



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Assignment 1

Student Name: Abhay Kejriwal

Branch: BE-CSE

Semester: 6th

Subject Name: Advanced Programming

UID: 22BCS14663

Section/Group: IOT-601-B

Date of Performance: 22/07/24

Subject Code: 22CSP – 351

DAY-1:

1. Remove Duplicates From The Sorted Array-

```
int removeDuplicates(vector<int>& nums) {  
    int k=0;int i=0;  
    int size=nums.size();  
    for(int j=1;j<size;j++){  
        if(nums[i]!=nums[j]){  
            nums[i+1]=nums[j];  
            i++;  
        }  
    }  
    return i+1;  
}
```

The screenshot shows a web browser displaying a LeetCode submission page for the problem "Remove Duplicates from Sorted Array". The submission is by a user named "Abhay..." and was submitted on Feb 05, 2025, at 09:47. The submission status is "Accepted", with 362 / 362 testcases passed. The runtime is 0 ms, which beats 100.00% of other submissions. The memory usage is 22.61 MB, which beats 50.97% of other submissions. The code is written in C++ and implements the two-pointer approach. The test case input is [1, 1, 2].

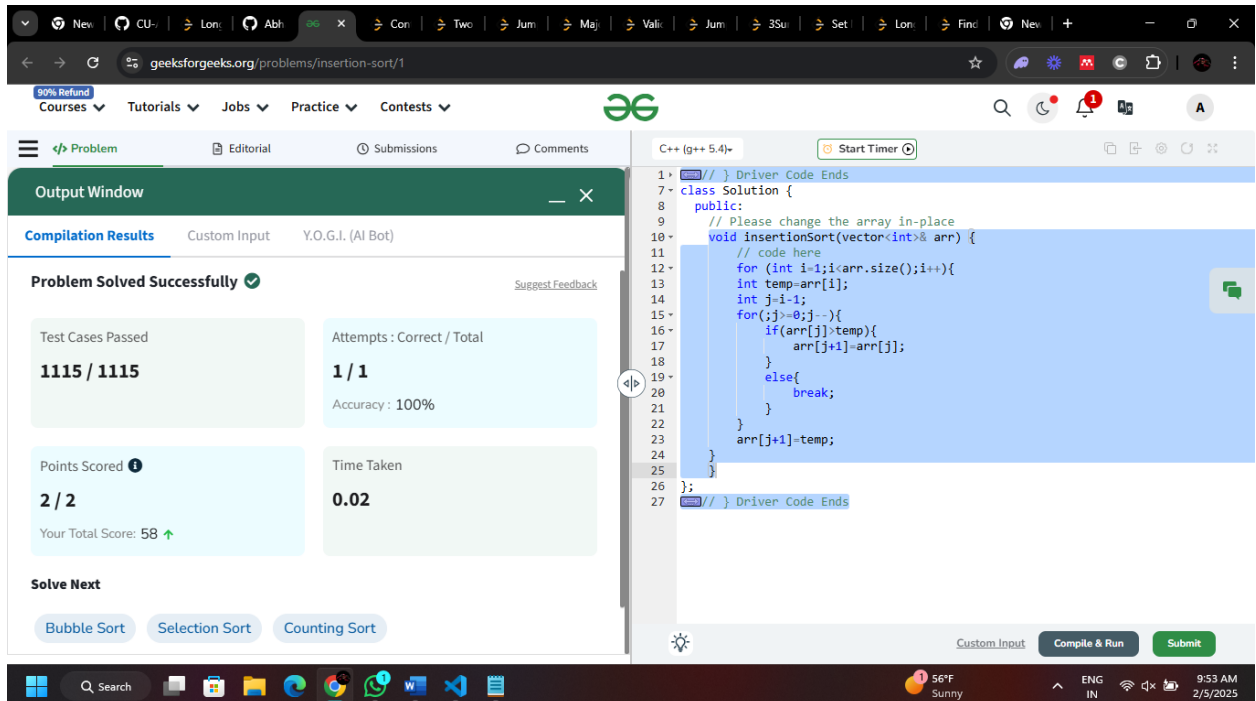
```
int removeDuplicates(vector<int>& nums) {  
    int k=0;int i=0;  
    int size=nums.size();  
    for(int j=1;j<size;j++){  
        if(nums[i]!=nums[j]){  
            nums[i+1]=nums[j];  
            i++;  
        }  
    }  
    return i+1;  
}
```

Testcase 1: Case 1

nums = [1, 1, 2]

2. Implementing Insertion Sort-

```
void insertionSort(vector<int>& arr) {
    // code here
    for (int i=1;i<arr.size();i++){
        int temp=arr[i];
        int j=i-1;
        for(;j>=0;j--){
            if(arr[j]>temp){
                arr[j+1]=arr[j];
            }
            else{
                break;
            }
        }
        arr[j+1]=temp;
    }
}
```



The screenshot shows a web browser window with the URL <https://www.geeksforgeeks.org/problems/insertion-sort/1>. The page is titled "Insertion Sort" and includes a "90% Refund" badge. The "Output Window" on the left shows the following results:

- Problem Solved Successfully ✓
- Test Cases Passed: 1115 / 1115
- Attempts: Correct / Total: 1 / 1
- Accuracy: 100%
- Points Scored: 2 / 2
- Time Taken: 0.02
- Your Total Score: 58 ↑

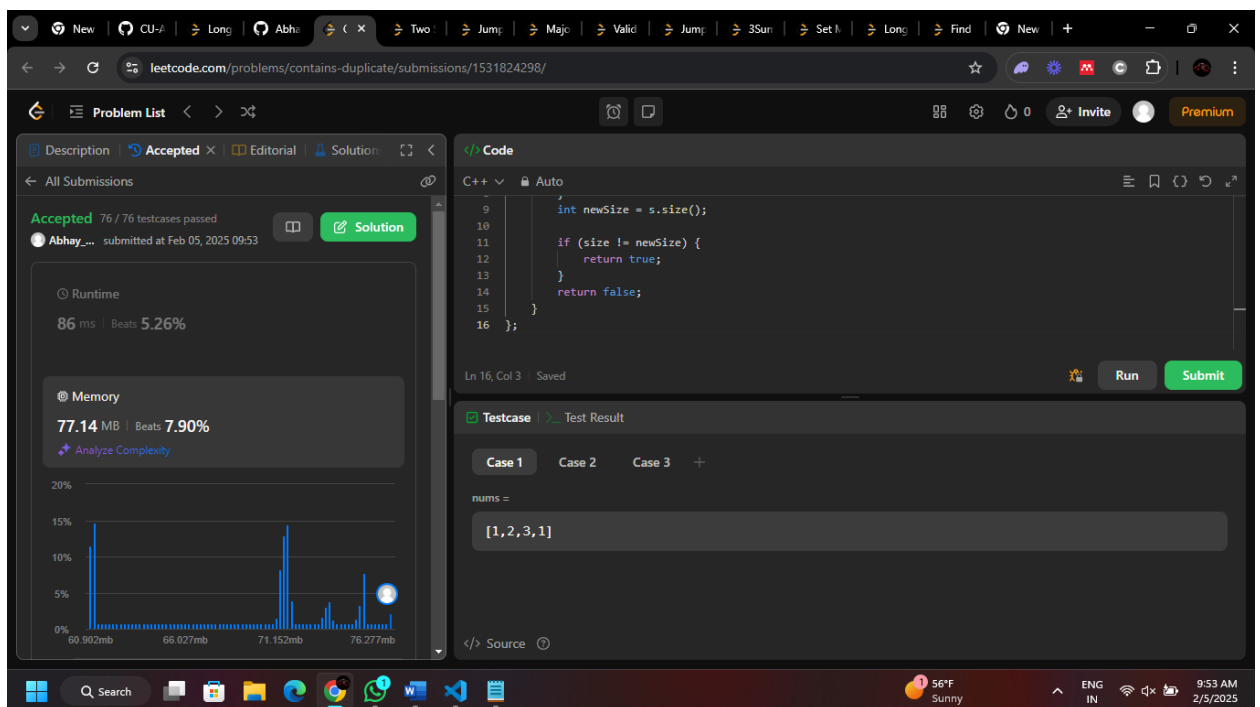
The "Solve Next" section lists "Bubble Sort", "Selection Sort", and "Counting Sort". The main code editor on the right shows the following C++ code:

```
1 // Driver Code Ends
2 class Solution {
3 public:
4     // Please change the array in-place
5     // code here
6     void insertionSort(vector<int>& arr) {
7         for (int i=1;i<arr.size();i++){
8             int temp=arr[i];
9             int j=i-1;
10            for(;j>=0;j--){
11                if(arr[j]>temp){
12                    arr[j+1]=arr[j];
13                }
14                else{
15                    break;
16                }
17            }
18            arr[j+1]=temp;
19        }
20    };
21 // Driver Code Ends
```

The bottom of the browser window shows the Windows taskbar with the date and time: 9:53 AM, 2/5/2025.

