

**Name : Navnidhi**

**UID : 22BCS15723**

**SECTION : IOT 622 B**

**DATE : 06/03/2025**

**SUBJECT : ADVANCED PROGRAMMING – 2**

**1 Print Linked List:**

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

```
1 // } Driver Code Ends
19 /*
20 struct Node {
21     int data;
22     struct Node* next;
23
24     Node(int x) {
25         data = x;
26         next = nullptr;
27     }
28 };
29 */
30 /*
31     Print elements of a linked list on console
32     Head pointer input could be NULL as well for empty list
33 */
34
35 class Solution {
36 public:
37     // Function to display the elements of a linked list in same line
38     void printList(Node *head) {
39         Node* temp = head;
40         while (temp) {
41             cout << temp->data<<" ";
42             temp = temp->next;
43         }
44     }
45 }
46 }
```

## 2 Remove duplicates from a sorted list:

```
class Solution {
```

```
public:
```

```
ListNode* deleteDuplicates(ListNode* head) {
```

```
    if (head == nullptr) {
```

```
        return head;
```

```
    }
```

```
    ListNode* current = head;
```

```
    while (current != nullptr && current->next != nullptr) {
```

```

        if (current->val == current->next->val) {
            current->next = current->next->next;
        } else {
            current = current->next;
        }
    }

    return head;
}
};

```

	Description	Editorial	Solutions	Submissions			
	Status ▾	Language ▾	Runtime	Memory	Notes		
1	Accepted Feb 17, 2025	C++	0 ms	15.9 MB			

### 3 Reverse a linked list:

```

class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        // Initialize pointers
        ListNode* prev = nullptr; // Previous node starts as NULL
        ListNode* next = nullptr; // Next node
        ListNode* curr = head;    // Current node starts at the head
    }
};

```

```
// Traverse the list
while (curr != nullptr) {
    // Save the next node
    next = curr->next;

    // Reverse the link
    curr->next = prev;

    // Move pointers forward
    prev = curr; // Move prev to the current node
    curr = next; // Move curr to the next node
}

// prev is now the new head of the reversed list
return prev;
}
};
```

Problem List
<
>
🔗

📄 Description
📖 Editorial
🧪 Solutions
🕒 Accepted
✕
🕒 Submissions

← All Submissions

Accepted
28 / 28 testcases passed

👤 Navnidhi Sharma
submitted at Mar 06, 2025 20:59

📖 Editorial
✍️ Solution

Code
C++

```

class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        // Initialize pointers
        ListNode* prev = nullptr; // Previous node starts as NULL
        ListNode* next = nullptr; // Next node
        ListNode* curr = head;    // Current node starts at the head
    }
}

