ASSIGNMENT-3

Student Name :- Pratham Kapoor **University ID :- 22BCS10732**

Branch :- B.E. (C.S.E.) **Section/Group :- IOT_610-B**

Semester :- Sixth **Date of Performance :- 05/3/25**

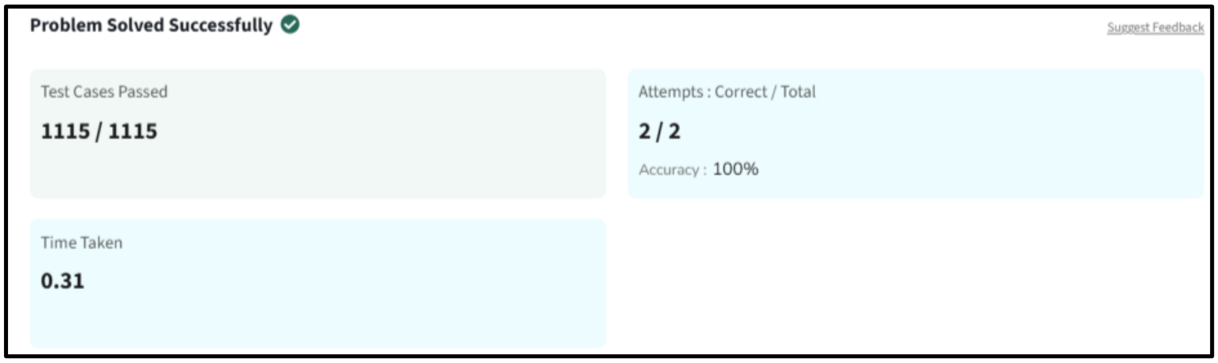
Subject Name :- Advanced Programming Lab **Subject Code :- 22CSP-351**

Problem-1 :- Given a linked list. Print all the elements of the linked list separated by space followed. **(GeeksforGeeks)**

Source Code

```
class Solution {
  public:
    // Function to display the elements of a linked list in the same line
    void printList(Node *head) {
        Node* temp = head;
        while (temp != nullptr) {
            cout << temp->data << " ";
            temp = temp->next; } }
};
```