3. Contains Duplicate-

```
bool containsDuplicate(vector<int>& nums) {  
    int size = nums.size();  
    set<int> s;  
    for (auto it : nums) {  
        s.insert(it);  
    }  
    int newSize = s.size();  
  
    if (size != newSize) {  
        return true;  
    }  
    return false;  
}
```



The screenshot shows a web browser displaying a LeetCode submission for the problem "Contains Duplicate". The submission is accepted, with a runtime of 86 ms and memory usage of 77.14 MB. The code is written in C++ and uses a set to check for duplicates. The test case input is [1, 2, 3, 1].

Browser tabs: New, CU-1, Long, Abhi, Two, Jump, Majo, Valid, Jump, 3Sun, Set 1, Long, Find, New, +

URL: leetcode.com/problems/contains-duplicate/submissions/1531824298/

Problem List < > >>

Description Accepted x Editorial Solution <

All Submissions

Accepted 76 / 76 testcases passed

Abhay... submitted at Feb 05, 2025 09:53

Runtime: 86 ms Beats 5.26%

Memory: 77.14 MB Beats 7.90%

Analyze Complexity

Testcase Test Result

Case 1 Case 2 Case 3 +

nums =

[1, 2, 3, 1]

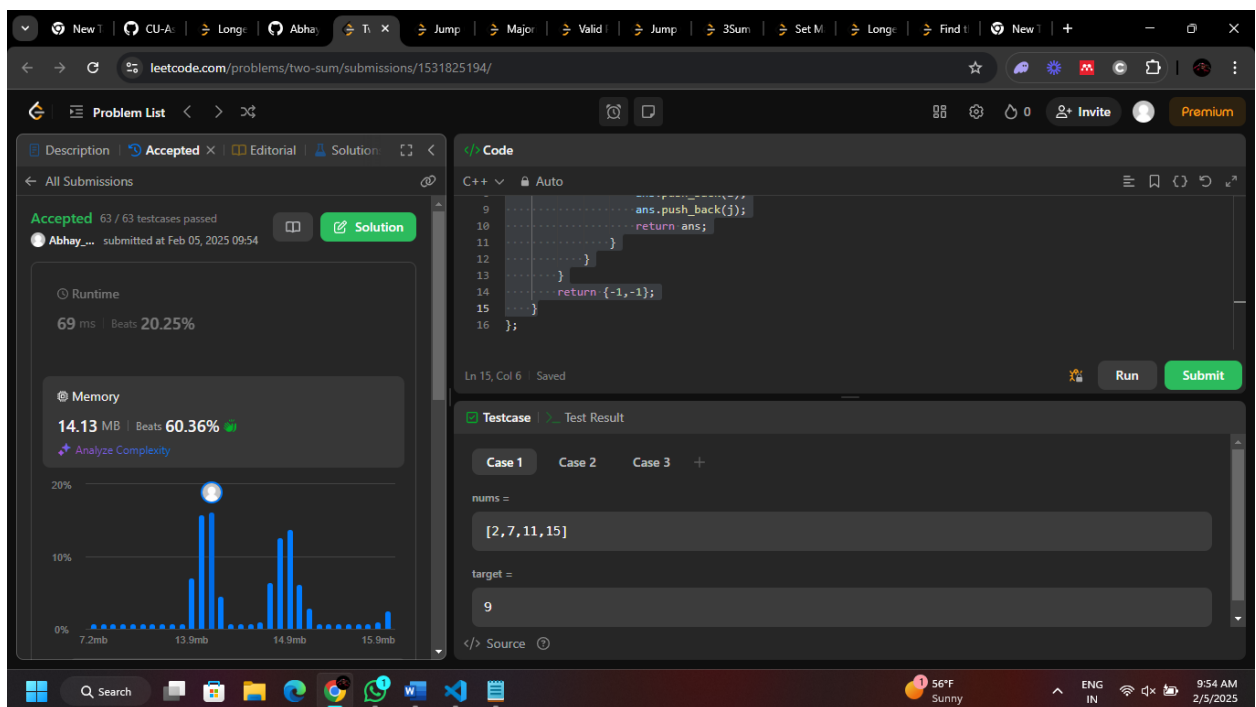
Ln 16, Col 3 Saved Run Submit

Source

Windows taskbar: Search, 56°F Sunny, 9:53 AM 2/5/2025

4. Two Sum –

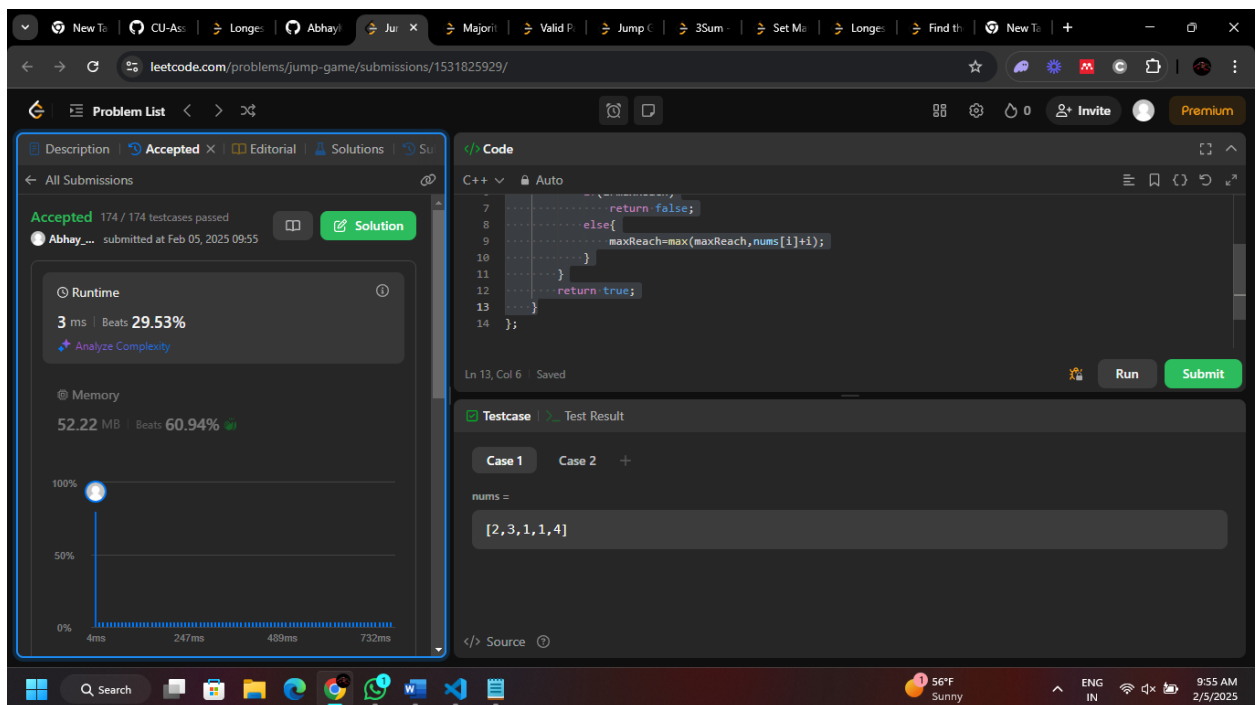
```
vector<int> twoSum(vector<int>& nums, int target) {
    vector<int> ans;
    for(int i=0;i<nums.size();i++){
        for(int j=i+1;j<nums.size();j++){
            if(nums[i]+nums[j]==target){
                ans.push_back(i);
                ans.push_back(j);
                return ans;
            }
        }
    }
    return {-1,-1};
}
```



The screenshot displays a LeetCode submission for the 'Two Sum' problem. The interface includes a sidebar with the problem description, a main code editor, and a test results section. The code is written in C++ and is marked as 'Accepted'. The runtime is 69 ms, and the memory usage is 14.13 MB. The test case shows the input array [2, 7, 11, 15] and the target value 9, with the expected output being the indices [0, 1].

