AP ASSIGNMENT

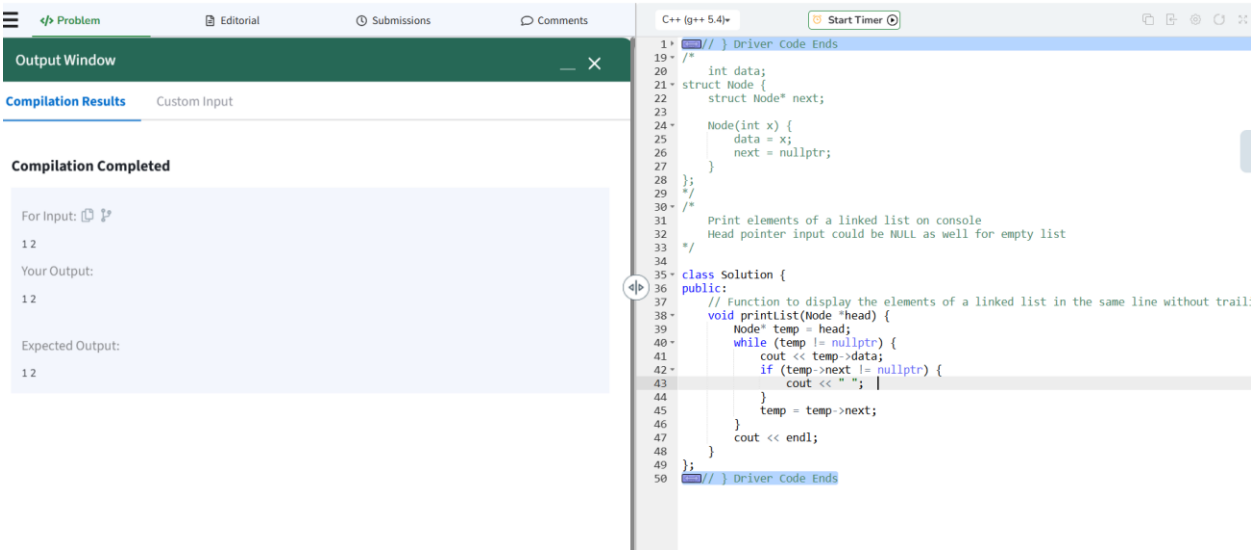
Name: Swatantra

UID: 22BCS12725

Section: 612-“B”

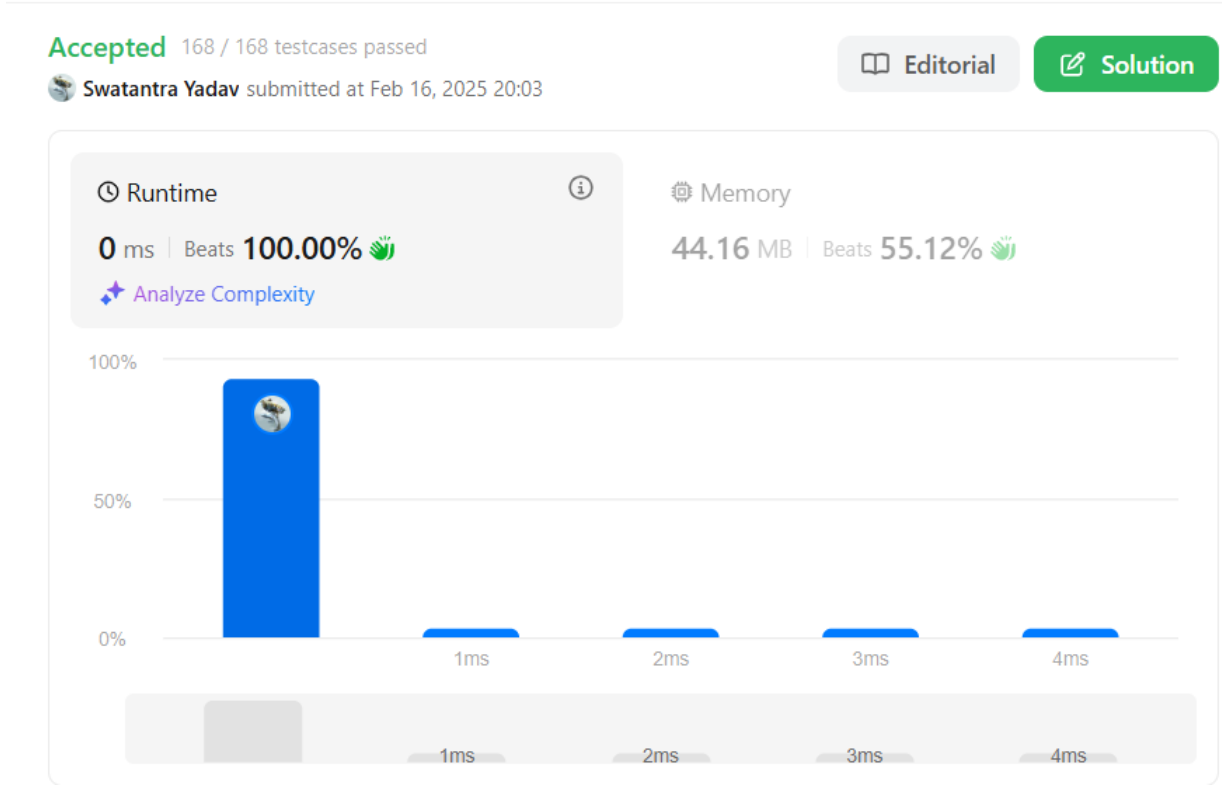
Print linked list – GFG

```
class Solution {
public:
    // Function to display the elements of a linked list in the same line without trailing space
    void printList(Node *head) {
        Node* temp = head;
        while (temp != nullptr) {
cout<< temp->data;
            if (temp->next != nullptr) {
cout<< " ";
            }
            temp = temp->next;
        }
cout<<endl;  }
};
```