Output

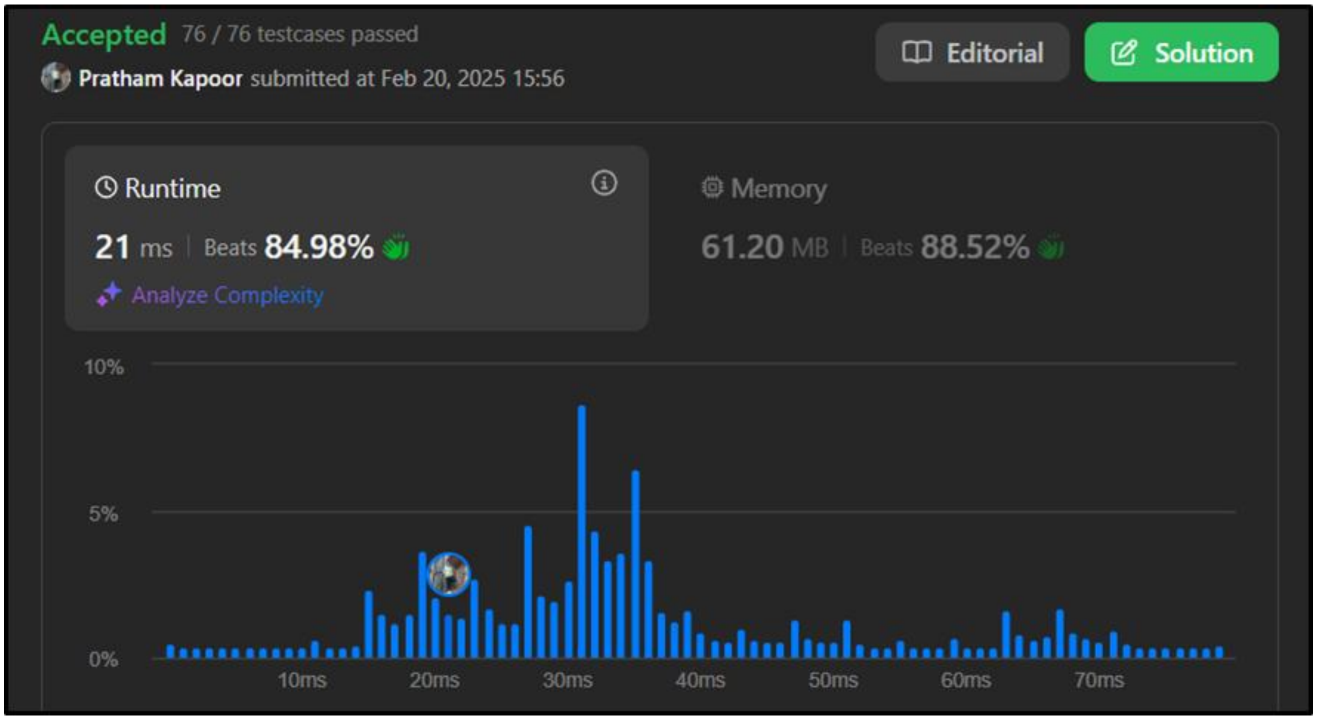

Problem Solved Successfully
Suggest Feedback
Test Cases Passed
1115 / 1115
Attempts : Correct / Total
2 / 2
Accuracy : 100%
Time Taken
0.31

Problem-2 :- Given the head of a sorted linked list, delete all duplicates such that each element appears only once. Return the linked list sorted as well. **(LeetCode)**

Source Code

```
class Solution:
   def deleteDuplicates(self, head: Optional[ListNode]) -> Optional[ListNode]:
        cur = head
        while cur and cur.next:
            if cur.val == cur.next.val:
                cur.next = cur.next.next
            else:
                cur = cur.next
        return head
```

Output



Accepted 76 / 76 testcases passed

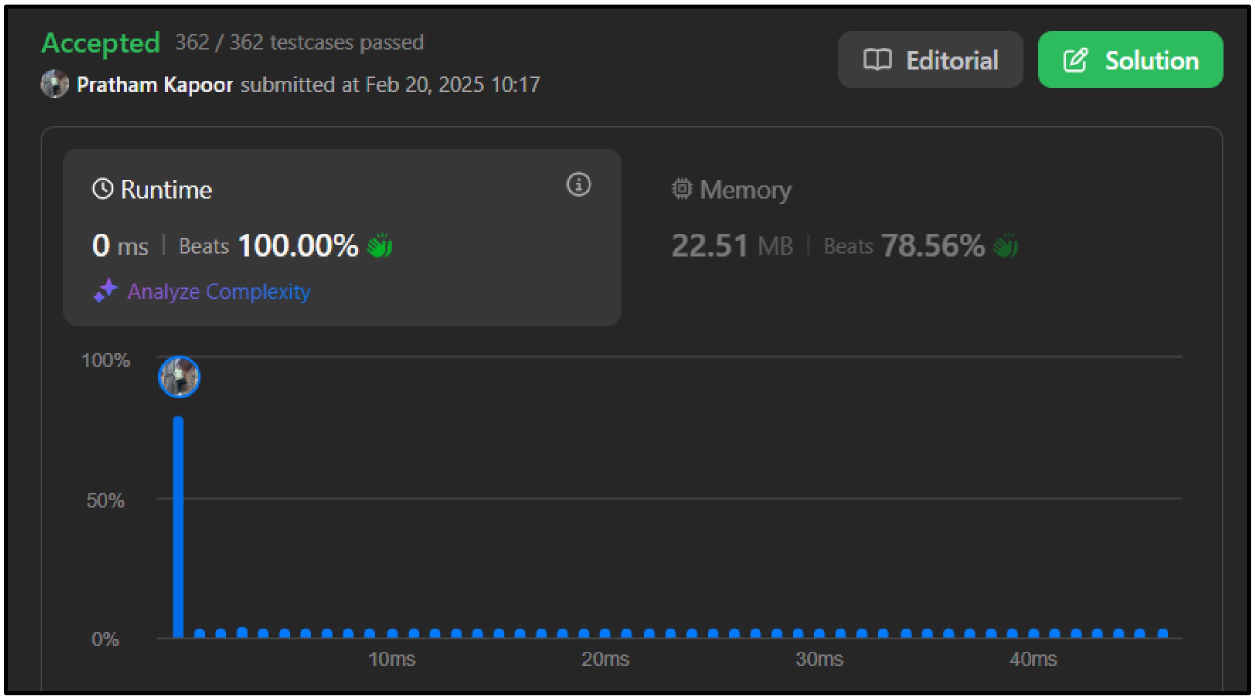
Pratham Kapoor submitted at Feb 20, 2025 15:56