5. Jump Game-

```
bool canJump(vector<int>& nums) {  
    int maxReach=0;  
    for(int i=0;i<nums.size();i++){  
        if(i>maxReach)  
            return false;  
        else{  
            maxReach=max(maxReach,nums[i]+i);  
        }  
    }  
    return true;  
}
```



The screenshot displays a web browser window with the URL <https://leetcode.com/problems/jump-game/submissions/1531825929/>. The page shows the submission details for the 'Jump Game' problem. The submission is marked as 'Accepted' with 174 / 174 testcases passed. The user 'Abhay...' submitted it on Feb 05, 2025 at 09:55. The runtime is 3 ms, which beats 29.53% of other submissions. The memory usage is 52.22 MB, which beats 60.94% of other submissions. A graph shows the performance comparison. The code is written in C++ and uses a greedy algorithm to determine if it's possible to jump from the first index to the last index of the array. The test case shows the input array `nums = [2, 3, 1, 1, 4]`.

```
7 ..... return false;  
8 ..... else{  
9 .....     maxReach=max(maxReach,nums[i]+i);  
10 ..... }  
11 ..... }  
12 ..... return true;  
13 ..... }  
14 .....
```

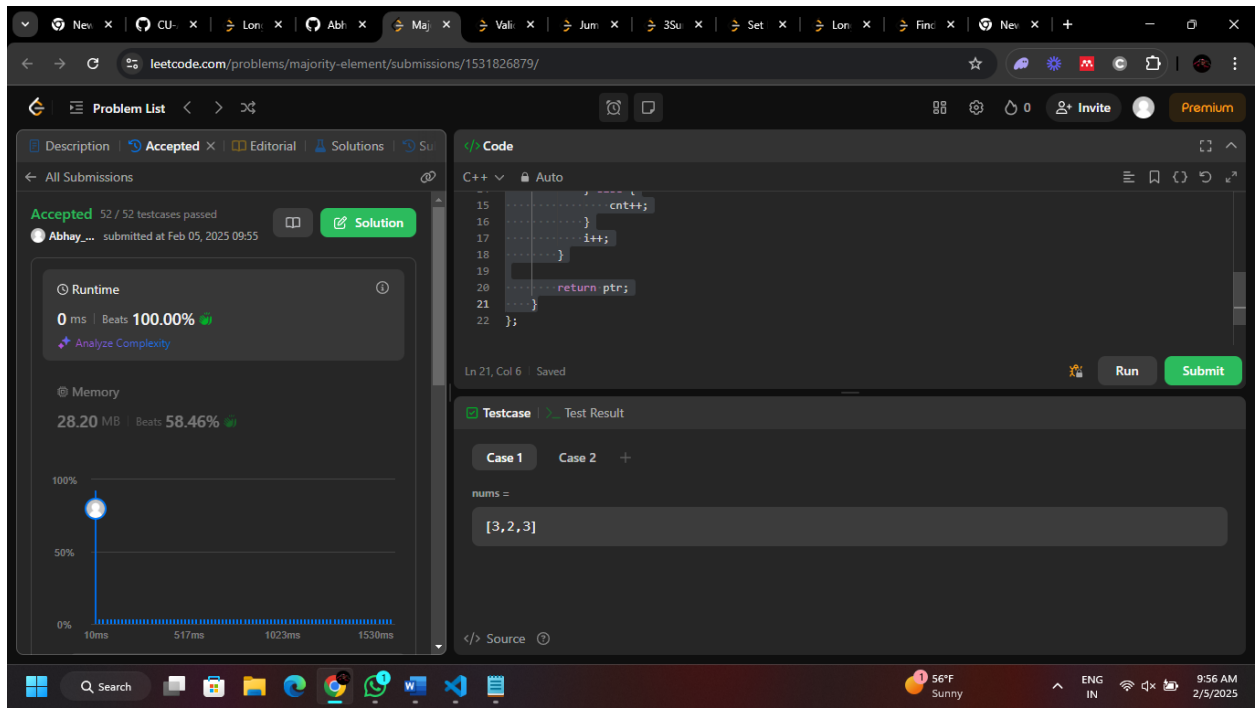
Case 1 Case 2 +

nums =

[2, 3, 1, 1, 4]

6. Majority Element-

```
int majorityElement(vector<int>& nums) {  
    int cnt = 0;  
    int i = 0;  
    int ptr = 0;  
  
    while (i < nums.size()) {  
        if (cnt == 0) {  
            ptr = nums[i];  
        }  
        if (ptr != nums[i]) {  
            cnt--;  
        } else {  
            cnt++;  
        }  
        i++;  
    }  
    return ptr;  
}
```



The screenshot displays a web browser window with multiple tabs. The active tab is 'Maj', showing the LeetCode problem 'Majority Element' (1531826879). The solution is marked as 'Accepted' with 52/52 testcases passed. The runtime is 0 ms, beating 100.00% of solutions. The memory usage is 28.20 MB, beating 58.46% of solutions. The code is written in C++ and implements a majority element algorithm using a counter and a pointer. The test case shown is [3, 2, 3], and the result is 3.

```
C++  
15 ... cnt++;  
16 ...  
17 ... i++;  
18 ...  
19 ...  
20 ... return ptr;  
21 ...  
22 ...;
```

Testcase 1: nums = [3, 2, 3]

7. Valid Palindrome –

```
bool valid(char ch){  
    if((ch>='a' && ch<='z') || (ch>='A' && ch<='Z') || (ch>='0' && ch<='9')){  
        return 1;  
    }  
    return 0;  
}
```

```
char toLower(char ch){  
    if((ch>='a' && ch<='z') || (ch>='0' && ch<='9')){  
        return ch;  
    }  
    else{  
        char temp= ch-'A'+'a';  
        return temp;  
    }  
}
```

```
bool checkPalindrome(string a){  
    int s=0;  
    int e=a.length()-1;  
  
    while(s<=e){  
        if(a[s] != a[e]){  
            return 0;  
        }  
        else{  
            s++;  
            e--;  
        }  
    }  
    return 1;  
}
```

public:

```
bool isPalindrome(string s) {  
    string temp="";  
    for(int j=0;j<s.length();j++){
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        if(valid(s[j])){
            temp.push_back(s[j]);
        }
    }

    for(int j=0;j<temp.length();j++){
        temp[j]=toLower(temp[j]);
    }

    return checkPalindrome(temp);
}
}
```

The screenshot displays a LeetCode submission for the 'Valid Palindrome' problem. The code is written in C++ and is shown in the 'Code' tab. The submission is 'Accepted' and has passed 486 out of 486 testcases. The runtime is 2 ms, which beats 39.72% of other submissions. The memory usage is 10.69 MB, which beats 9.85% of other submissions. A bar chart shows the distribution of runtime results. The test case shown is 'A man, a plan, a canal: Panama'.

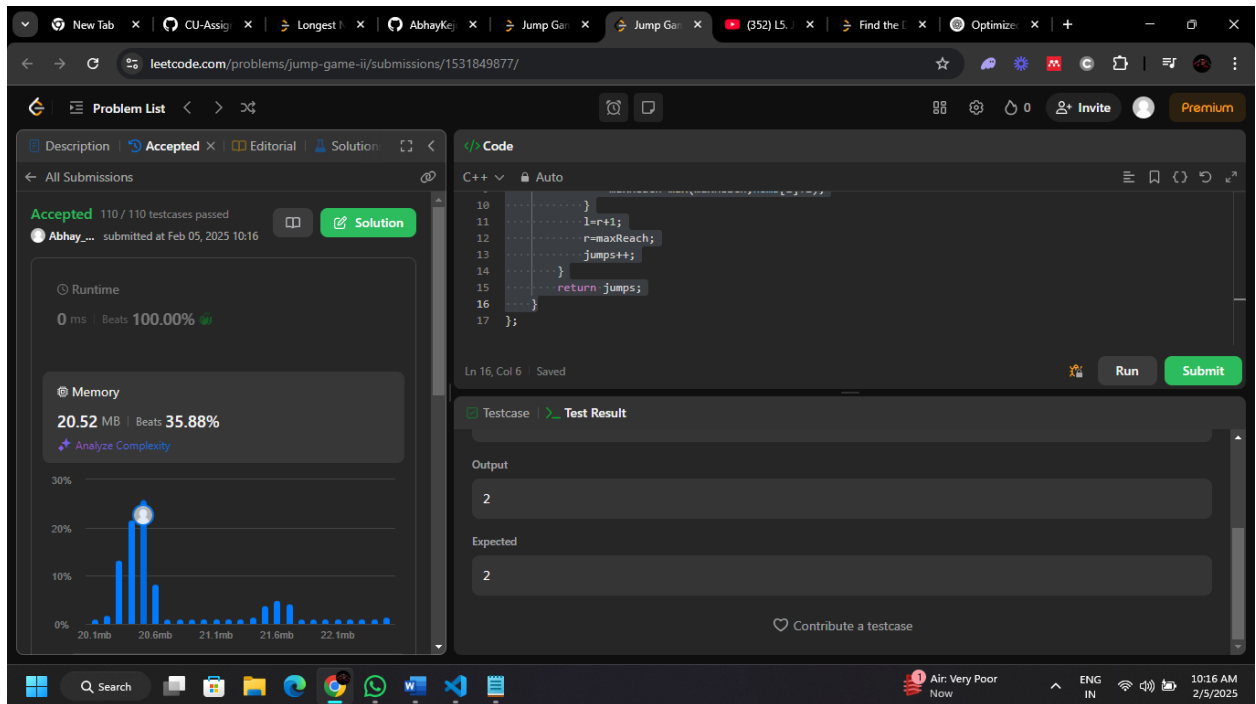
Accepted 486 / 486 testcases passed
Abhay... submitted at Feb 05, 2025 09:56

