted-lists/description/](https://leetcode.com/problems/merge-k-sorted-lists/description/)

```
class Solution {
public:
    ListNode* mergeKLists(vector<ListNode*>& lists) {
        auto cmp = [](ListNode* a, ListNode* b) { return a->val > b->val; };
        priority_queue<ListNode*, vector<ListNode*>, decltype(cmp)>
minHeap(cmp);
```

```
        for (auto list : lists) {
            if (list) minHeap.push(list);
        }

        ListNode dummy(0);
        ListNode* tail = &dummy;

        while (!minHeap.empty()) {
            ListNode* node = minHeap.top();
            minHeap.pop();

            tail->next = node;
            tail = node;

            if (node->next) minHeap.push(node->next);
        }

        return dummy.next;
    }
};
```


Accepted 134 / 134 testcases passed

Mannat Gupta submitted at Mar 07, 2025 10:07

Runtime

4 ms | Beats 48.97%

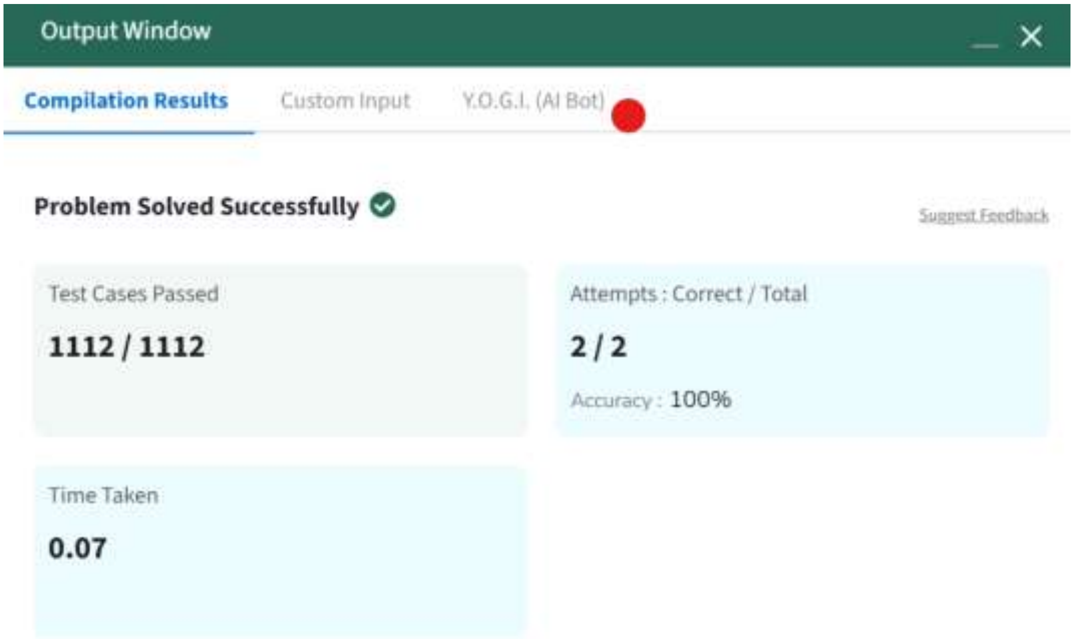
Memory

18.42 MB | Beats 66.14%

6. **Print Linked List:**

**https://www.geeksforgeeks.org/problems/print-linked-list-elements/0**

```
class Solution {

  public:
    void printList(Node *head) {
        Node* current = head;
        while (current != NULL) {
            cout << current->data << " ";
            current = current->next;
        }
    }
};
```