Problem-3 :- Given the head of a singly linked list, reverse the list, and return the reversed list. **(LeetCode)**

Source Code

```
class Solution:
    def reverseList(self, head: ListNode) -> ListNode:
        dummy = ListNode()
        curr = head
        while curr:
            next = curr.next
            curr.next = dummy.next
            dummy.next = curr
            curr = next
        return dummy.next
```

Output


Accepted 362 / 362 testcases passed

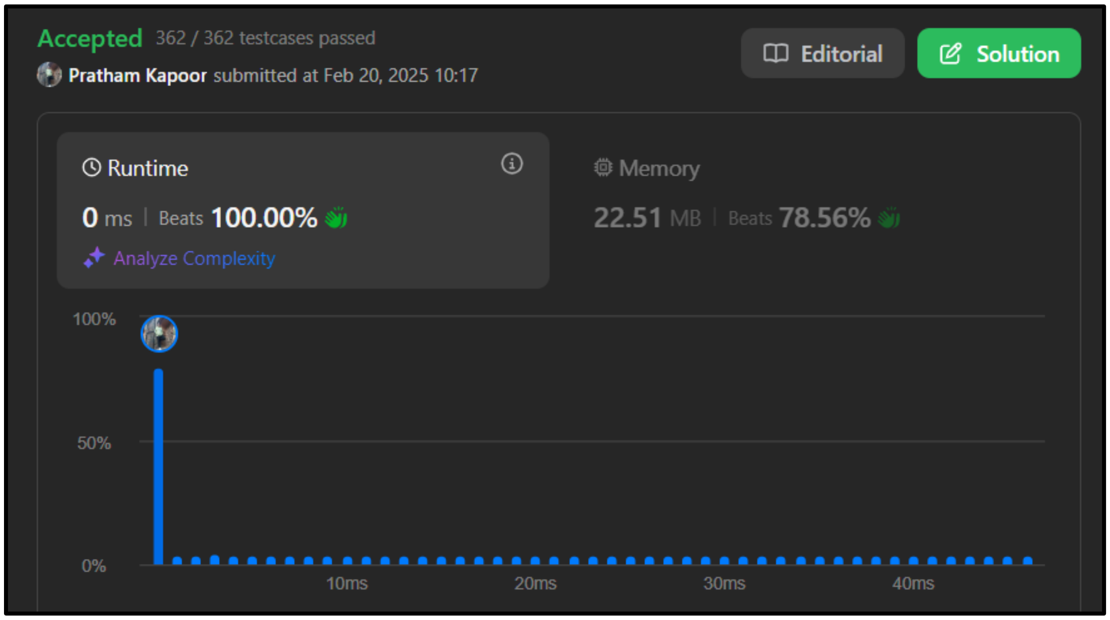
Pratham Kapoor submitted at Feb 20, 2025 10:17

Problem-4 :- Given You are given the head of a linked list. Delete the middle node, and return the head of the modified linked list. The middle node of a linked list of size n is the ⌊n / 2⌋th node from the start using 0-based indexing, where ⌊x⌋ denotes the largest integer less than or equal to x. For n = 1, 2, 3, 4, and 5, the middle nodes are 0, 1, 1, 2 and 2, respectively. **(LeetCode)**

Source Code

```
class Solution {
    public ListNode deleteMiddle(ListNode head) {
        ListNode dummy = new ListNode(0, head);
        ListNode slow = dummy, fast = head;
        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next; }
        slow.next = slow.next.next;
        return dummy.next; }
}
```

Output



Accepted 362 / 362 testcases passed
Pratham Kapoor submitted at Feb 20, 2025 10:17