Runtime: 2 ms | Beats 39.72%
Memory: 10.69 MB | Beats 9.85%

Testcase: Case 1 Case 2 Case 3 +
s =
"A man, a plan, a canal: Panama"

8. Jump Game 2-

```
int jump(vector<int>& nums) {  
    int jumps=0;  
    int l=0,r=0;  
    while(r<nums.size()-1){  
        int maxReach=0;  
        for(int i=l;i<=r;i++){  
            maxReach=max(maxReach,nums[i]+i);  
        }  
        l=r+1;  
        r=maxReach;  
        jumps++;  
    }  
    return jumps;  
}
```



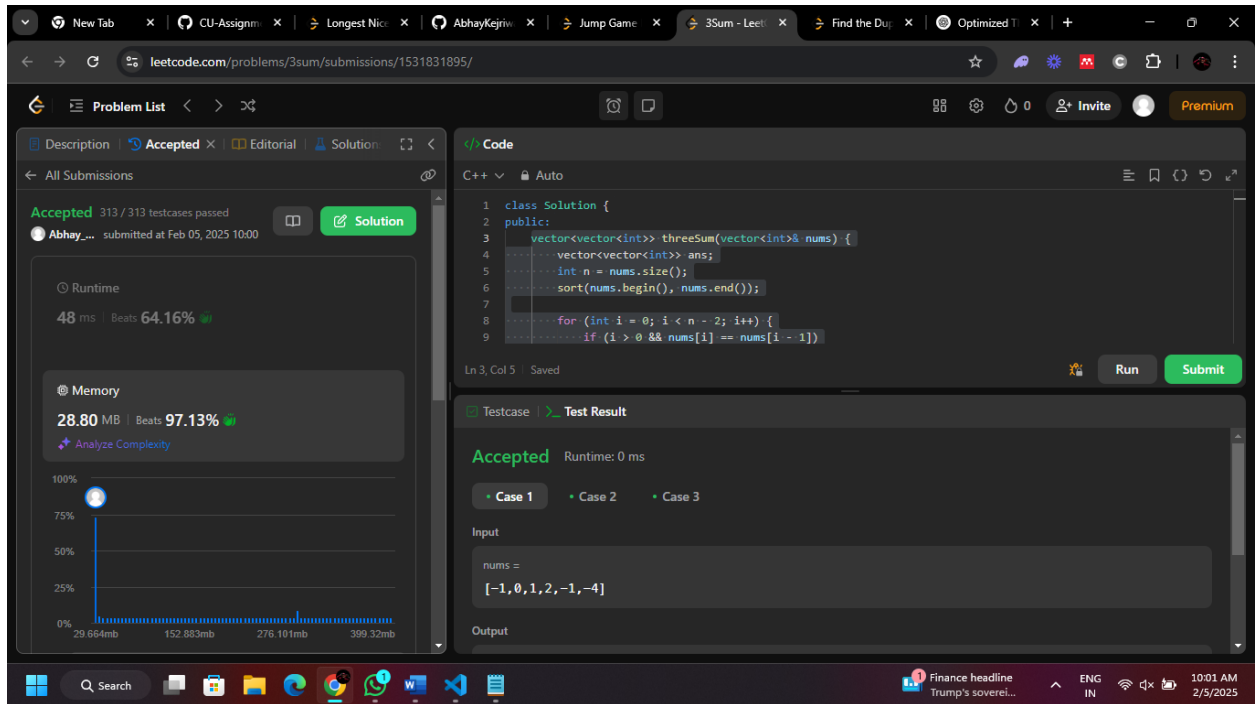
The screenshot shows a web browser displaying a LeetCode submission for the problem "Jump Game II". The submission is for user "Abhay_..." and is marked as "Accepted". The code is written in C++ and implements a greedy algorithm to find the minimum number of jumps to reach the end of the array. The submission details show a runtime of 0 ms, beating 100.00% of submissions, and a memory usage of 20.52 MB, beating 35.88% of submissions. A bar chart shows the memory usage distribution. The code is as follows:

```
10 .....  
11 ..... l=r+1;  
12 ..... r=maxReach;  
13 ..... jumps++;  
14 .....  
15 ..... return jumps;  
16 .....  
17 .....
```

The submission is successful, with the output "2" matching the expected output "2".

9. 3 Sum-

```
vector<vector<int>> threeSum(vector<int>& nums) {  
    vector<vector<int>> ans;  
    int n = nums.size();  
    sort(nums.begin(), nums.end());  
  
    for (int i = 0; i < n - 2; i++) {  
        if (i > 0 && nums[i] == nums[i - 1])  
            continue;  
        int left = i + 1, right = n - 1;  
        while (left < right) {  
            int sum = nums[i] + nums[left] + nums[right];  
  
            if (sum == 0) {  
                ans.push_back({nums[i], nums[left], nums[right]});  
  
                while (left < right && nums[left] == nums[left + 1])  
                    left++;  
                while (left < right && nums[right] == nums[right - 1])  
                    right--;  
  
                left++, right--;  
            } else if (sum < 0) {  
                left++; // Increase the sum by moving left pointer  
            } else {  
                right--; // Decrease the sum by moving right pointer  
            }  
        }  
    }  
    return ans;  
}
```



10. Set Matrix Zeros-

```
void setZeroes(vector<vector<int>>& matrix) {
    int row=matrix.size();
    int col=matrix[0].size();
```

```
    vector<int> indexRow(row,0);
    vector<int> indexCol(col,0);
```

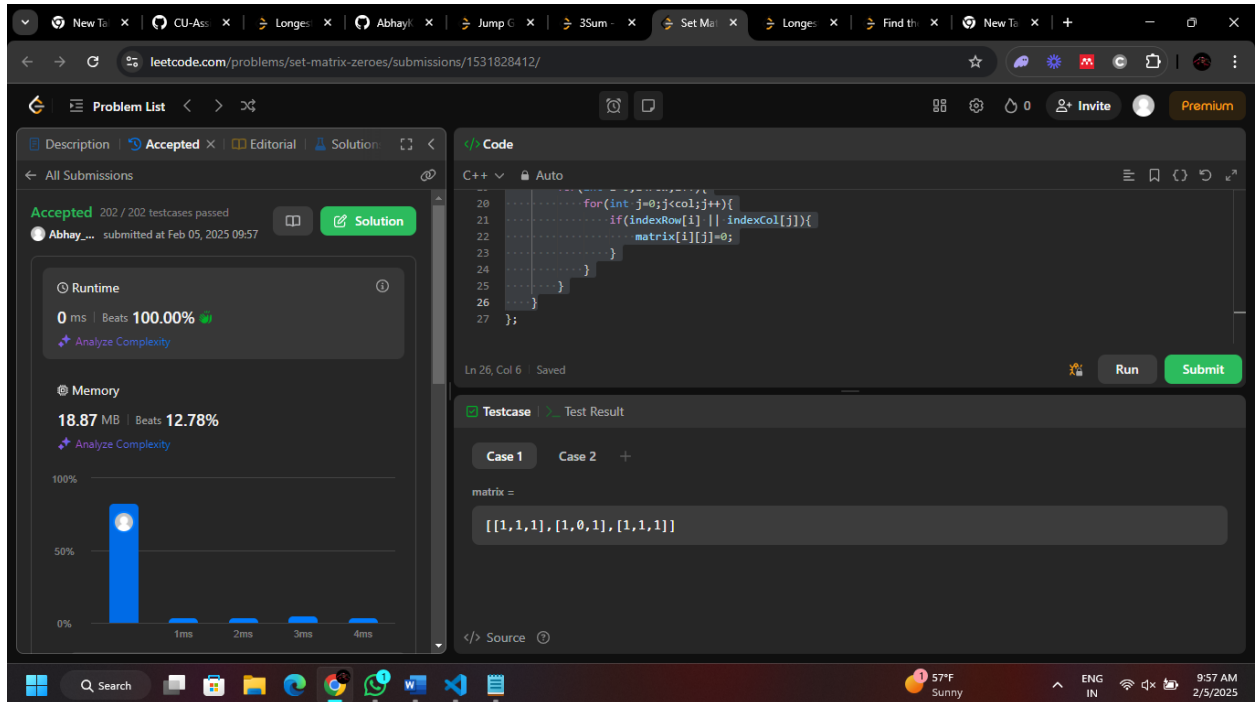
```
    for(int i=0;i<row;i++){
        for(int j=0;j<col;j++){
            if(matrix[i][j]==0){
                indexRow[i] =1;
                indexCol[j]=1;
            }
        }
    }
}
```

```
for(int i=0;i<row;i++){
    for(int j=0;j<col;j++){
        if(indexRow[i] || indexCol[j]){
```

```

        matrix[i][j]=0;
    }
}
}
}

```



The screenshot shows a LeetCode submission for the problem 'Set Matrix Zeroes'. The submission is accepted, with a runtime of 0 ms and memory usage of 18.87 MB. The code is in C++ and uses a row and column index tracking method. The test case shows a 3x3 matrix with the result [[1,1,1],[1,0,1],[1,1,1]].