Remove duplicates from a linkedlist

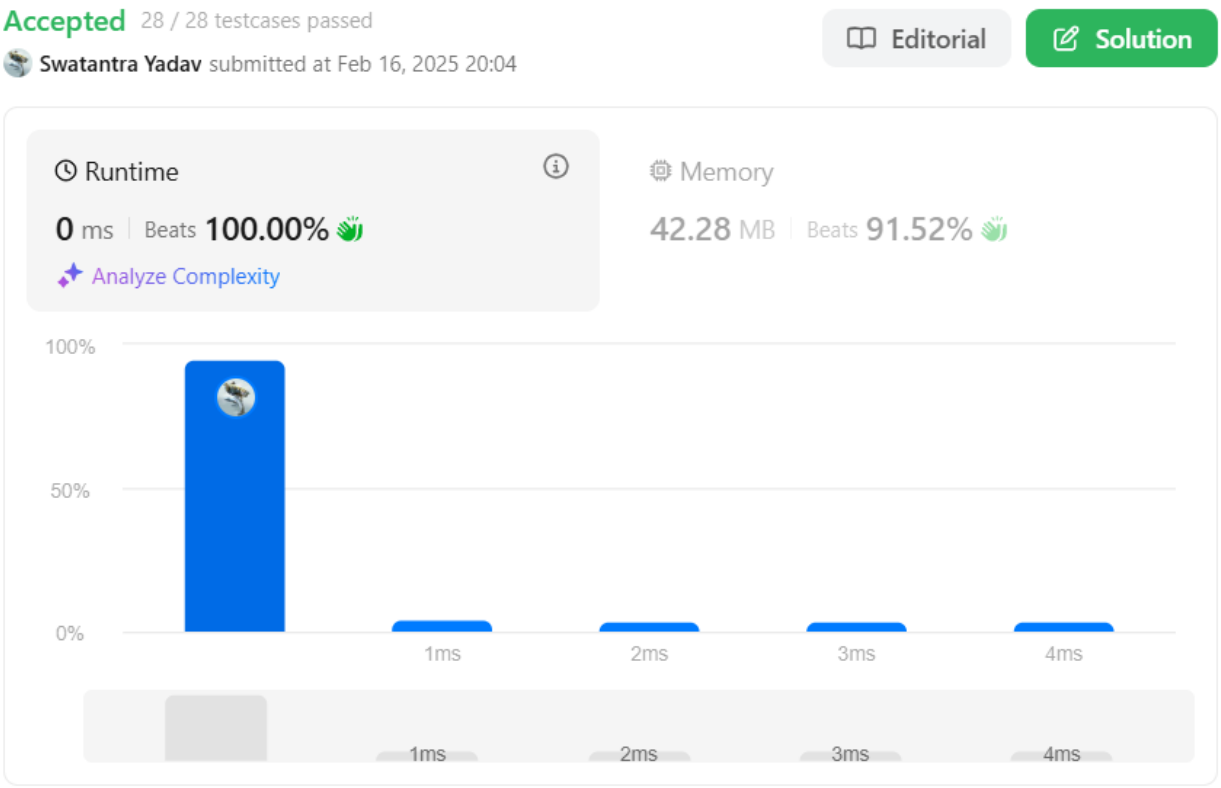
```
class Solution {
  public ListNode deleteDuplicates(ListNode head) {
      ListNode current = head;

      while (current != null && current.next != null) {
         if (current.val == current.next.val) {
            current.next = current.next.next;
         } else {
            current = current.next;
         }
      }

      return head;
   }
}
```