Problem Solved Successfully

Test Cases Passed
1112 / 1112

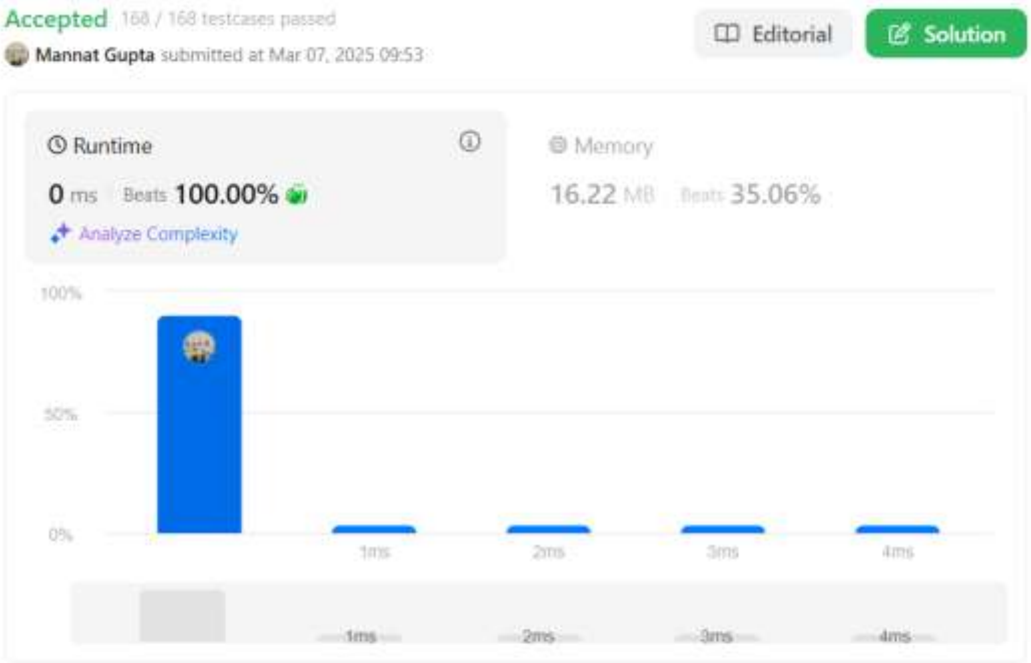
Attempts : Correct / Total
2 / 2
Accuracy : 100%

Time Taken
0.07

7. **Remove duplicates from a sorted list:**

**https://leetcode.com/problems/remove-duplicates-from-sorted-list/description/**

```
class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        ListNode* current = head;
        while (current && current->next) {
            if (current->val == current->next->val) {
                current->next = current->next->next;
            } else {
                current = current->next;
            }
        }
        return head;
    }
};
```