11. Longest Substring Without Repeating Characters-

```

int lengthOfLongestSubstring(string s) {
    vector<int> mpp(256, -1);

```

```

    int left = 0, right = 0;
    int n = s.size();
    int len = 0;
    while (right < n) {
        if (mpp[s[right]] != -1)
            left = max(mpp[s[right]] + 1, left);

```

```

        mpp[s[right]] = right;

```

```

        len = max(len, right - left + 1);

```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        right++;  
    }  
    return len;  
}
```

The screenshot shows a web browser window displaying a LeetCode submission for the problem "Longest Substring Without Repeating Characters". The submission is in C++ and has been accepted. The code uses a sliding window approach to find the longest substring without repeating characters. The submission details show a runtime of 0 ms and a memory usage of 11.56 MB, both of which are optimal. A test case is provided with the input string "abcabcbb", and the output is 3.

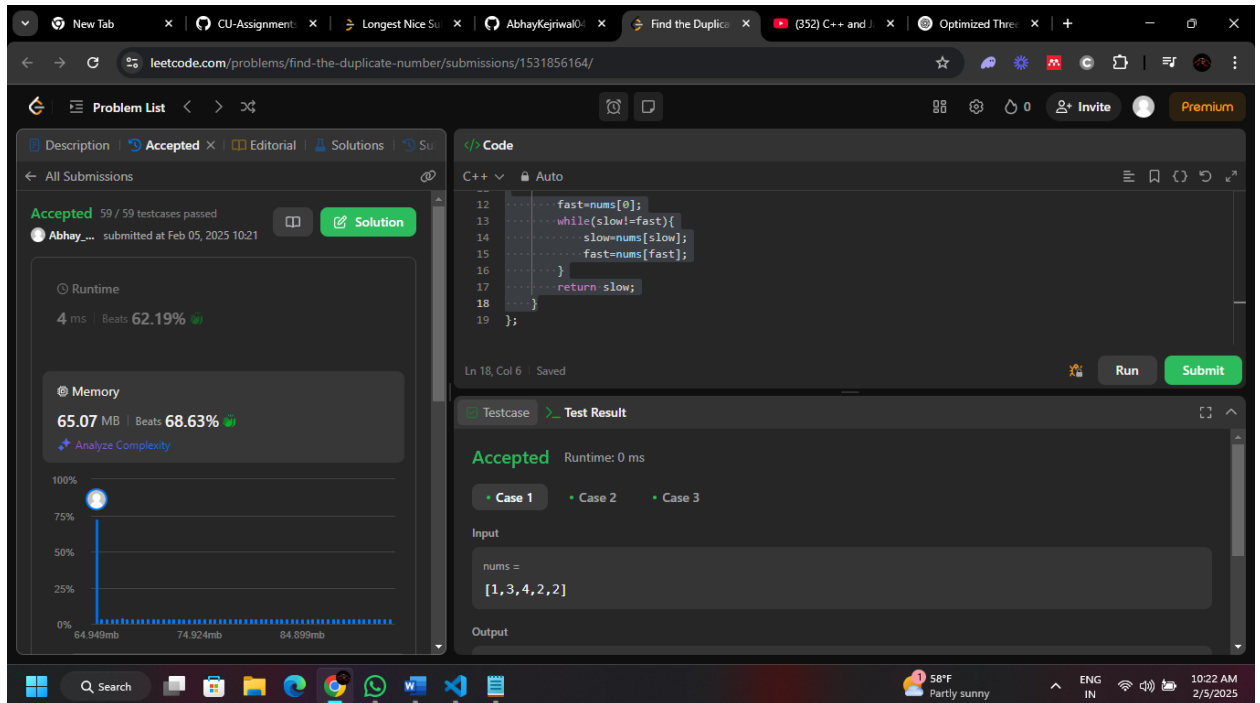
```
13     mpp[s[right]] = right;  
14  
15     len = max(len, right - left + 1);  
16     right++;  
17 }  
18 return len;  
19 }  
20 }
```

Runtime: 0 ms | Beats 100.00%
Memory: 11.56 MB | Beats 81.40%

Testcase: Case 1 Case 2 Case 3 +
s =
"abcabcbb"

12. Finding Duplicate Number-

```
int findDuplicate(vector<int>& nums) {  
    int slow=nums[0];  
    int fast=nums[0];  
  
    do{  
        slow=nums[slow];  
        fast=nums[nums[fast]];  
    }while(slow!=fast);  
  
    fast=nums[0];  
    while(slow!=fast){  
        slow=nums[slow];  
        fast=nums[fast];  
    }  
    return slow;  
}
```



The screenshot shows a LeetCode submission for the problem "Find the Duplicate Number". The submission is accepted, with a runtime of 4 ms and memory usage of 65.07 MB. The code is in C++ and implements the Floyd's Cycle-Finding algorithm. The test case input is [1, 3, 4, 2, 2] and the output is 2.

```
12     fast=nums[0];  
13     while(slow!=fast){  
14         slow=nums[slow];  
15         fast=nums[fast];  
16     }  
17     return slow;  
18 }  
19 ;
```

Testcase: Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

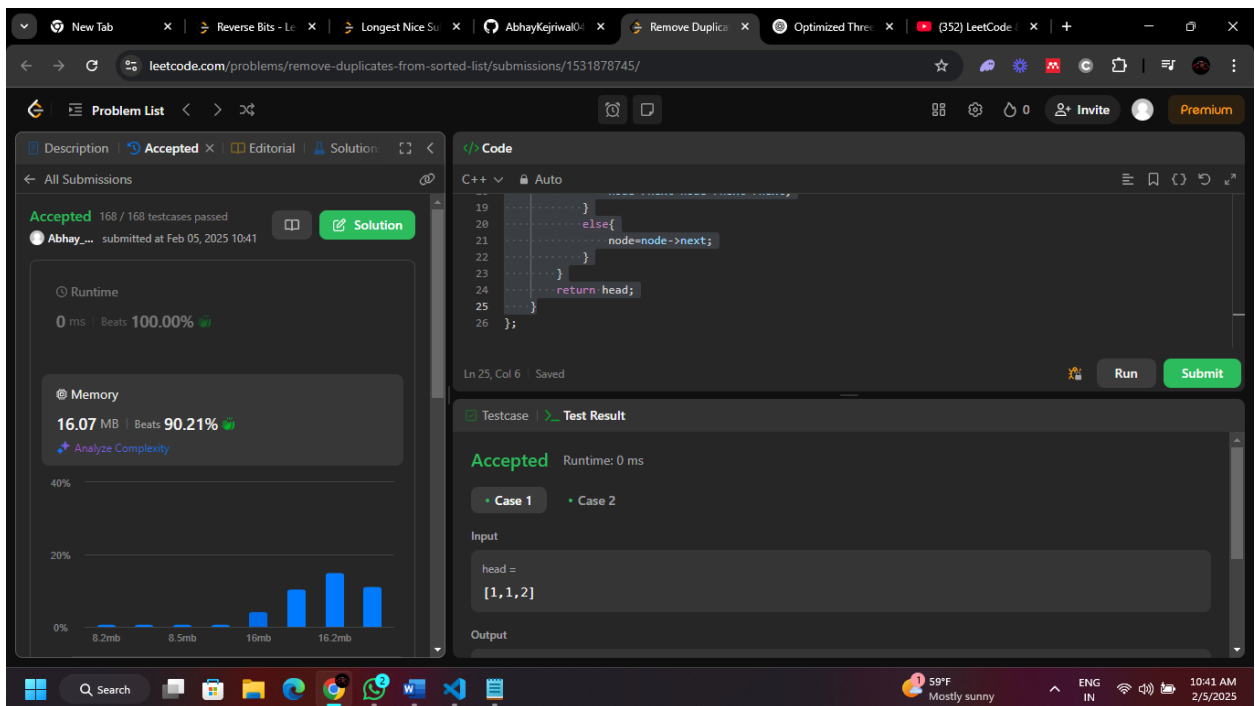
Input: nums = [1, 3, 4, 2, 2]

Output: 2

DAY-2:

1. Remove Duplicates From A Sorted List-