Reverse a linked list

```
class Solution {
    public ListNode reverseList(ListNode head) {
        ListNode node = null;

        while (head != null) {
            ListNode temp = head.next;
            head.next = node;
            node = head;
            head = temp;
        }

        return node;
```

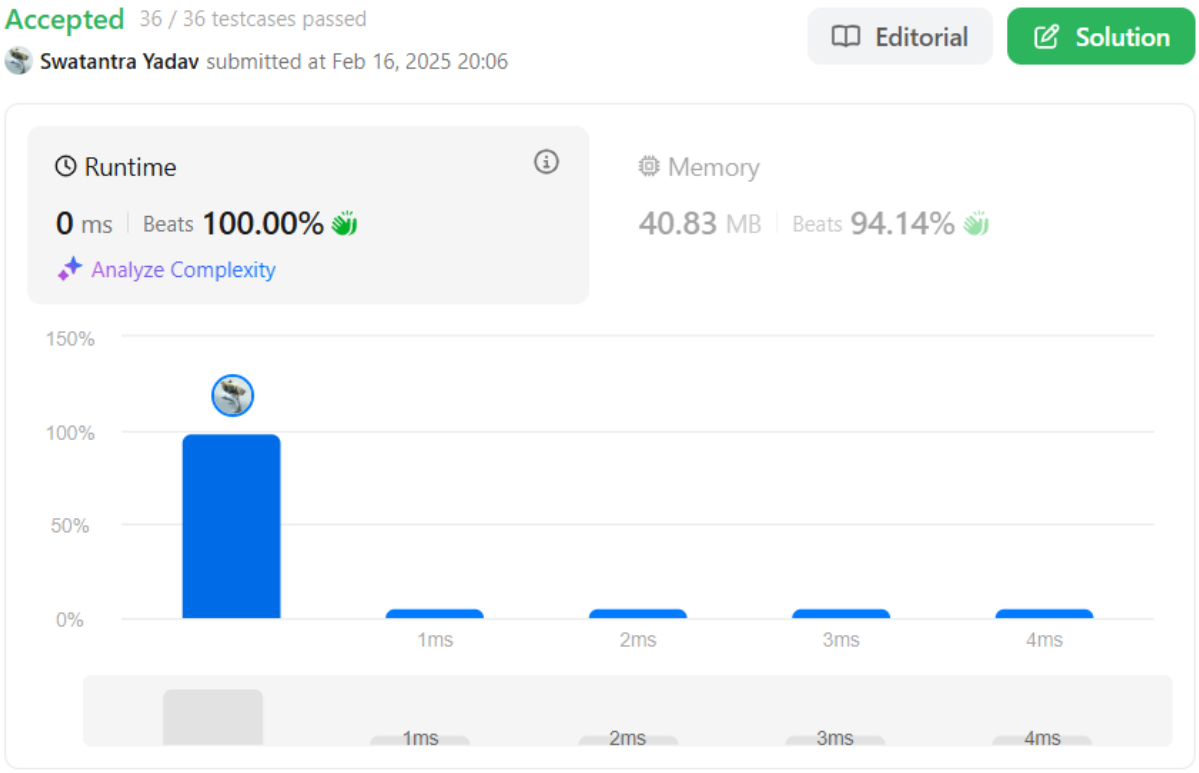
```
    }
}    return prev;
    }
```



Delete middle node of a list

```
class Solution {
    public ListNode middleNode(ListNode head) {
        ListNode slow = head;
        ListNode fast = head;
        while(fast!=null && fast.next!=null){
            slow = slow.next;
            fast = fast.next.next;
        }
        return slow;
    }
```