```

⌵ View more

#### 4 Delete middle node of a 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 {

```

```

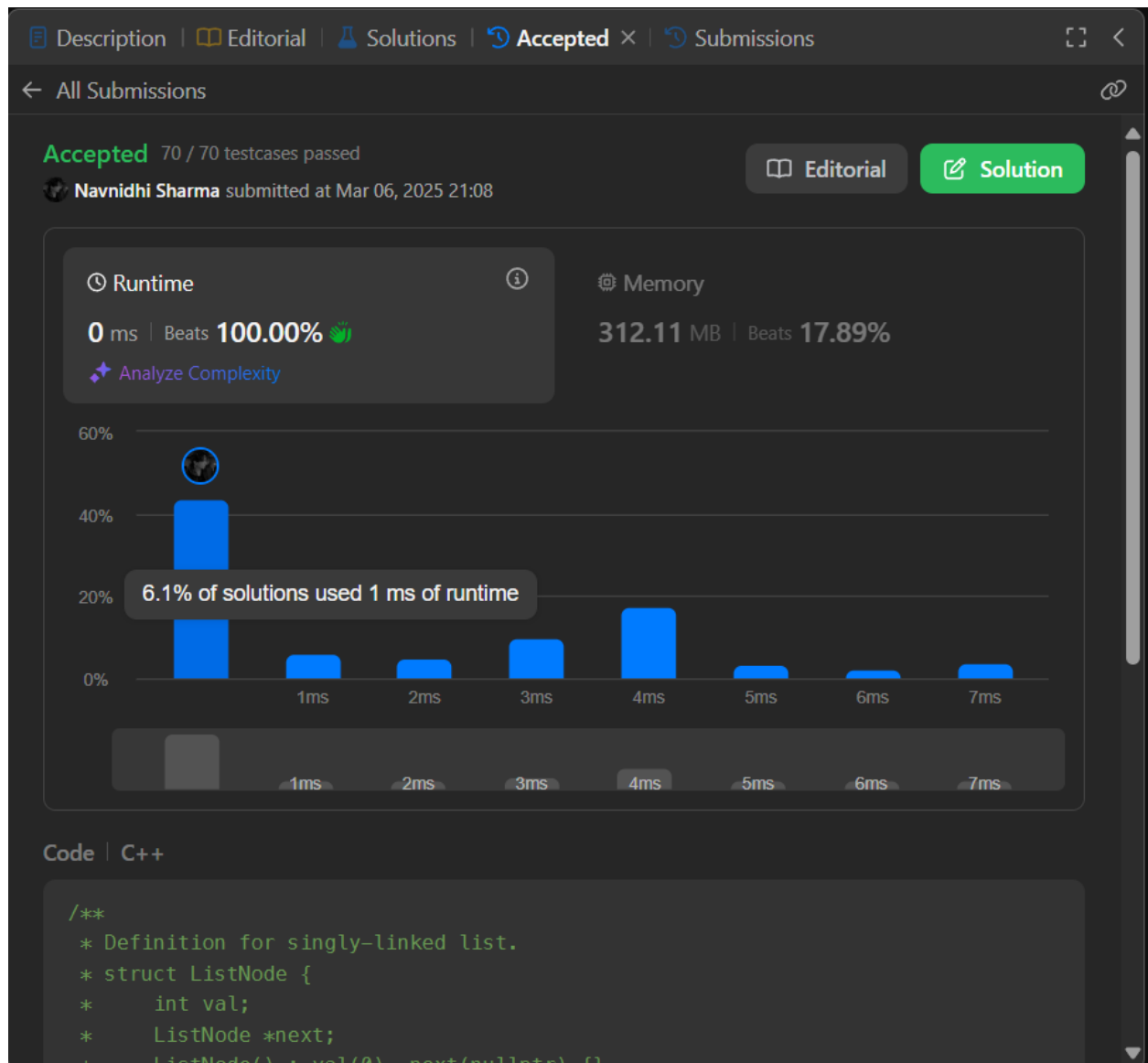
public:

```

```

    ListNode* deleteMiddle(ListNode* head) {
        if(head==NULL || head->next==NULL) return NULL;
        ListNode* slow=head;
        ListNode* fast=head->next;
        while(fast->next!=NULL && fast->next->next!=NULL)
        {
            slow=slow->next;
            fast=fast->next->next;
        }
        slow->next=slow->next->next;
        return head;
    }
};

```



## 5 Merge two sorted linked lists:

```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
        if(list1 == NULL || list2 == NULL){
            return list1 == NULL ? list2 : list1;
        }
    }
};
```