Runtime
0 ms | Beats 100.00%

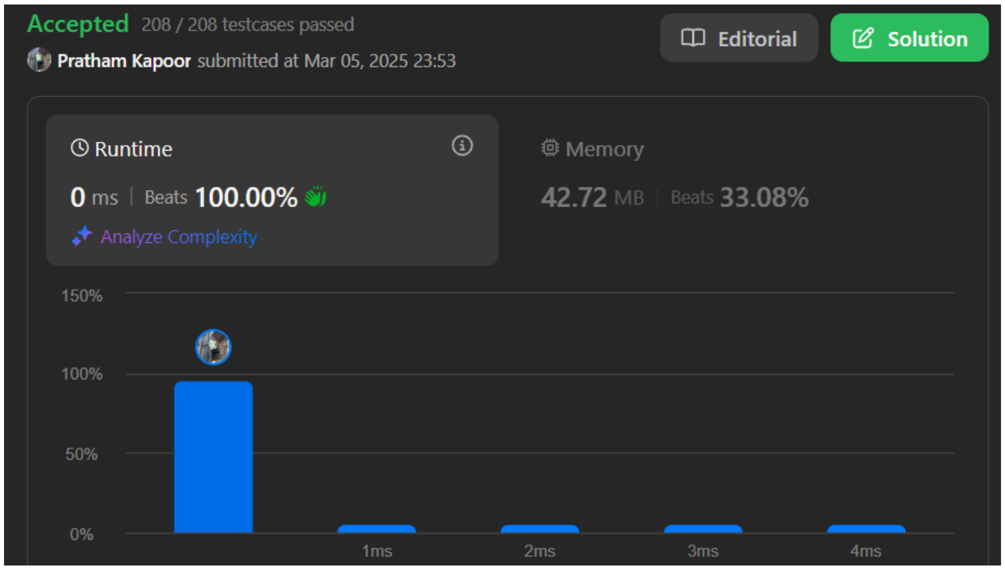
Memory
22.51 MB | Beats 78.56%

Problem-5 :- You are given the heads of two sorted linked lists list1 and list2. Merge the two lists into one sorted list. The list should be made by splicing together the nodes of the first two lists. Return the head of the merged linked list. **(LeetCode)**

Source Code

```
class Solution {
    public ListNode mergeTwoLists(ListNode list1, ListNode list2) {
        if (list1 == null) {
            return list2; }
        if (list2 == null) {
            return list1; }
        if (list1.val <= list2.val) {
            list1.next = mergeTwoLists(list1.next, list2);
            return list1;
        } else {
            list2.next = mergeTwoLists(list1, list2.next);
            return list2; }}}
```

Output


Accepted 208 / 208 testcases passed
Pratham Kapoor submitted at Mar 05, 2025 23:53