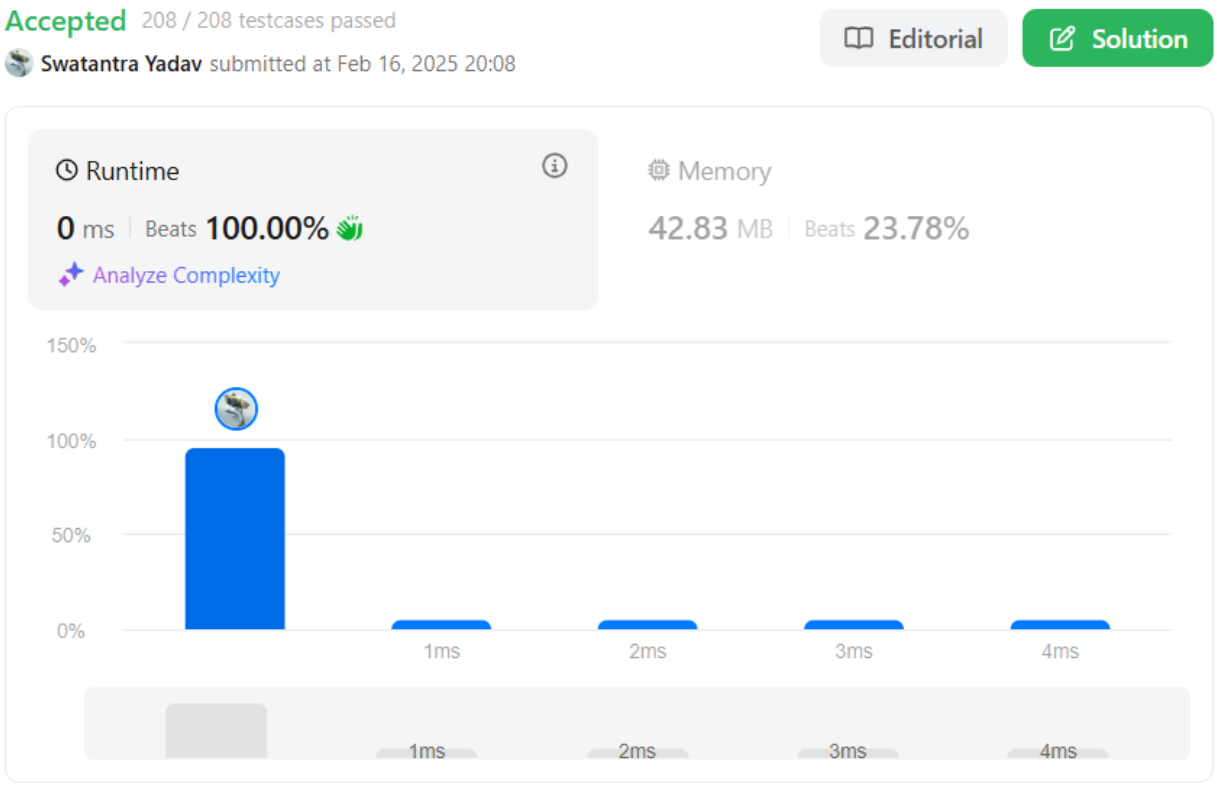
}



Merge two sorted linked lists

```
class Solution {
    public ListNode mergeTwoLists(ListNode list1, ListNode list2) {

        if(list1!=null && list2!=null){
        if(list1.val<list2.val){
            list1.next=mergeTwoLists(list1.next,list2);
            return list1;
            }
```