```
}

if(list1->val <= list2->val){
    list1->next = mergeTwoLists(list1->next, list2);
    return list1;
}
else{
    list2->next = mergeTwoLists(list1, list2->next);
    return list2;
}
};
```



Problem List < > 🔍

Description | Editorial | Solutions | Accepted × | Submissions

← All Submissions

**Accepted** 208 / 208 testcases passed

Navnidhi Sharma submitted at Mar 06, 2025 21:11

Editorial Solution

**Runtime** 0 ms | Beats 100.00% 🏆  
Analyze Complexity

**Memory** 19.37 MB | Beats 86.51% 🏆

100%  
50%  
0%

0.99% of solutions used 2 ms of runtime

1ms 2ms 3ms 4ms

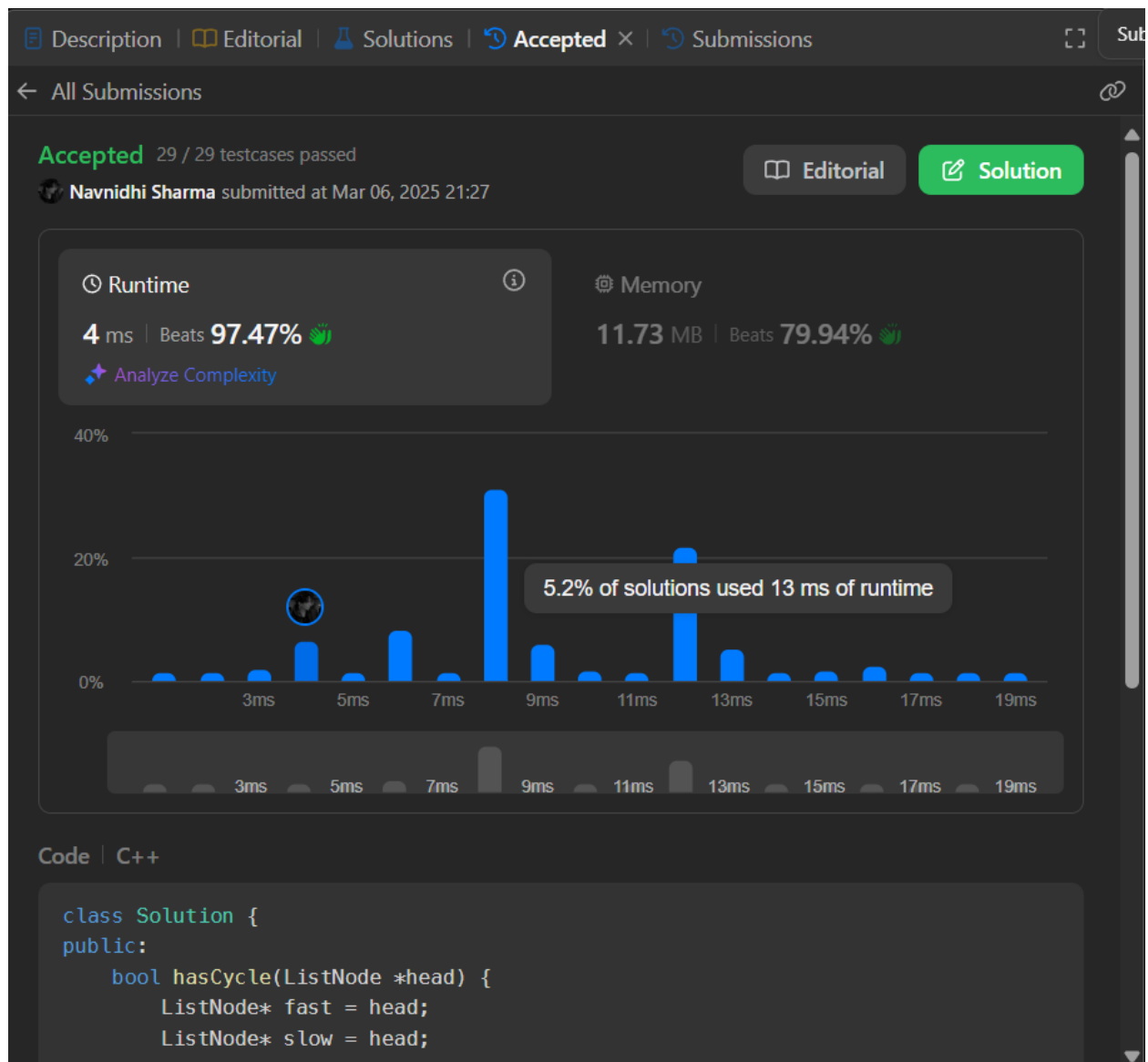
Code | C++

```
class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
        if(list1 == NULL || list2 == NULL){
            return list1 == NULL ? list2 : list1;
```

## 6 Detect a cycle in a linked list:

```
class Solution {
public:
    bool hasCycle(ListNode *head) {
        ListNode* fast = head;
        ListNode* slow = head;
```

