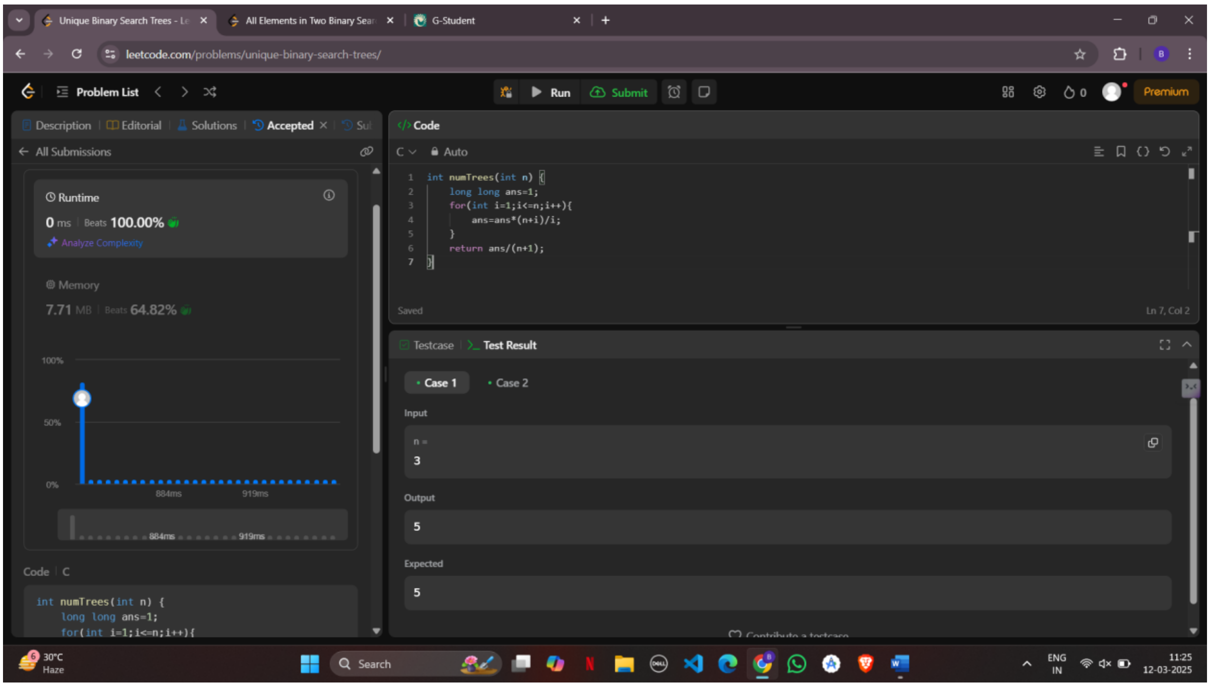
# ASSIGNMENT-4

Bobba Mahidhar
VU21CSEN0300110
CSE-AIML

## Unique Binary Search Trees



Runtime
0 ms Beats 100.00%

Memory
7.71 MB Beats 64.82%

Code | C

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

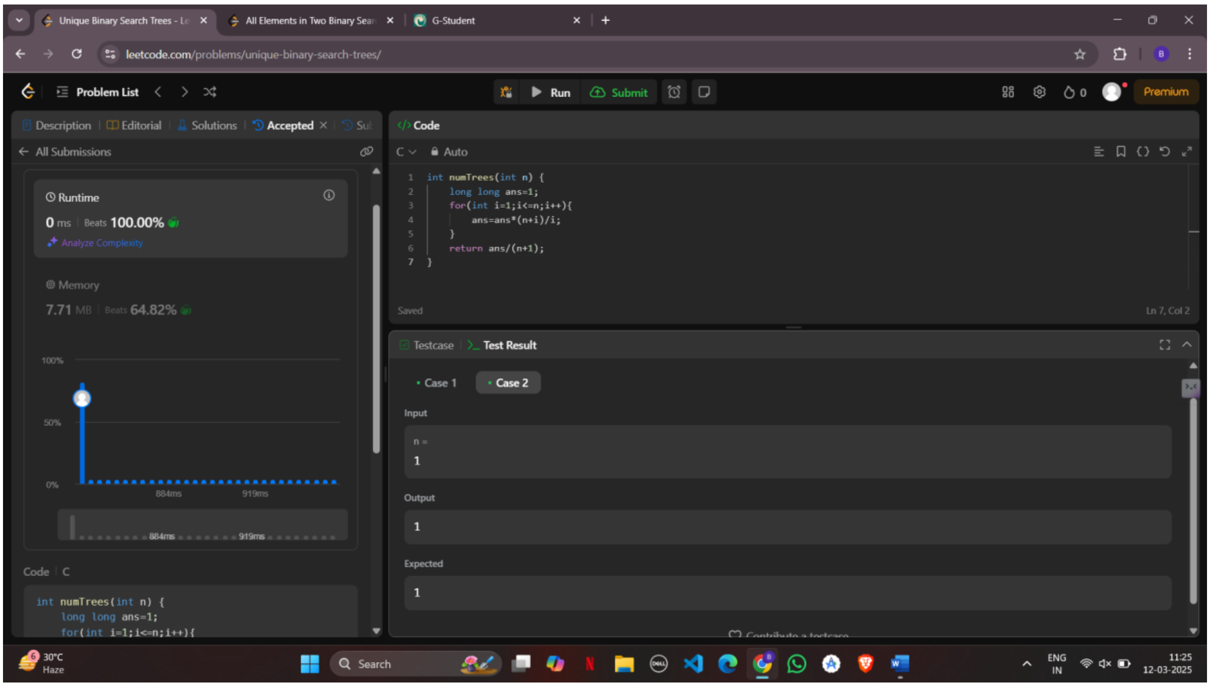
Saved Ln 7, Col 2

Case 1 Case 2

Input
n =
3

Output
5

Expected
5



Runtime
0 ms Beats 100.00%

Memory
7.71 MB Beats 64.82%

Code | C

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

Saved Ln 7, Col 2

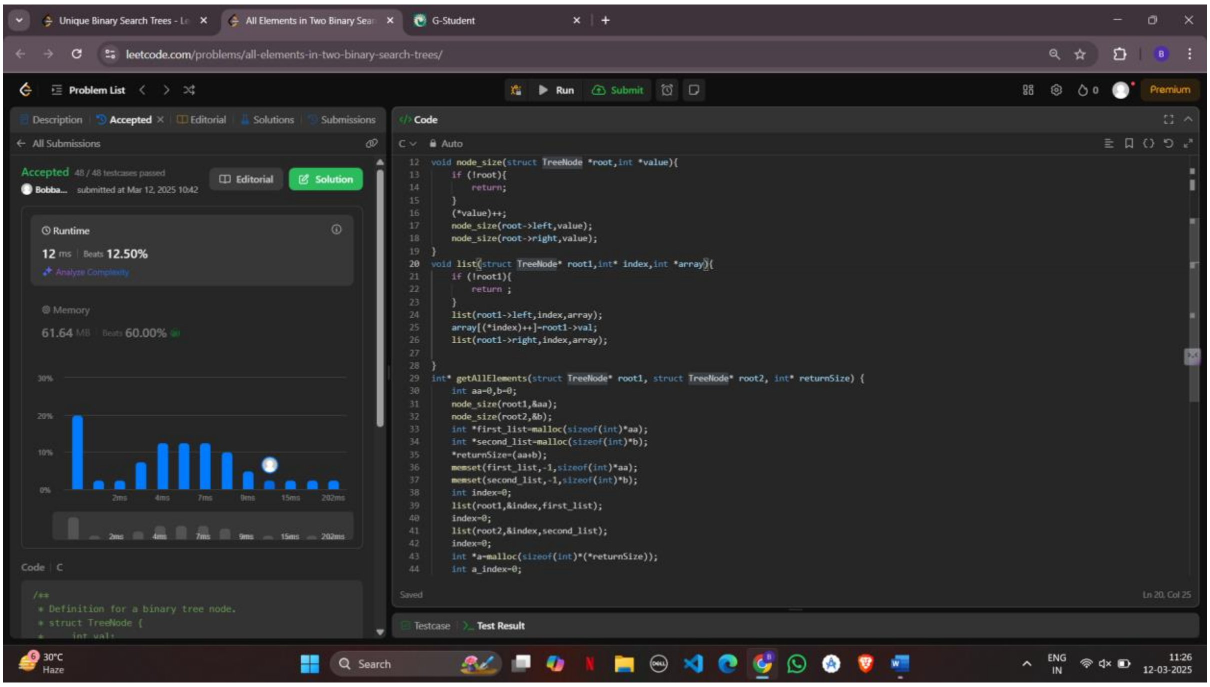
Case 1 Case 2

Input
n =
1

Output
1

Expected
1

# All Elements in Two Binary Search Trees



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