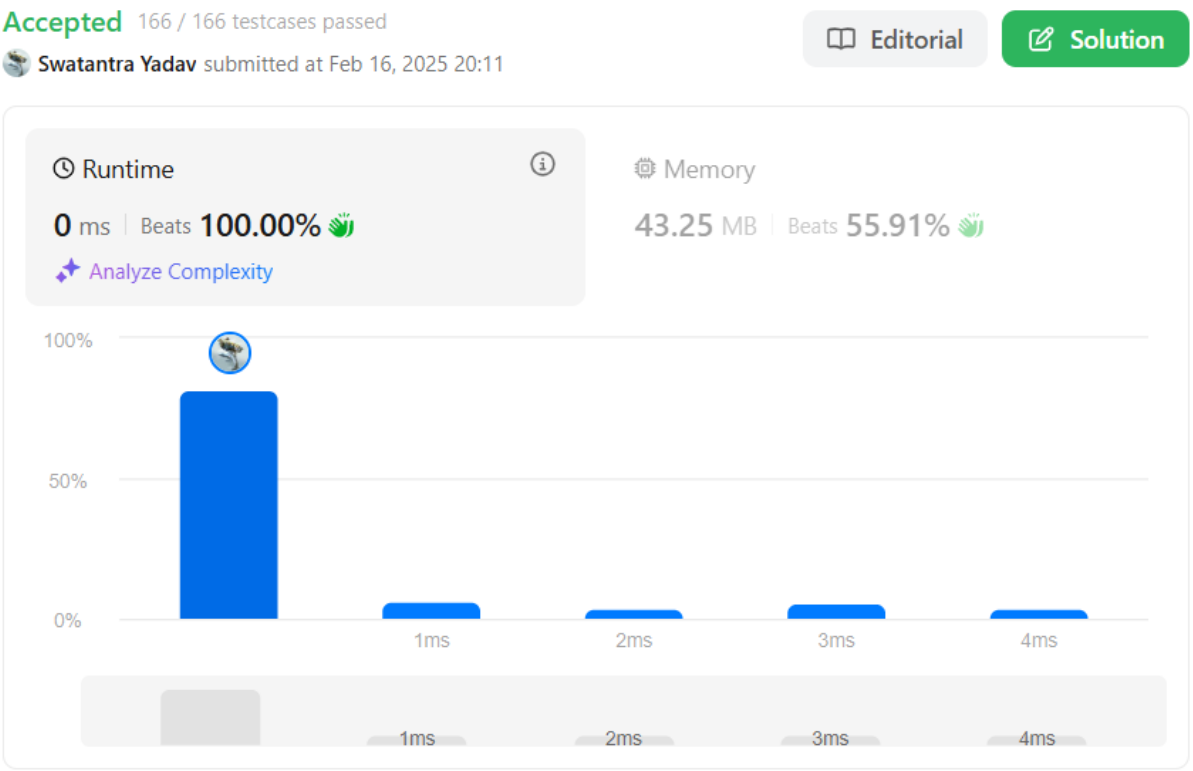
```
        else{
            list2.next=mergeTwoLists(list1,list2.next);
            return list2;
        }
        }
        if(list1==null)
            return list2;
        return list1;
    }
}
```



Remove duplicates from sorted lists 2

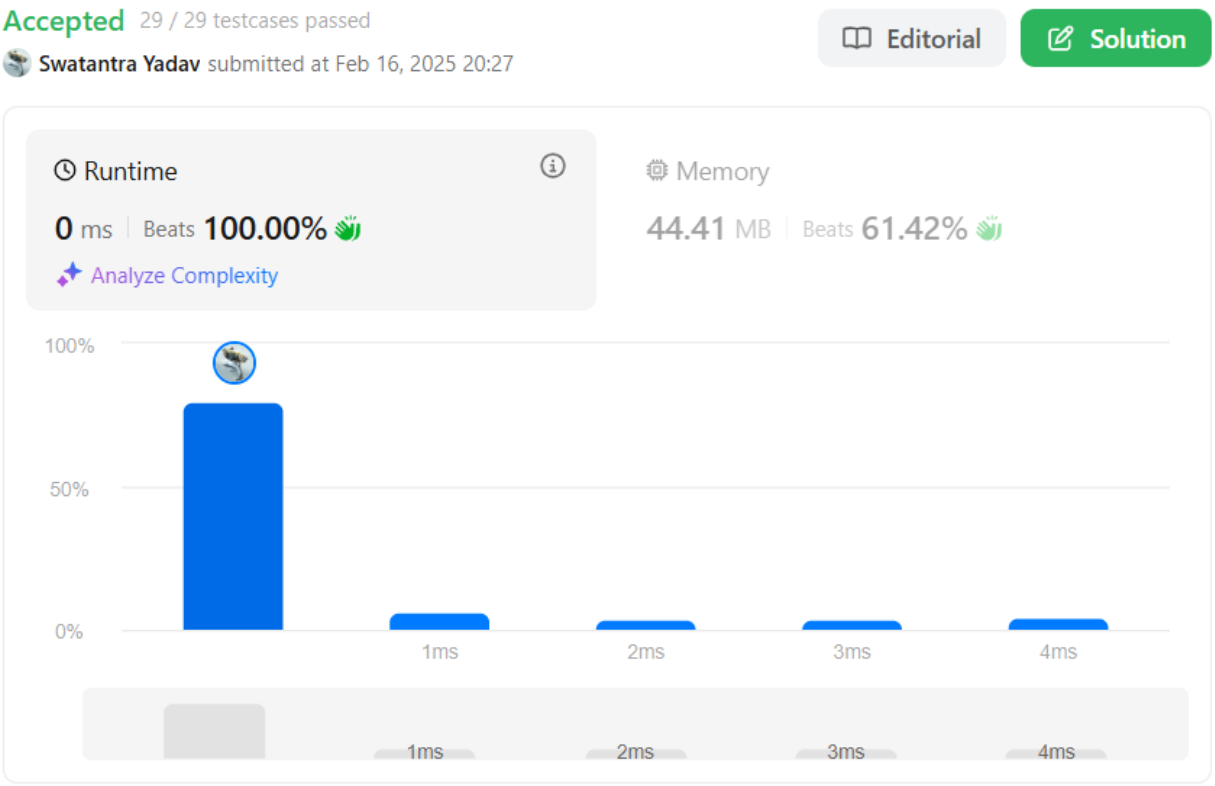
```
class Solution {
    public ListNode deleteDuplicates(ListNode head) {
        if (head == null || head.next == null) return head;
```

```
        ListNode res = new ListNode(0,head);
        ListNode prev = res;

        while(head != null && head.next != null){
            if(head.next.val == head.val){
                while(head.next != null && head.next.val == head.val){
                    head = head.next;
                }
                prev.next = head.next;
            }else{
                prev = prev.next;
            }
            head = head.next;
        }
        return res.next;
    }
}
```