```
while (fast != nullptr && fast->next != nullptr) {  
    fast = fast->next->next;  
    slow = slow->next;  
  
    if (fast == slow) {  
        return true;  
    }  
}  
  
return false;  
}  
};
```



## 7 Rotate a list:

/\*\*

\* Definition for singly-linked list.

\* struct ListNode {

\* int val;

\* ListNode \*next;

\* ListNode() : val(0), next(nullptr) {}

\* ListNode(int x) : val(x), next(nullptr) {}

```

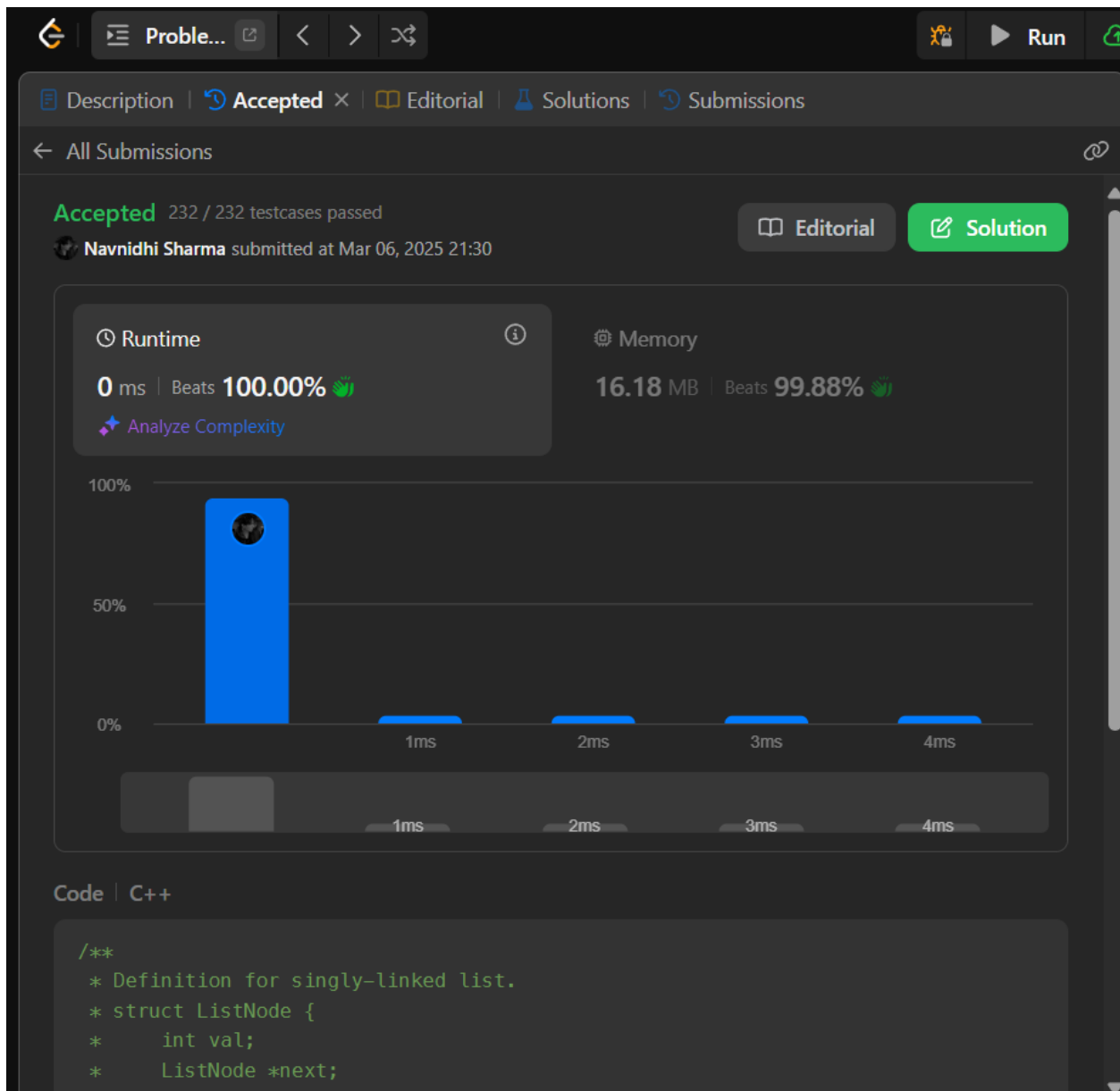
*   ListNode(int x, ListNode *next) : val(x), next(next) {}
* };
*/

class Solution {
public:
    ListNode* rotateRight(ListNode* head, int k) {
        // base condition
        if(head==NULL || head->next==NULL || k==0) return head;