```
ListNode* deleteDuplicates(ListNode* head) {  
    ListNode* node=head;  
    while(node!=NULL && node->next!=NULL){  
        if(node->val==node->next->val){  
            ListNode* temp=node->next;  
            node->next=node->next->next;  
        }  
        else{  
            node=node->next;  
        }  
    }  
    return head;  
}
```



The screenshot displays a web browser window with multiple tabs open, including 'Reverse Bits - Le', 'Longest Nice Su', 'AbhayKejriwal01', 'Remove Duplica', 'Optimized Thre', and '(352) LeetCode'. The active tab shows the LeetCode problem 'Remove Duplicates from Sorted List' with a submission ID of 1531878745. The submission is marked as 'Accepted' with 168 / 168 testcases passed. The user 'Abhey...' submitted it on Feb 05, 2025 at 10:41. The runtime is 0 ms, beating 100.00% of other submissions. The memory usage is 16.07 MB, beating 90.21% of other submissions. A bar chart shows the memory usage distribution. The code is written in C++ and uses a linked list approach to remove duplicates. The input is 'head = [1,1,2]' and the output is '[1,1,2]'. The code is as follows:

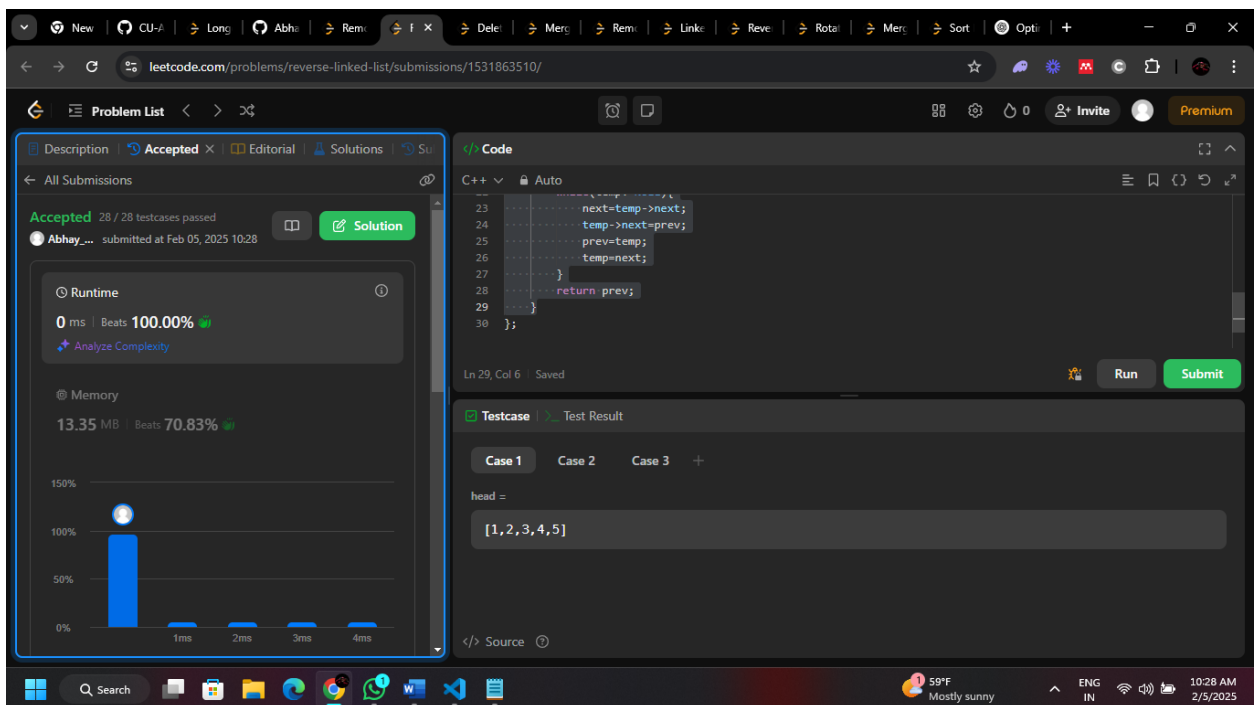
```
19 .....  
20 .....else{  
21 .....    node=node->next;  
22 .....}  
23 .....  
24 .....    return head;  
25 .....  
26 .....
```

2. Reverse A Linked List –

```
ListNode* reverseList(ListNode* head) {  
    if(head==NULL || head->next==NULL){  
        return head;  
    }  
}
```

```
ListNode* prev=NULL;  
ListNode* temp=head;  
ListNode* next=NULL;
```

```
while(temp!=NULL){  
    next=temp->next;  
    temp->next=prev;  
    prev=temp;  
    temp=next;  
}  
return prev;  
}
```



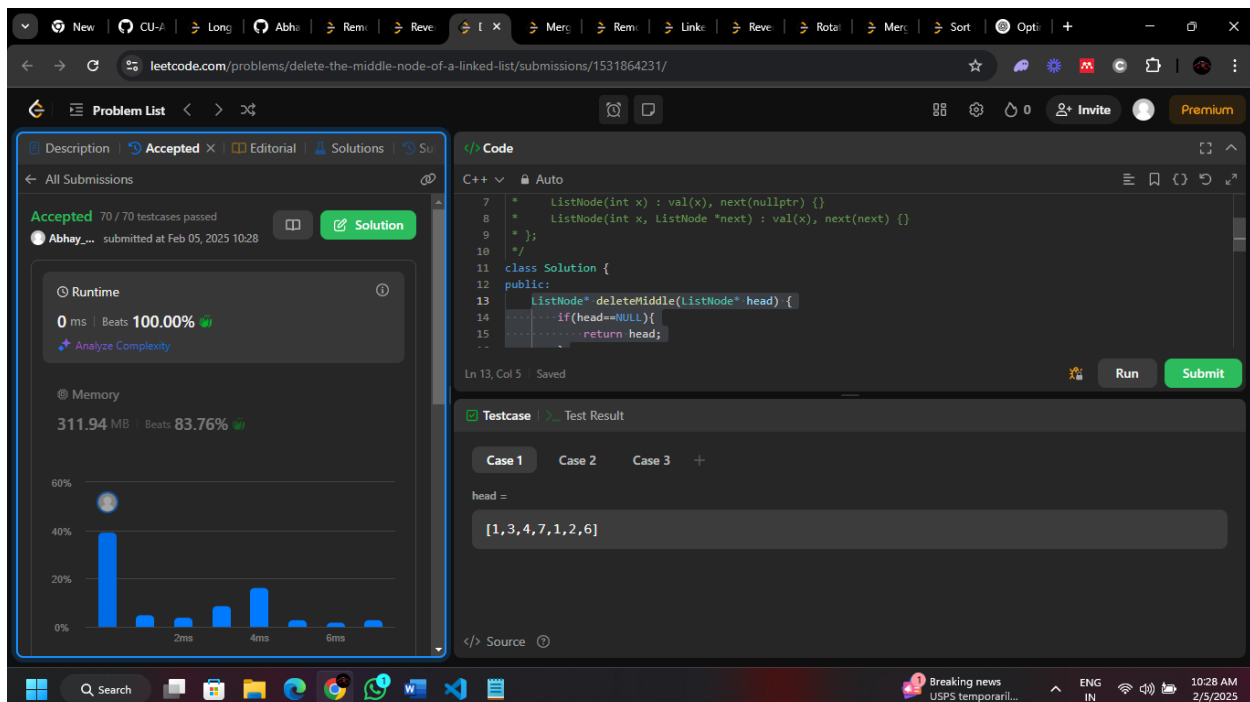
The screenshot shows a web browser window displaying a LeetCode submission for the problem "Reverse a Linked List" (problem ID 1531863510). The submission is in C++ and has been accepted, passing all 28 test cases. The left sidebar shows the submission details, including the runtime (0 ms) and memory usage (13.35 MB). The main area displays the C++ code for reversing the linked list, which uses a three-pointer approach (prev, temp, next). The code is as follows:

```
23 ...next=temp->next;  
24 ...temp->next=prev;  
25 ...prev=temp;  
26 ...temp=next;  
27 ...}  
28 ...return prev;  
29 ...}  
30 ...;
```

Below the code, there is a "Testcase" section showing the input for "Case 1": head = [1,2,3,4,5]. The bottom of the screen shows a Windows taskbar with the date and time as 10:28 AM on 2/5/2025.

3. Delete Middle Node Of A List-

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



The screenshot shows a LeetCode submission for the problem "Delete the Middle Node of a Linked List". The submission is accepted, with a runtime of 0 ms and memory usage of 311.94 MB. The code is in C++ and implements the deleteMiddle function. The test case input is [1, 3, 4, 7, 1, 2, 6].

Runtime: 0 ms | Beats 100.00%

Memory: 311.94 MB | Beat: 83.76%

Code:

```
C++  
class Solution {  
public:  
    ListNode* deleteMiddle(ListNode* head) {  
        if(head==NULL){  
            return head;  
        }  
        if(head->next==NULL){  
            head=head->next;  
            return head;  
        }  
        ListNode* fast=head;  
        ListNode* slow=head;  
        ListNode* prev=NULL;  
        while(fast!=NULL && fast->next!=NULL){  
            fast=fast->next->next;  
            prev=slow;  
            slow=slow->next;  
        }  
        prev->next=slow->next;  
        slow=slow->next;  
        return head;  
    }  
};
```

Testcase: Case 1 Case 2 Case 3 +

head = [1, 3, 4, 7, 1, 2, 6]

4. Merge Two Sorted Linked List-

