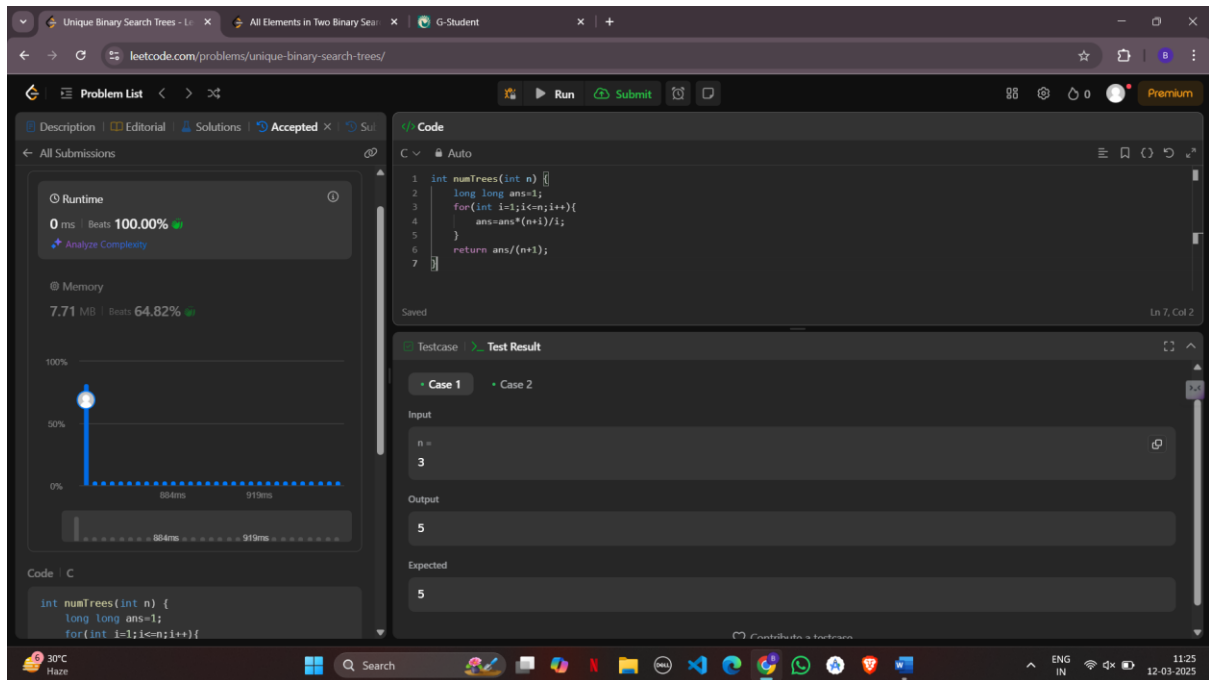


# ASSIGNMENT-4

Bobba Mahidhar  
VU21CSEN0300110  
CSE-AIML

## Unique Binary Search Trees

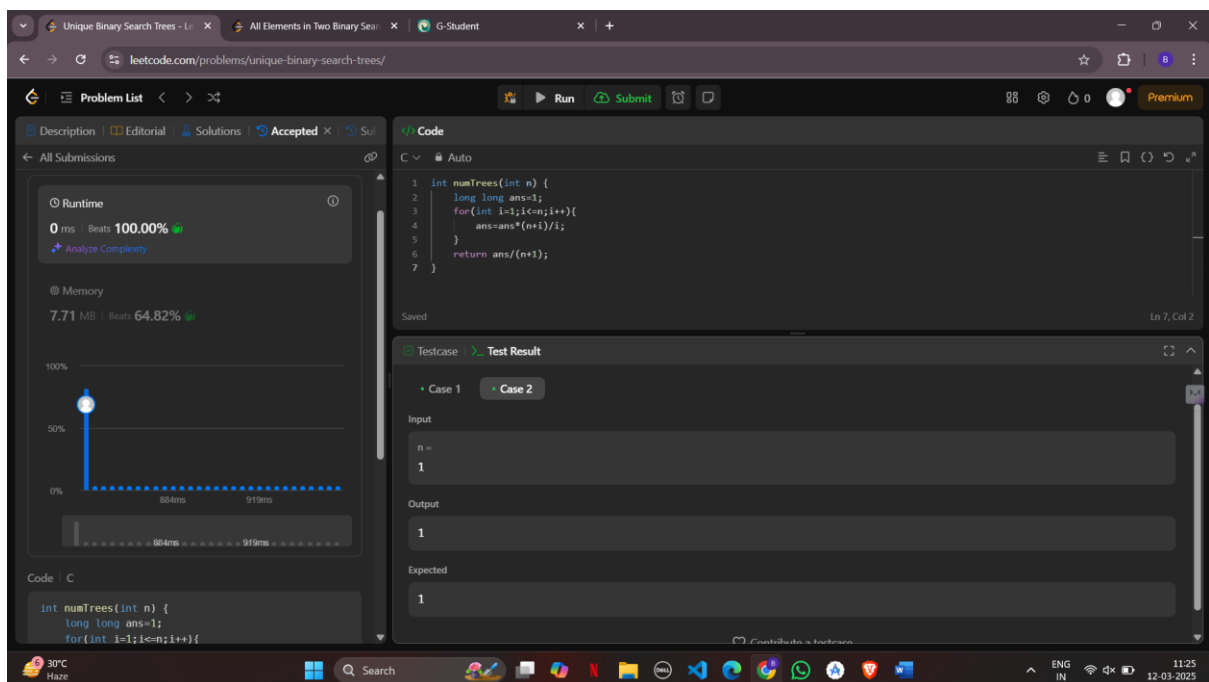


This screenshot shows the LeetCode interface for the problem "Unique Binary Search Trees". The solution is implemented in C++ and has achieved a runtime of 0 ms (beating 100.00%) and a memory usage of 7.71 MB (beating 64.82%). The test case for n=3 shows an output of 5, which matches the expected result.

```
int numTrees(int n) {  
    long long ans=1;  
    for(int i=1;i<=n;i++){  
        ans=ans*(n+1)/i;  
    }  
    return ans/(n+1);  
}
```

Runtime: 0 ms | Beats 100.00%  
Memory: 7.71 MB | Beats 64.82%

Testcase: Case 1  
Input: n = 3  
Output: 5  
Expected: 5



This screenshot shows the same LeetCode interface for the problem "Unique Binary Search Trees", but with the test case for n=1. The solution remains the same, and the output is 1, which matches the expected result.

```
int numTrees(int n) {  
    long long ans=1;  
    for(int i=1;i<=n;i++){  
        ans=ans*(n+1)/i;  
    }  
    return ans/(n+1);  
}
```

Runtime: 0 ms | Beats 100.00%  
Memory: 7.71 MB | Beats 64.82%

Testcase: Case 1  
Input: n = 1  
Output: 1  
Expected: 1