        ListNode* curr=head;
        int count=1;
        while(curr->next!=NULL){
            curr=curr->next;
            count++;
        }
        curr->next=head;
        k=count-(k%count);
        while(k-->0){
            curr=curr->next;
        }
        head=curr->next;
        curr->next=NULL; // curr points to tail node sorta

        return head;
    }
};

```



## 8 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) {}
 * };
 */
```

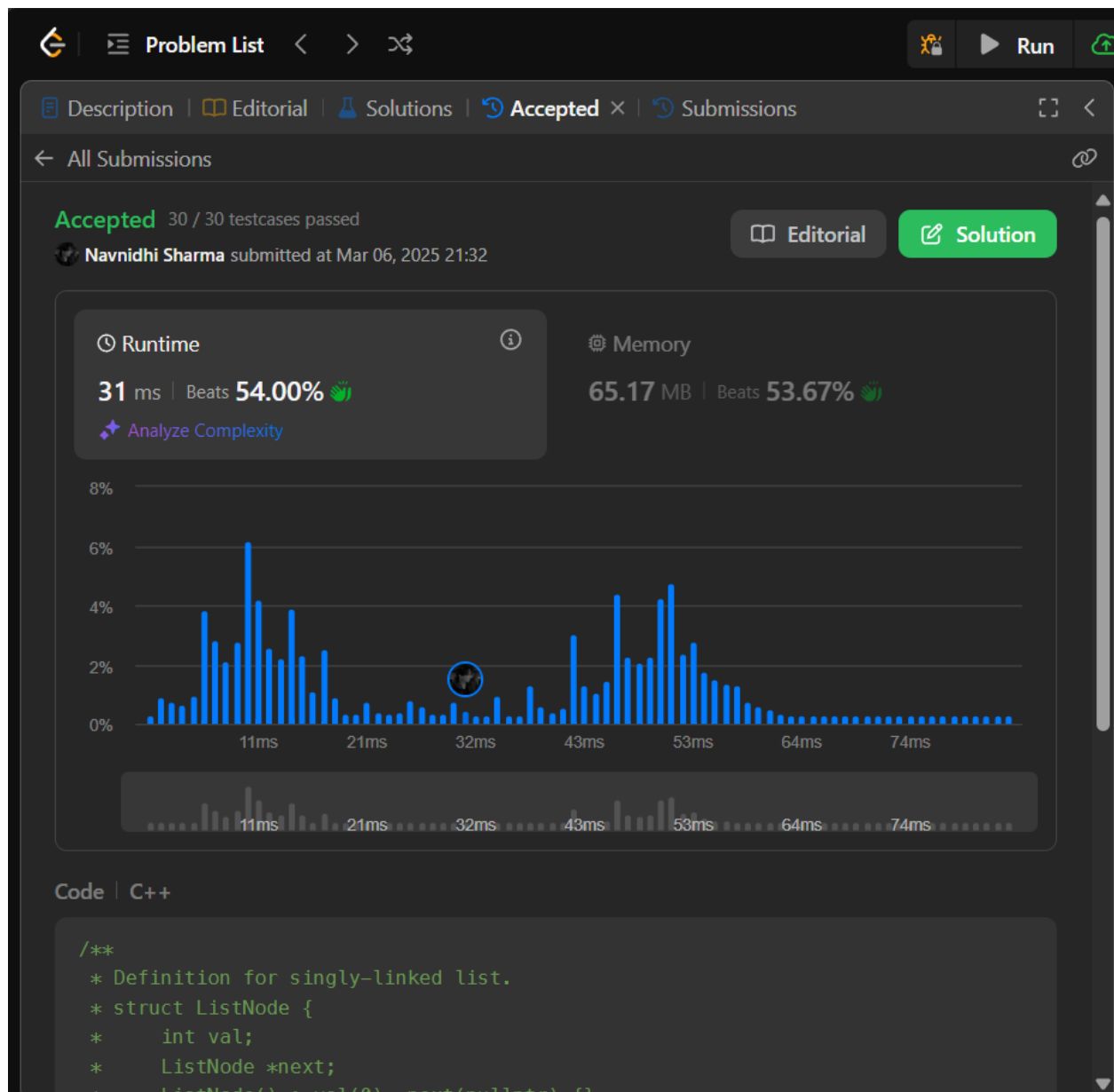
```

*   ListNode(int x) : val(x), next(nullptr) {}
*   ListNode(int x, ListNode *next) : val(x), next(next) {}
* };
*/

class Solution {
public:
    ListNode* sortList(ListNode* head) {
        if(head==NULL) return NULL;
        ListNode *damy=NULL,*current=NULL;
        vector<int>v;
        while(head){
            v.push_back(head->val);
            head=head->next;
        }
        sort(v.begin(),v.end());
        for(auto u:v){
            ListNode *tem = new ListNode(u);
            if(damy==NULL){
                damy=tem;
                current=tem;
            }
            else{
                current->next=tem;
                current=current->next;
            }
        }
        return damy;
    }
}