```
ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {  
    if (list1 == NULL && list2 == NULL) {  
        return NULL;  
    }  
  
    if (list1 == NULL) {  
        return list2;  
    }  
    if (list2 == NULL) {  
        return list1;  
    }  
  
    ListNode* dummy = new ListNode(-1);  
    ListNode* head = dummy;  
  
    while (list1 != NULL && list2 != NULL) {  
        if (list1->val <= list2->val) {  
            head->next = list1;  
            list1 = list1->next;  
        } else {  
            head->next = list2;  
            list2 = list2->next;  
        }  
        head = head->next;  
    }  
  
    if (list1 != NULL) {  
        head->next = list1;  
    } else if (list2 != NULL) {  
        head->next = list2;  
    }  
  
    return dummy->next;  
}
```



Discover. Learn. Empower.

[illegible]

5. Detect A Cycle In A Linked List-

```
bool hasCycle(ListNode *head) {
```

```
    ListNode* fast=head;
```

```
    ListNode* slow=head;
```

```
    while(fast!=NULL && fast->next!=NULL){
```

```
        fast=fast->next->next;
```

```
        slow=slow->next;
```

```
        if(slow==fast){
```

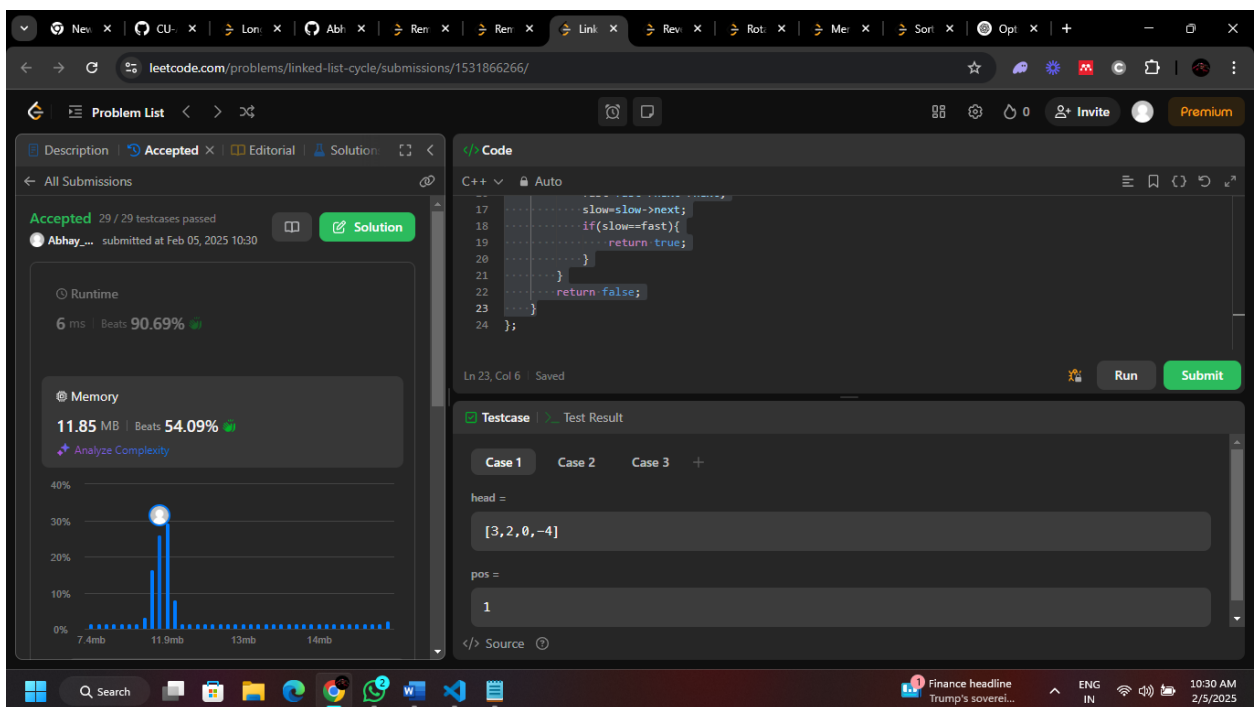
```
            return true;
```

```
        }
```

```
    }
```

```
    return false;
```

```
}
```



The screenshot displays a web browser window with the URL leetcode.com/problems/linked-list-cycle/submissions/1531866265/. The page shows the submission details for the 'Linked List Cycle' problem. The submission is marked as 'Accepted' with 29/29 testcases passed. The runtime is 6 ms, and the memory usage is 11.85 MB, both of which are within the constraints. The code is written in C++ and uses the Floyd's Cycle-Finding algorithm. The test case shows a linked list with values [3, 2, 0, -4] and a position of 1.

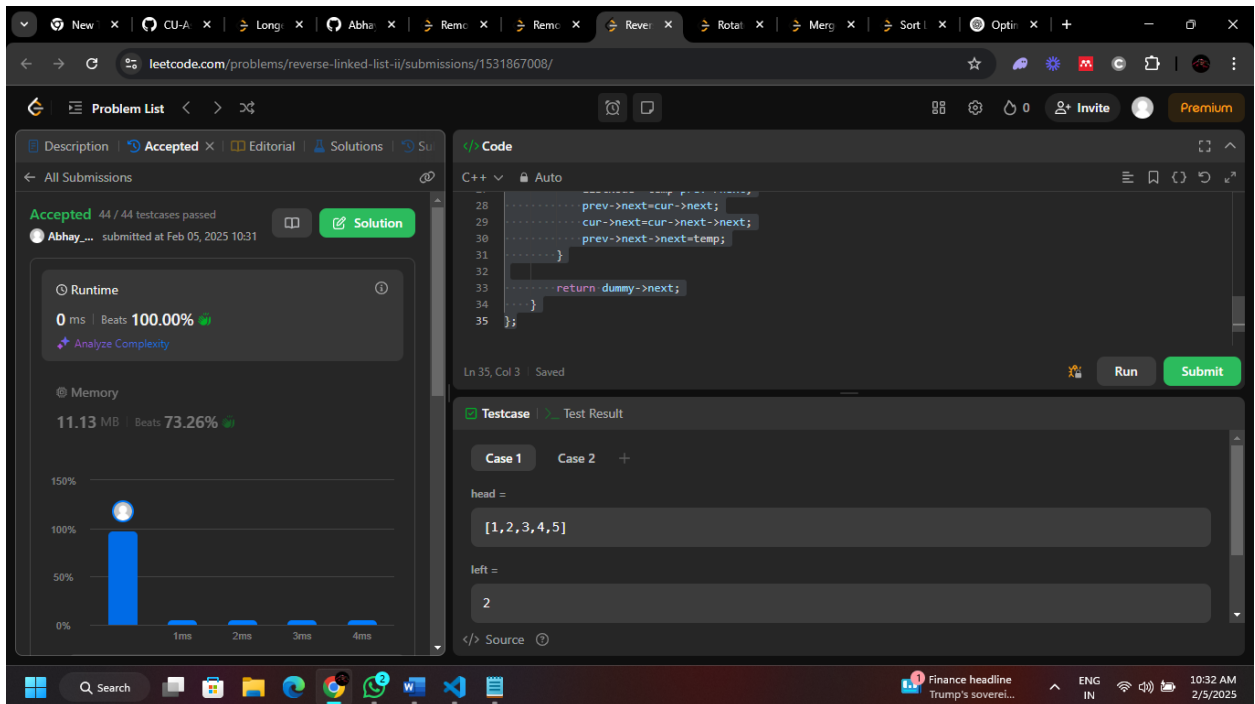
```
bool hasCycle(ListNode *head) {
    ListNode* fast=head;
    ListNode* slow=head;

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

Testcase 1: head = [3, 2, 0, -4], pos = 1

6. Reverse Linked List 2-

```
ListNode* reverseBetween(ListNode* head, int left, int right) {  
    if(head==NULL || head->next==NULL){  
        return head;  
    }  
    ListNode* dummy=new ListNode(-1);  
    dummy->next=head;  
    ListNode* prev=dummy;  
    for(int i=1;i<left;i++){  
        prev=prev->next;  
    }  
    ListNode* cur=prev->next;  
    for(int i=0;i<right-left;i++){  
        ListNode* temp=prev->next;  
        prev->next=cur->next;  
        cur->next=cur->next->next;  
        prev->next->next=temp;  
    }  
    return dummy->next;  
}
```



The screenshot displays a web browser window with the URL leetcode.com/problems/reverse-linked-list-ii/submissions/1531867008/. The page shows the submission details for the problem "Reverse Linked List II". The submission is marked as "Accepted" with 44/44 testcases passed. The user "Abhey..." submitted it on Feb 05, 2025 at 10:31. The code is written in C++ and implements the reverseBetween function. The test case shows head = [1,2,3,4,5] and left = 2, resulting in [1,4,3,2,5]. The runtime is 0 ms and memory is 11.13 MB.