# All Elements in Two Binary Search Trees

This screenshot shows a LeetCode submission for the problem "All Elements in Two Binary Search Trees". The submission is in C and has been accepted. The left sidebar displays the problem description, submission status (Accepted), and performance metrics: Runtime (12 ms, 12.50% faster) and Memory (61.64 MB, 60.00% less). A bar chart shows the runtime distribution across different time intervals. The main code editor displays the C code, which defines a recursive function to traverse both trees and store their elements in two separate arrays, which are then merged into a result array.

```
12 void node_size(struct Treenode *root, int *value){
13     if (!root){
14         return;
15     }
16     (*value)++;
17     node_size(root->left, value);
18     node_size(root->right, value);
19 }
20 void list(struct Treenode *root1, int *index, int *array){
21     if (!root1){
22         return;
23     }
24     list(root1->left, index, array);
25     array[*index++] = root1->val;
26     list(root1->right, index, array);
27 }
28
29 int* getAllElements(struct Treenode *root1, struct Treenode *root2, int* returnSize) {
30     int aa=0, bb=0;
31     node_size(root1, &aa);
32     node_size(root2, &bb);
33     int *first_list=malloc(sizeof(int)*aa);
34     int *second_list=malloc(sizeof(int)*bb);
35     *returnSize=(aa+bb);
36     memset(first_list, -1, sizeof(int)*aa);
37     memset(second_list, -1, sizeof(int)*bb);
38     int index=0;
39     list(root1, &index, first_list);
40     index=0;
41     list(root2, &index, second_list);
42     index=0;
43     int *a=malloc(sizeof(int)*(*returnSize));
44     int a_index=0;
```