```

```
};
```



## 9 Merge k sorted lists

```
/**
 * 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* mergeKLists(vector<ListNode*>& lists) {
        priority_queue<pair<int,ListNode*>, vector<pair<int, ListNode*>>,
greater<pair<int,ListNode*>>> pq;

        //Pushing all the head of linked lists in priority queue.
        for(int i=0;i<lists.size();i++){
            if(lists[i])  pq.push({lists[i]->val , lists[i]});
        }
        ListNode* dummy = new ListNode(-1);
        ListNode* temp = dummy;

        while(!pq.empty()){
            auto it = pq.top();
            if(it.second->next){
                pq.push({it.second->next->val,it.second->next});
            }
            pq.pop();
            temp->next = it.second;
            temp=temp->next;
        }
        return dummy->next;
    }
};

```





Problem List



Run



Description



Editorial



Solutions



Accepted



Submissions



← All Submissions



**Accepted** 134 / 134 testcases passed

Navnidhi Sharma submitted at Mar 06, 2025 21:33



Editorial



Solution

⌚ Runtime

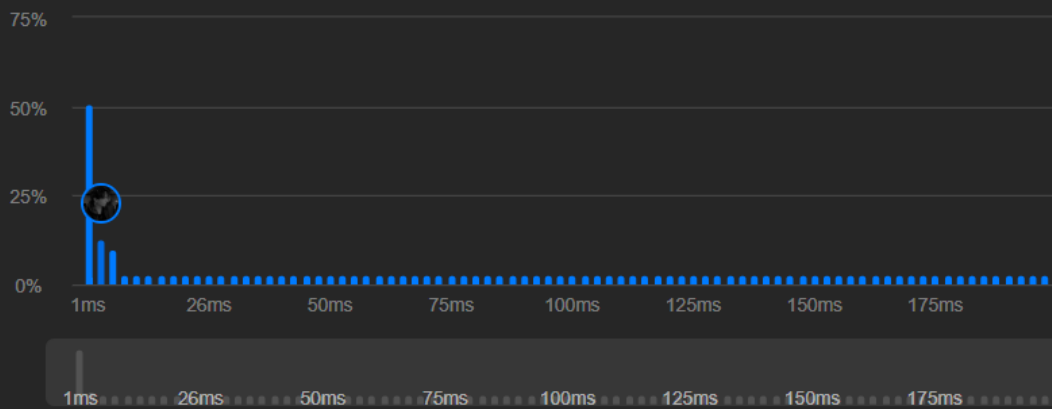


3 ms | Beats **64.88%** 🌱

🔮 Analyze Complexity

⚙ Memory

18.52 MB | Beats **50.86%** 🌱



Code | C++

```
/**
 * 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) {}
 * }
```