Runtime
0 ms | Beats 100.00%

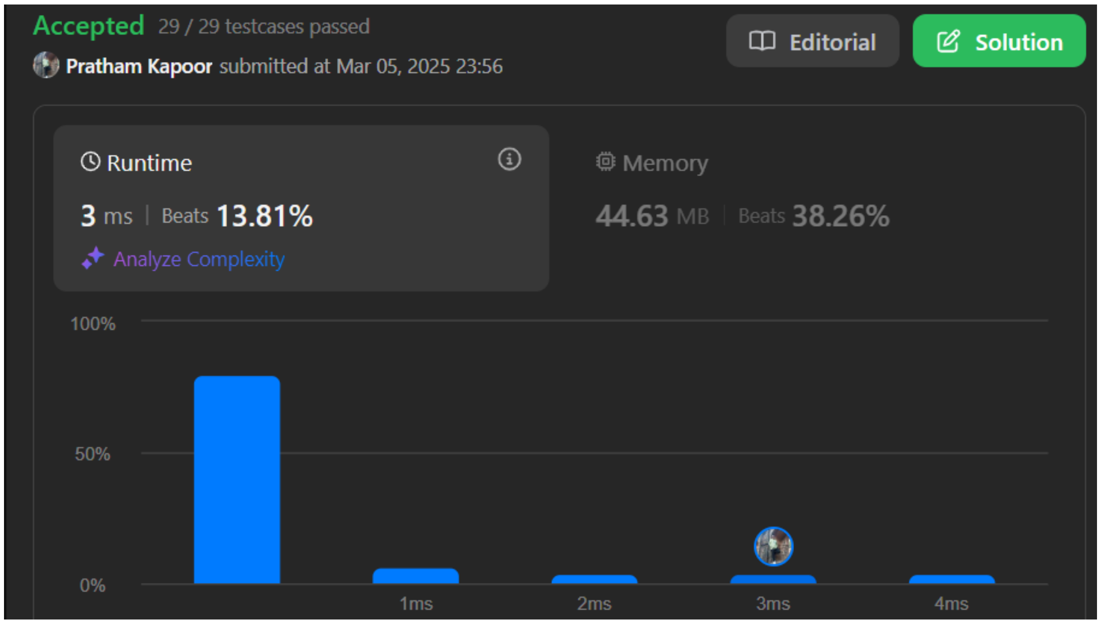
Memory
42.72 MB | Beats 33.08%

Problem-6 :- Given head, the head of a linked list, determine if the linked list has a cycle in it. There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to. Note that pos is not passed as a parameter. Return true if there is a cycle in the linked list. Otherwise, return false. **(LeetCode)**

Source Code

```
public class Solution {
    public boolean hasCycle(ListNode head) {
        Set<ListNode> s = new HashSet<>();
        for (; head != null; head = head.next) {
            if (!s.add(head)) {
                return true; }
        }
        return false; }
}
```

Output



Accepted 29 / 29 testcases passed

Pratham Kapoor submitted at Mar 05, 2025 23:56

Runtime

3 ms | Beats 13.81%

Memory

44.63 MB | Beats 38.26%

Problem-7 :- Given the head of a linked list, rotate the list to the right by k places. **(LeetCode)**

Source Code

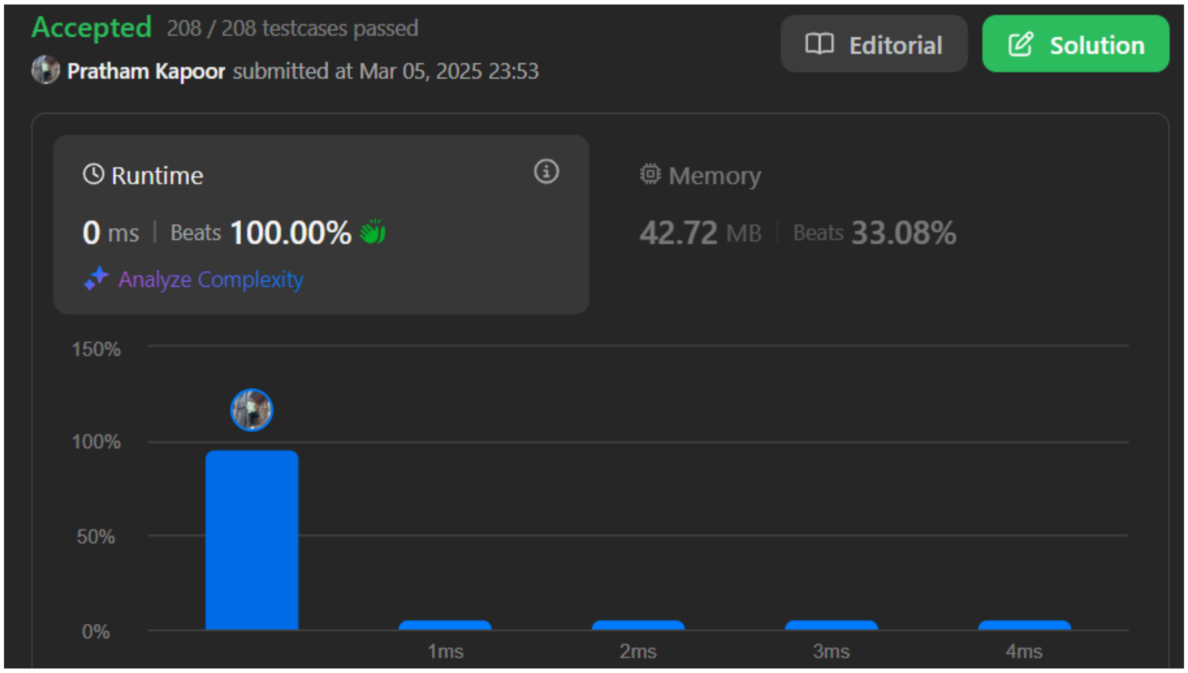
```
class Solution:
  def rotateRight(self, head: Optional[ListNode], k: int) -> Optional[ListNode]:
      if head is None or head.next is None:
         return head
      cur, n = head, 0

      while cur:
         n += 1
         cur = cur.next
      k %= n
      if k == 0:
         return head

      fast = slow = head
      for _ in range(k):
         fast = fast.next
      while fast.next:
         fast, slow = fast.next, slow.next

      ans = slow.next
      slow.next = None
      fast.next = head
      return ans
```