Detect a cycle in a linked list

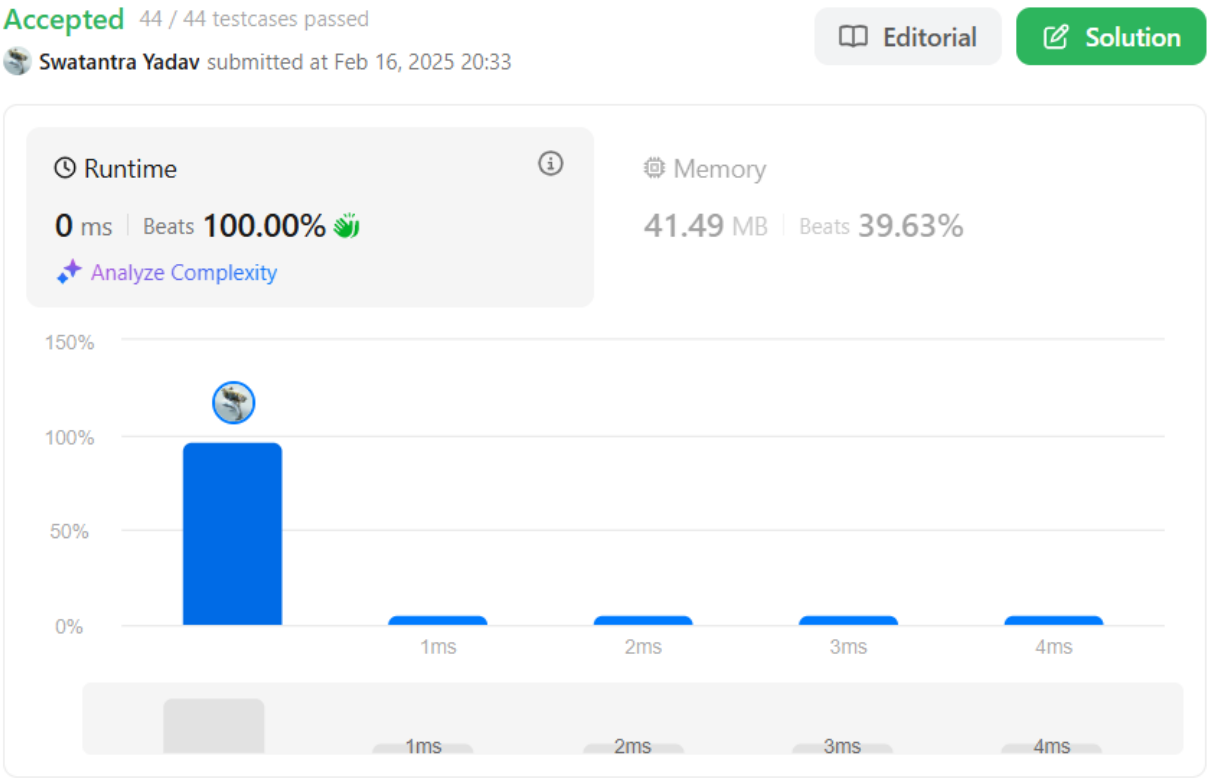
```
public class Solution {
    public boolean hasCycle(ListNode head) {
        ListNode slow = head, fast = head;
        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
            if (slow == fast) return true;
        }
        return false;
    }
}
```