Accepted 168 / 168 testcases passed

Mannat Gupta submitted at Mar 07, 2025 09:53

Runtime

0 ms Beats 100.00%

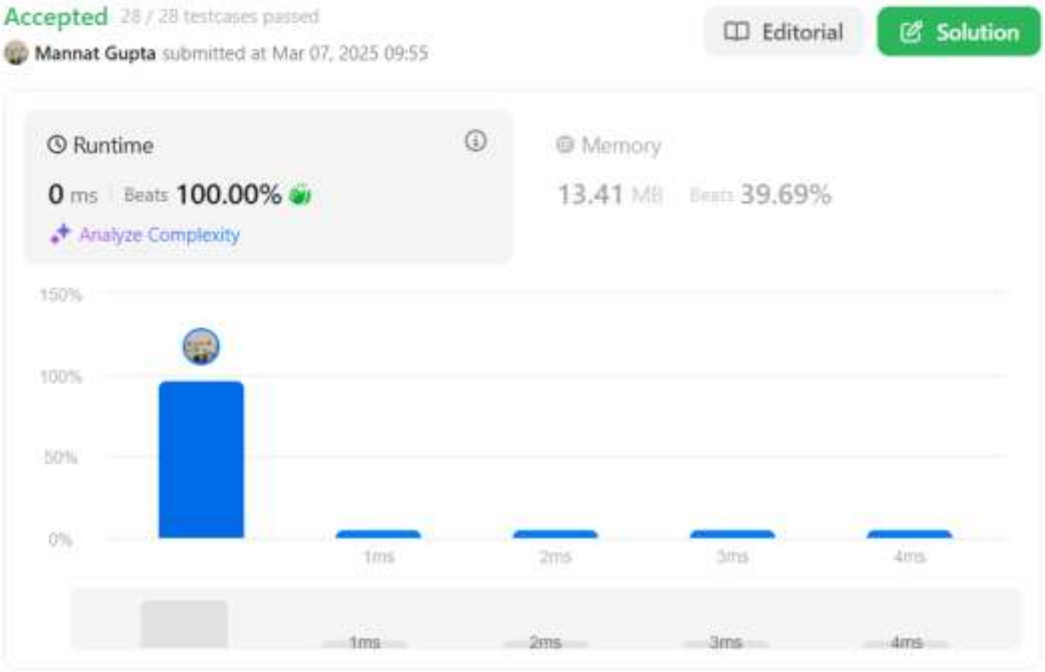
Memory

16.22 MB Beats 35.06%

## 8. Reverse a linked list:

[https://leetcode.com/problems/reverse-linked-list/description/](https://leetcode.com/problems/reverse-linked-list/description/)

```
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode* prev = NULL;
        ListNode* current = head;

        while (current) {
            ListNode* nextNode = current->next;
            current->next = prev;
            prev = current;
            current = nextNode;
        }

        return prev;
    }
};
```



Accepted 28 / 28 testcases passed
Mannat Gupta submitted at Mar 07, 2025 09:55

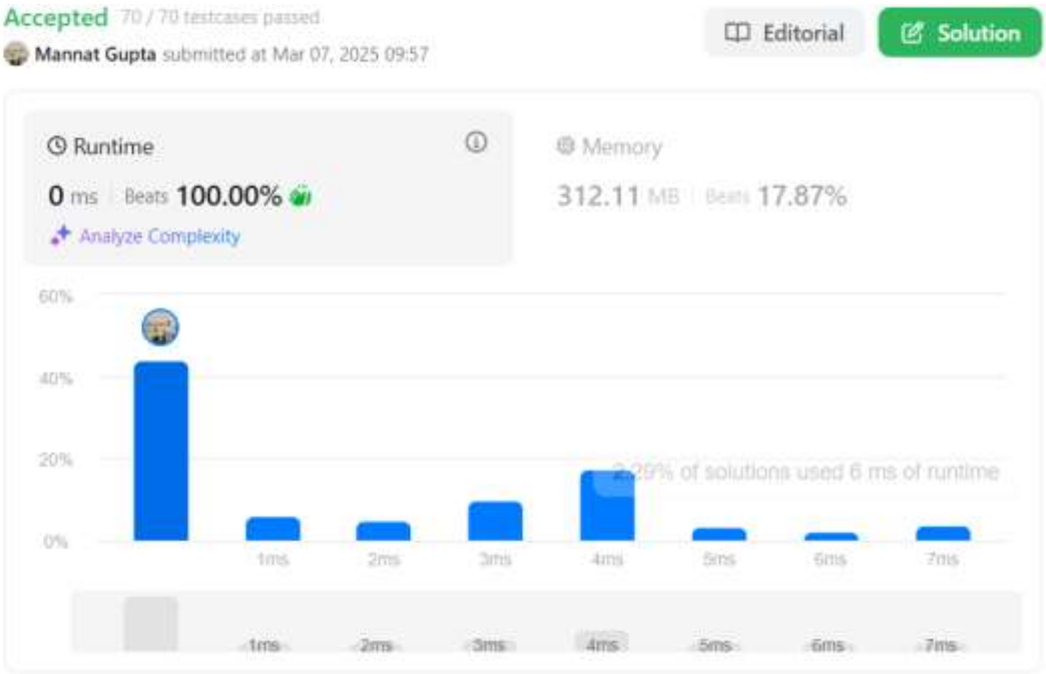
Runtime
0 ms | Beats 100.00%

Memory
13.41 MB | Beats 39.69%

9. **Delete middle node of a list:**

**[https://leetcode.com/problems/delete-the-middle-node-of-a-linked-list/description/](https://leetcode.com/problems/delete-the-middle-node-of-a-linked-list/description/)**

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

        ListNode* slow = head, *fast = head, *prev = nullptr;

        while (fast && fast->next) {
            prev = slow;
            slow = slow->next;
            fast = fast->next->next;
        }

        prev->next = slow->next;
        delete slow;
        return head;
    }
};
```



Accepted 70 / 70 testcases passed

Mannat Gupta submitted at Mar 07, 2025 09:57

Runtime

0 ms | Beats 100.00%

Memory

312.11 MB | Beats 17.87%

2.29% of solutions used 6 ms of runtime