Output


Accepted 208 / 208 testcases passed

Pratham Kapoor submitted at Mar 05, 2025 23:53

Runtime

0 ms | Beats 100.00%

Memory

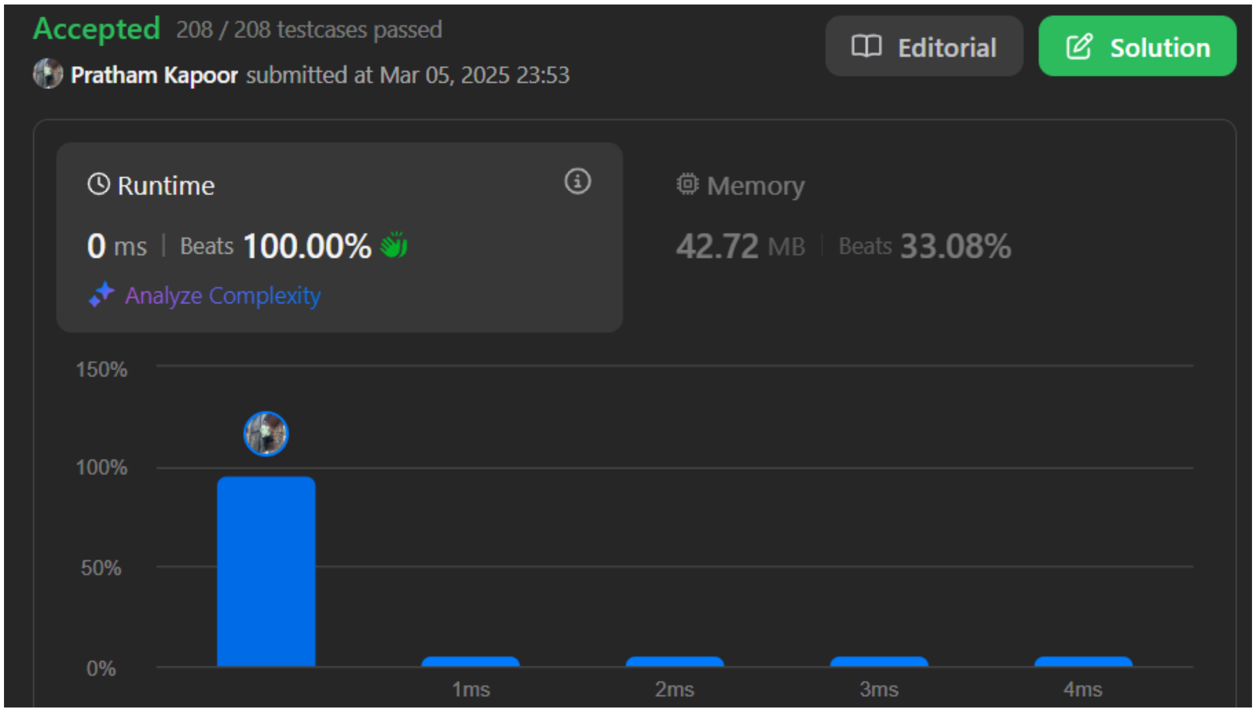
42.72 MB | Beats 33.08%

Problem-8 :- Given the head of a linked list, return the list after sorting it in ascending order. **(LeetCode)**

Source Code

```
class Solution {
    public ListNode sortList(ListNode head) {
        if (head == null || head.next == null) {
            return head;
        }
        ListNode slow = head, fast = head.next;
        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
        }
        ListNode l1 = head, l2 = slow.next;
        slow.next = null;
        l1 = sortList(l1);
        l2 = sortList(l2);
        ListNode dummy = new ListNode();
        ListNode tail = dummy;
        while (l1 != null && l2 != null) {
            if (l1.val <= l2.val) {
                tail.next = l1;
                l1 = l1.next;
            } else {
                tail.next = l2;
                l2 = l2.next; }
            tail = tail.next; }
        tail.next = l1 != null ? l1 : l2;
        return dummy.next; }
}
```