Reverse linked list 2

```
class Solution {
    public ListNode reverseBetween(ListNode head, int left, int right) {
        ArrayList<Integer> list = new ArrayList<>();

        while(head != null){
            list.add(head.val);
            head = head.next;
        }

        int i = left-1, j = right-1;
        while(i < j){
            int temp = list.get(i);
            list.set(i,list.get(j));
            list.set(j,temp);
```

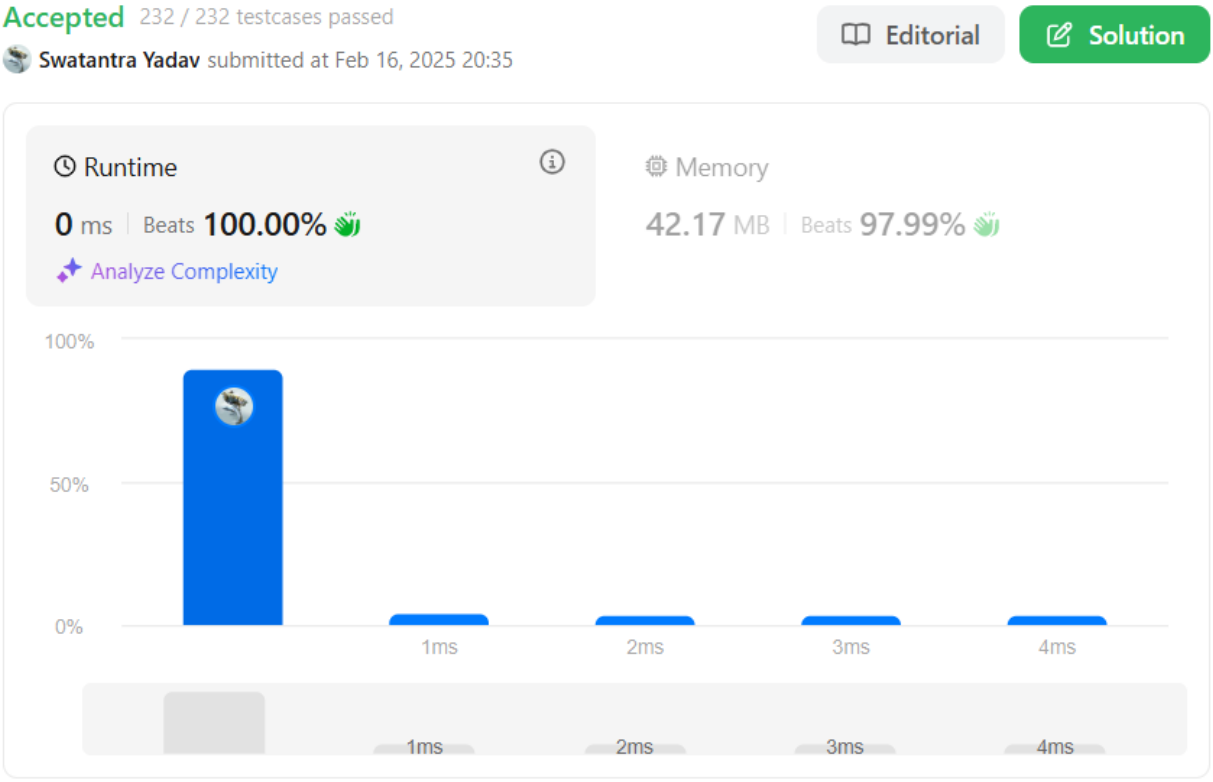
```
            i++;
            j--;
        }

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

        for(int ind = 0;ind<list.size();ind++){
            curr.next = new ListNode(list.get(ind));
            curr = curr.next;
        }
        return dummy.next;
    }
}
```



rotate a list

```
class Solution {
    public ListNode rotateRight(ListNode head, int k) {
        if (head == null || head.next == null || k == 0) {
            return head;
        }

        int length = 1;
        ListNode temp = head;

        while (temp.next != null) {
            temp = temp.next;
            length++;
        }

        temp.next = head;
        k = k % length;
        k = length - k;

        while (k-- > 0) {
            temp = temp.next;
        }

        head = temp.next;
        temp.next = null;

        return head;
    }
```