Accepted 48 / 48 testcases passed
Bobba... submitted at Mar 12, 2025 10:42

Runtime
12 ms | Beats 12.50%

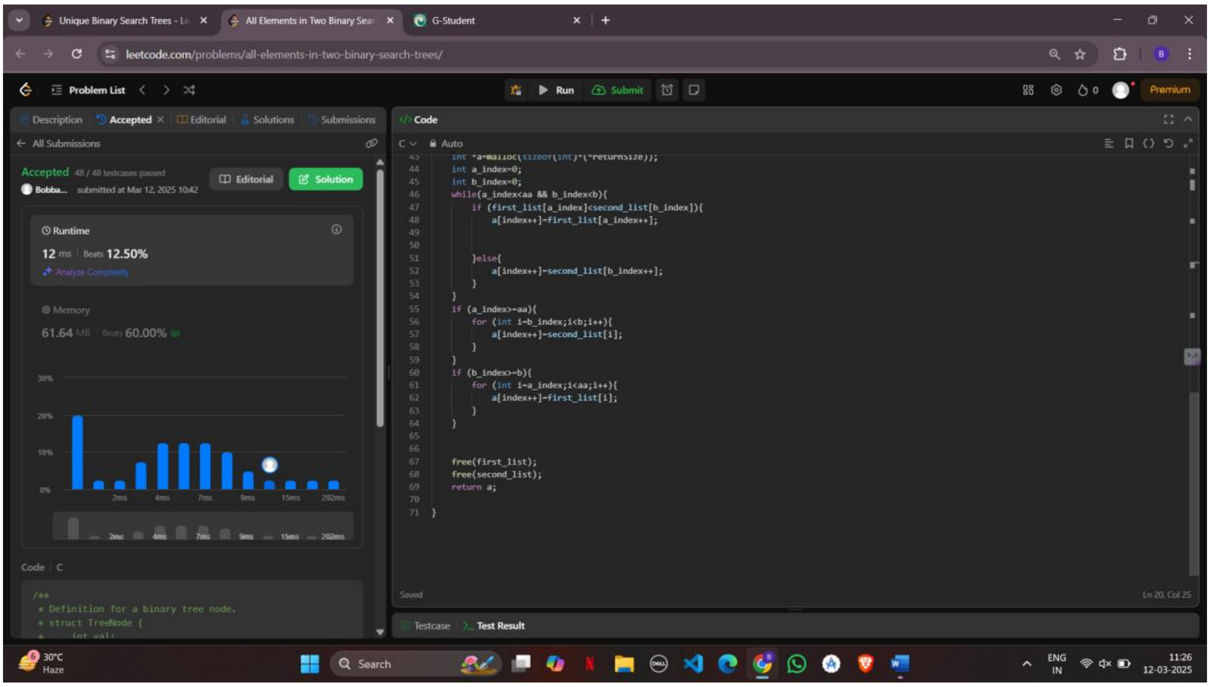
Memory
61.64 MB | Beats 60.00%

Code | C

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

```c
void node_size(struct TreeNode *root,int *value){
    if (!root){
        return;
    }
    (*value)++;
    node_size(root->left,value);
    node_size(root->right,value);
}
void list(struct TreeNode* root1,int* index,int *array){
    if (!root1){
        return ;
    }
    list(root1->left,index,array);
    array[(*index)++]=root1->val;