Runtime: 0 ms | Beats 100.00%
Memory: 11.13 MB | Beats 73.26%

Testcase: Case 1 Case 2 +

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

left =
2

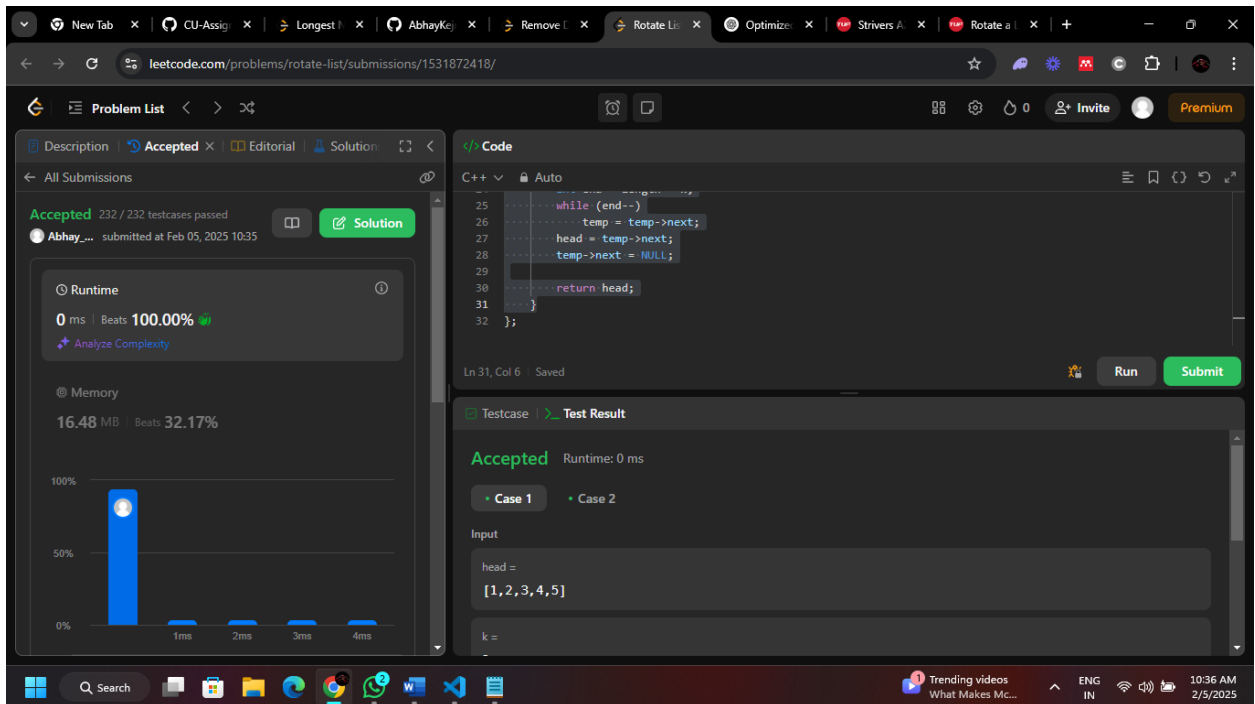
Result: [1,4,3,2,5]

```

ListNode* rotateRight(ListNode* head, int k) {
    if (head == NULL || head->next == NULL || k == 0)
        return head;
    ListNode* temp = head;
    int length = 1;
    while (temp->next != NULL) {
        ++length;
        temp = temp->next;
    }
    temp->next = head;
    k = k % length;
    int end = length - k;
    while (end--)
        temp = temp->next;
    head = temp->next;
    temp->next = NULL;

    return head;
}

```



8. Sort List –

```
ListNode* findMiddle(ListNode* head){
    ListNode* slow=head;
    ListNode* fast=head->next;

    while(fast!=NULL && fast->next!=NULL){
        slow=slow->next;
        fast=fast->next->next;
    }
    return slow;
}

ListNode* mergeTwoList(ListNode* left, ListNode* right){
    ListNode* dummy=new ListNode(-1);
    ListNode* temp=dummy;

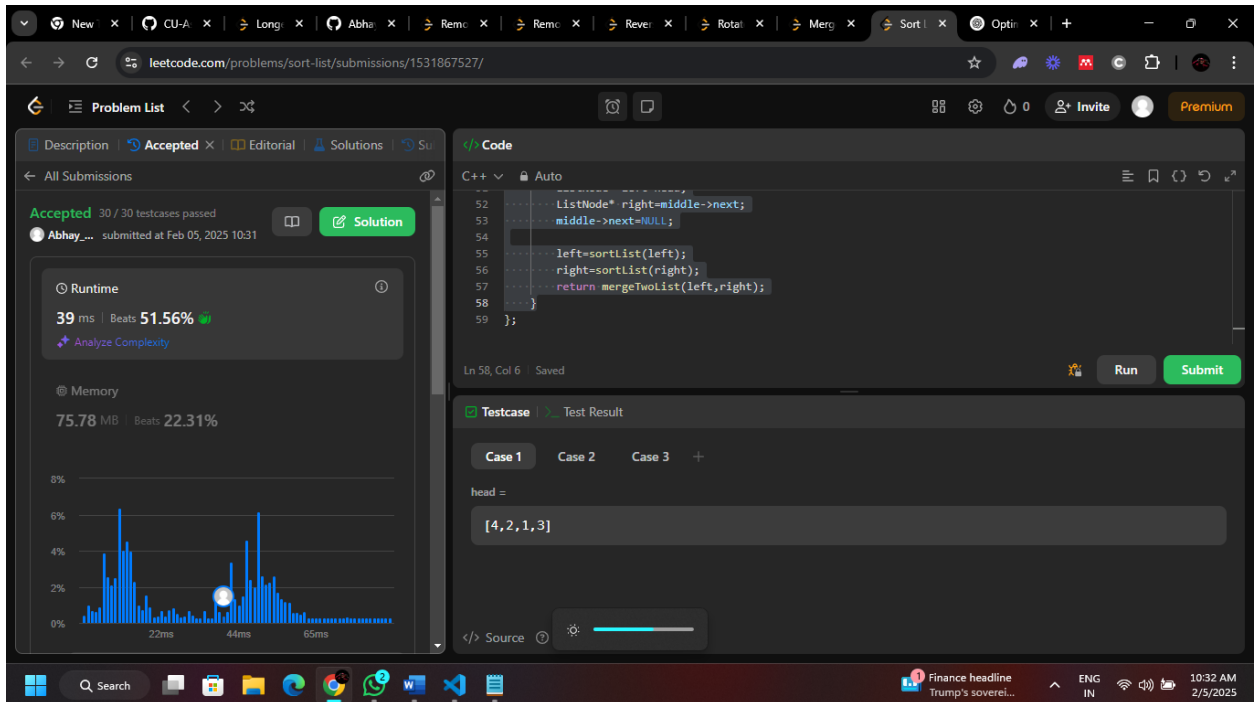
    while(left!=NULL && right!=NULL){
        if(left->val < right->val){
            temp->next=left;
            temp=left;
            left=left->next;
        }
        else{
            temp->next=right;
            temp=right;
            right=right->next;
        }
    }

    if(left)temp->next=left;
    else temp->next=right;

    return dummy->next;
}

ListNode* sortList(ListNode* head) {
    if (head==NULL || head->next==NULL)return head;
```

```
ListNode* middle=findMiddle(head);  
ListNode* left=head;  
ListNode* right=middle->next;  
middle->next=NULL;  
  
left=sortList(left);  
right=sortList(right);  
return mergeTwoList(left,right);  
}
```



The screenshot displays a web browser window with the URL leetcode.com/problems/sort-list/submissions/1531867527/. The page shows the 'Sort List' problem submission details. The submission is marked as 'Accepted' with 30/30 testcases passed. The user 'Abhey...' submitted it on Feb 05, 2025 at 10:31. The runtime is 39 ms, beating 51.56% of solutions. The memory usage is 75.78 MB, beating 22.31% of solutions. A performance graph is visible. The code editor shows the following C++ code:

```
52 ... ListNode* right=middle->next;  
53 ... middle->next=NULL;  
54 ...  
55 ... left=sortList(left);  
56 ... right=sortList(right);  
57 ... return mergeTwoList(left,right);  
58 ...  
59 ... };
```

The test case section shows 'Case 1' with the input 'head = [4, 2, 1, 3]' and the output '[1, 2, 3, 4]'. The bottom of the screen shows a Windows taskbar with the time 10:32 AM on 2/5/2025.