This screenshot shows another LeetCode submission for the same problem, also in C and accepted. The performance metrics are identical to the first submission: Runtime (12 ms, 12.50% faster) and Memory (61.64 MB, 60.00% less). The code in this submission uses a different approach, where it first traverses both trees to determine their sizes, then uses two pointers to traverse the trees and merge the elements into a result array. It also includes a cleanup step to free the memory allocated for the intermediate arrays.

```
43 int *a=malloc(sizeof(int)*(*returnSize));
44 int a_index=0;
45 int b_index=0;
46 while(a_index < aa && b_index < bb){
47     if (first_list[a_index] < second_list[b_index]){
48         a[a_index++] = first_list[a_index];
49     }
50     else{
51         a[a_index++] = second_list[b_index];
52     }
53 }
54
55 if (a_index < aa){
56     for (int i=b_index; i<bb; i++){
57         a[a_index++] = second_list[i];
58     }
59 }
60 if (b_index < bb){
61     for (int i=a_index; i<aa; i++){
62         a[a_index++] = first_list[i];
63     }
64 }
65
66 free(first_list);
67 free(second_list);
68 return a;
69
70 }
71 }
```

Unique Binary Search Trees - L... All Elements in Two Binary Sea... G-Student

leetcode.com/problems/all-elements-in-two-binary-search-trees/

Problem List < > < > < >

Description Accepted x Editorial Solutions Submissions

All Submissions

Accepted 48 / 48 testcases passed

Bobba... submitted at Mar 12, 2025 10:42

Editorial Solution

Runtime

12 ms | Beats: 12.50%

Analyze Complexity

Memory

61.64 MB | Beats: 60.00%

30% 20% 10% 0%

2ms 4ms 6ms 8ms 10ms 12ms 14ms 16ms 18ms 20ms 22ms 24ms 26ms 28ms 30ms

Code C

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     struct TreeNode *left;
 *     struct TreeNode *right;
 * };
 */
```

Code

```
12 void node_size(struct TreeNode *root, int *value){
13     if (!root){
14         return;
15     }
16     (*value)++;
17     node_size(root->left, value);
18     node_size(root->right, value);
19 }
20 void list(struct TreeNode* root1, int* index, int *array){
```

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

root1 =

[2, 1, 4]

root2 =

[1, 0, 3]

Output

[0, 1, 1, 2, 3, 4]

Expected

[0, 1, 1, 2, 3, 4]

30°C Haze

Search

ENG IN

11:27 12-03-2025

Unique Binary Search Trees - L... All Elements in Two Binary Sea... G-Student

leetcode.com/problems/all-elements-in-two-binary-search-trees/

Problem List < > < > < >

Description Accepted x Editorial Solutions Submissions

All Submissions

Accepted 48 / 48 testcases passed

Bobba... submitted at Mar 12, 2025 10:42

Editorial Solution

Runtime

12 ms | Beats: 12.50%

Analyze Complexity

Memory

61.64 MB | Beats: 60.00%

30% 20% 10% 0%

2ms 4ms 6ms 8ms 10ms 12ms 14ms 16ms 18ms 20ms 22ms 24ms 26ms 28ms 30ms

Code C

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     struct TreeNode *left;
 *     struct TreeNode *right;
 * };
 */
```

Code

```
12 void node_size(struct TreeNode *root, int *value){
13     if (!root){
14         return;
15     }
16     (*value)++;
17     node_size(root->left, value);
18     node_size(root->right, value);
19 }
20 void list(struct TreeNode* root1, int* index, int *array){
```

Testcase Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

root1 =

[1, null, 8]

root2 =

[8, 1]

Output

[1, 1, 8, 8]

Expected

[1, 1, 8, 8]

30°C Haze

Search

ENG IN

11:27 12-03-2025