    list(root1->right,index,array);

}
int* getAllElements(struct TreeNode* root1, struct TreeNode* root2, int* returnSize) {
    int aa=0,b=0;
    node_size(root1,&aa);
    node_size(root2,&b);
    int *first_list=malloc(sizeof(int)*aa);
    int *second_list=malloc(sizeof(int)*b);
    *returnSize=(aa+b);
    memset(first_list,-1,sizeof(int)*aa);
    memset(second_list,-1,sizeof(int)*b);
    int index=0;
    list(root1,&index,first_list);
    index=0;
    list(root2,&index,second_list);
    index=0;
    int *a=malloc(sizeof(int)*(*returnSize));
    int a_index=0;
```



Accepted 48 / 48 testcases passed
Bobba... submitted at Mar 12, 2025 10:42

Runtime
12 ms | Beats 12.50%

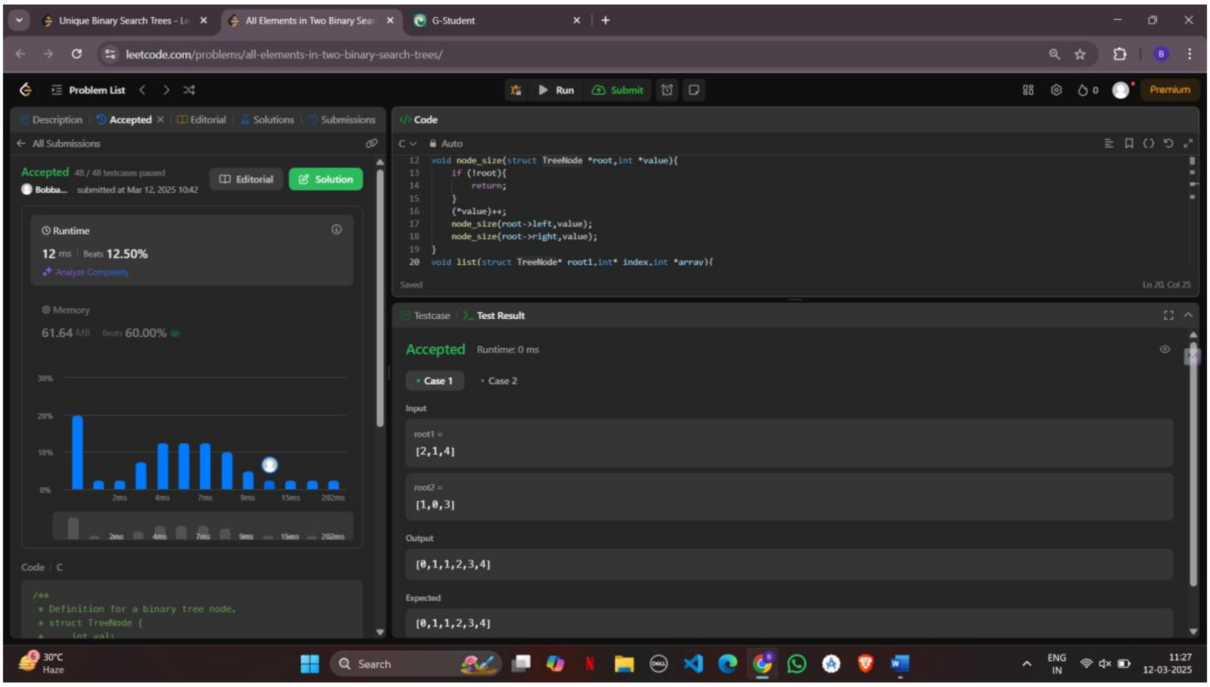
Memory
61.64 MB | Beats 60.00%

```c
    int a_index=0;
    int b_index=0;
    while(a_index<aa && b_index<b){
        if (first_list[a_index]<second_list[b_index]){
            a[index++]=first_list[a_index++];


        }else{
            a[index++]=second_list[b_index++];
        }
    }
    if (a_index>=aa){
        for (int i=b_index;i<b;i++){
            a[index++]=second_list[i];
        }
    }
    if (b_index>=b){
        for (int i=a_index;i<aa;i++){
            a[index++]=first_list[i];
        }
    }


    free(first_list);
    free(second_list);
    return a;

}
```