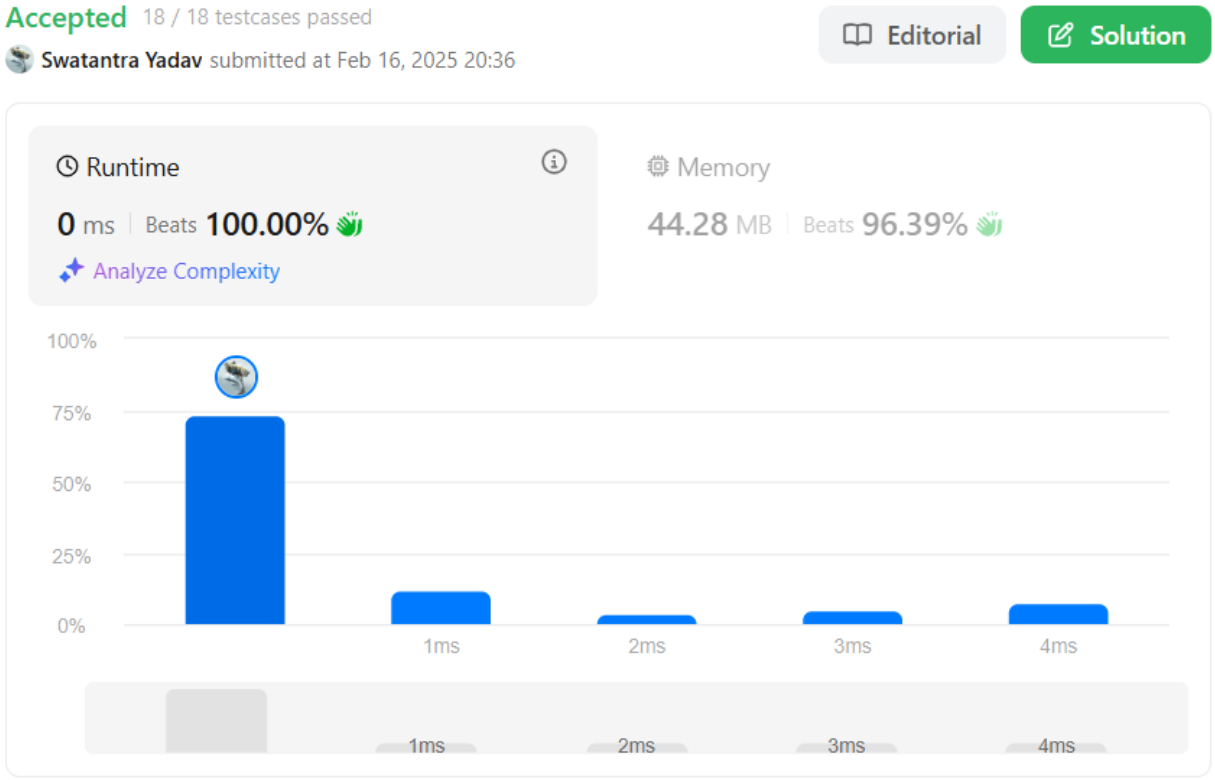
}



Detect a cycle in a linked list 2

```
public class Solution {
    public ListNode detectCycle(ListNode head) {
        ListNode slow = head, fast = head;
        while (fast != null && fast.next != null) {
            slow = slow.next;
            fast = fast.next.next;
            if (slow == fast) break;
        }
        if (fast == null || fast.next == null) return null;
        while (head != slow) {
            head = head.next;
            slow = slow.next;
```

```
        }
        return head;
    }
}
```


Accepted
18 / 18 testcases passed
Swatantra Yadav submitted at Feb 16, 2025 20:36
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
44.28 MB | Beats 96.39%
100%
75%
50%
25%
0%
1ms
2ms
3ms
4ms