Output



Accepted 208 / 208 testcases passed

Pratham Kapoor submitted at Mar 05, 2025 23:53

Runtime

0 ms | Beats 100.00%

Memory

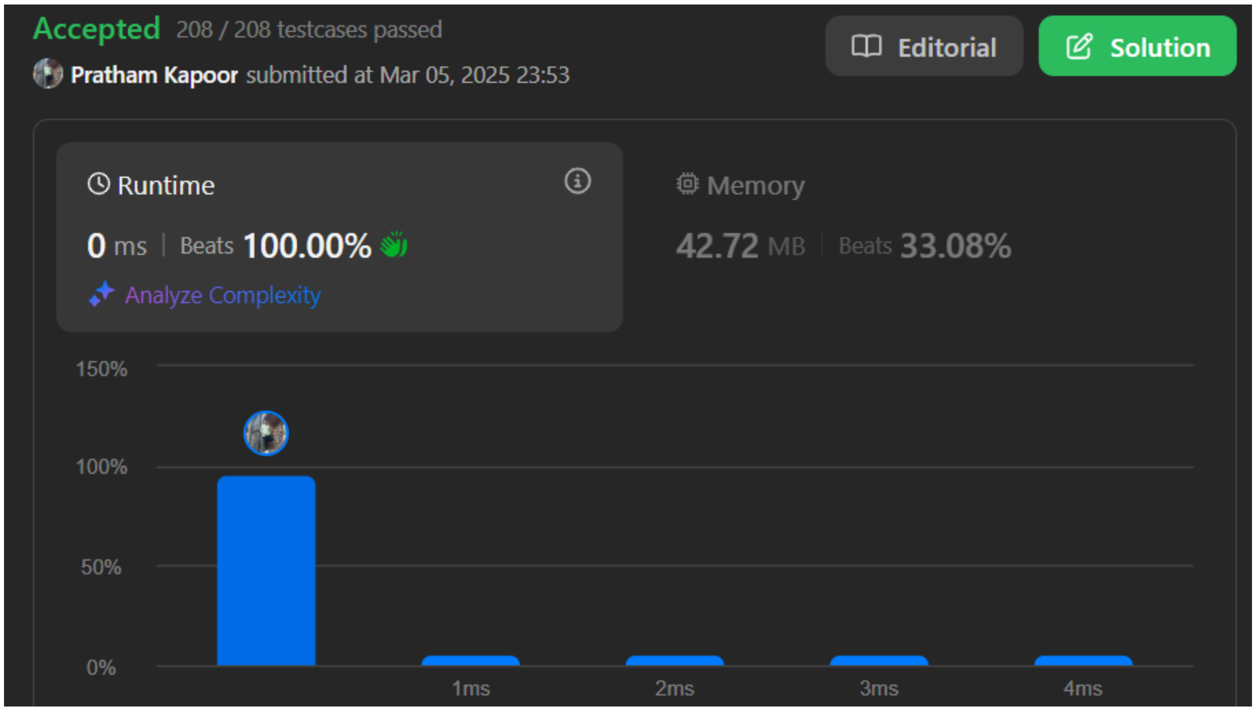
42.72 MB | Beats 33.08%

Problem-9 :- You are given an array of k linked-lists lists, each linked-list is sorted in ascending order. Merge all the linked-lists into one sorted linked-list and return it. **(LeetCode)**

Source Code

```
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)> pq;
        for (auto head : lists) {
            if (head) {
                pq.push(head);
            }
        }
        ListNode* dummy = new ListNode();
        ListNode* cur = dummy;
        while (!pq.empty()) {
            ListNode* node = pq.top();
            pq.pop();
            if (node->next) {
                pq.push(node->next);
            }
            cur->next = node;
            cur = cur->next;
        }
        return dummy->next;
    }
};
```

Output



Accepted 208 / 208 testcases passed

Pratham Kapoor submitted at Mar 05, 2025 23:53

Runtime

0 ms | Beats 100.00%

Memory

42.72 MB | Beats 33.08%