Accepted 48 / 48 testcases passed

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

Runtime

12 ms | Beats 12.50%

Memory

61.64 MB | Beats 60.00%

Code | C

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

```
void node_size(struct TreeNode *root,int *value){
    if (!root){
        return;
    }
    (*value)++;
    node_size(root->left,value);
    node_size(root->right,value);
}
void list(struct TreeNode* root1,int* index,int *array){
```

Accepted Runtime: 0 ms

Case 1 · Case 2

Input

root1 =

```
[2,1,4]
```

root2 =

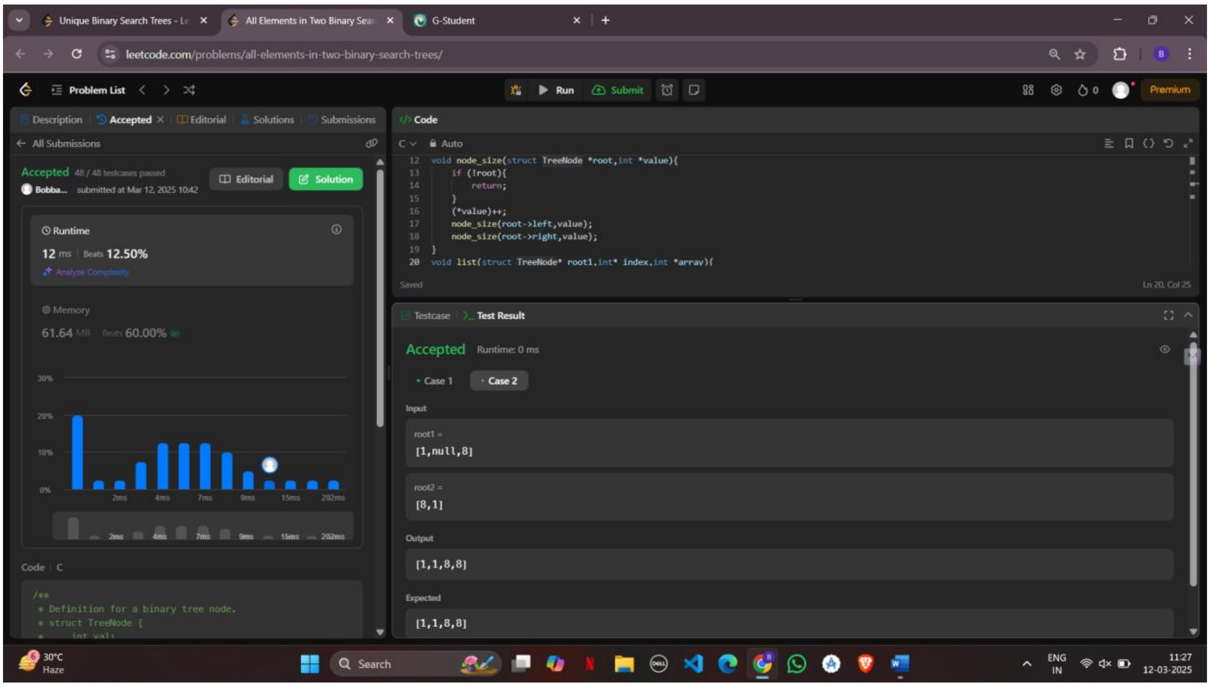
```
[1,0,3]
```

Output

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

Expected

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



Accepted 48 / 48 testcases passed

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

Runtime

12 ms | Beats 12.50%

Memory

61.64 MB | Beats 60.00%

Code | C

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

```
void node_size(struct TreeNode *root,int *value){
    if (!root){
        return;
    }
    (*value)++;
    node_size(root->left,value);
    node_size(root->right,value);
}
void list(struct TreeNode* root1,int* index,int *array){
```

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]
```