Problem List

Submit

Premium

Description | Accepted | Editorial | Solutions | Submissions

All Submissions

Accepted 182 / 182 testcases passed

Shiwani_Singh submitted at Dec 22, 2025 00:15

Editorial | Solution

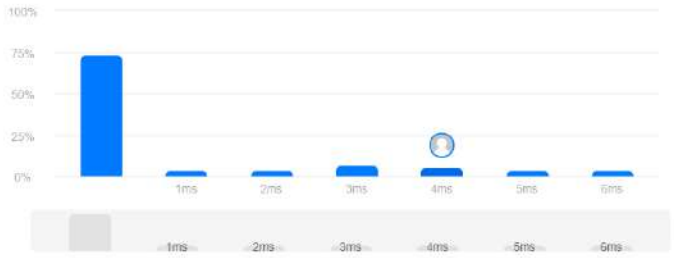
Runtime

4 ms | Beats 14.53%

Analyze Complexity

Memory

33.57 MB | Beats 50.93%



Code | C++

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

Code

C++ | Auto

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    TreeNode* mergeTrees(TreeNode* root1, TreeNode* root2) {
        if (root1 == NULL && root2 == NULL)
            return NULL;

        // If one is null, return the other
        if (root1 == NULL)
            return root2;

        if (root2 == NULL)
            return root1;

        // Both are not null
        TreeNode* node = new TreeNode(root1->val + root2->val);

        node->left = mergeTrees(root1->left, root2->left);
```

Saved

Ln 31, Col 21

Testcase | Test Result

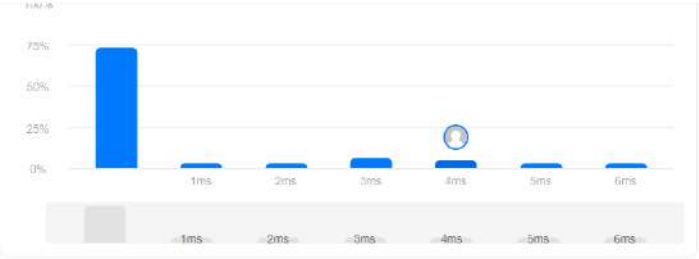
Problem List

Submit

Premium

Description | Accepted | Editorial | Solutions | Submissions

All Submissions



Code | C++

```cpp
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     TreeNode *left;
 *     TreeNode *right;
 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left)
 * };
 */
class Solution {
public:
    TreeNode* mergeTrees(TreeNode* root1, TreeNode* root2) {
        if (root1 == NULL && root2 == NULL)
            return NULL;
```

Code

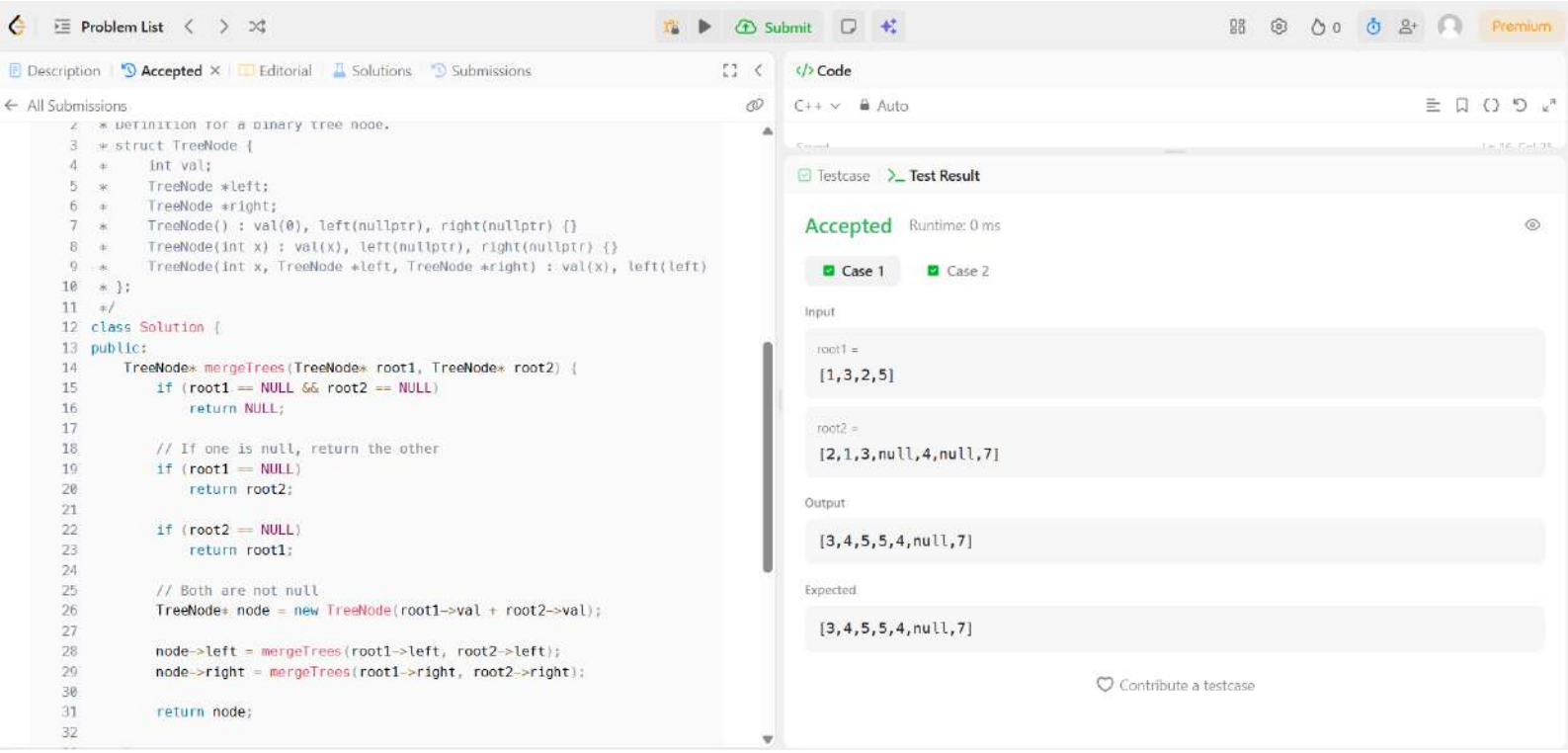
C++ | Auto

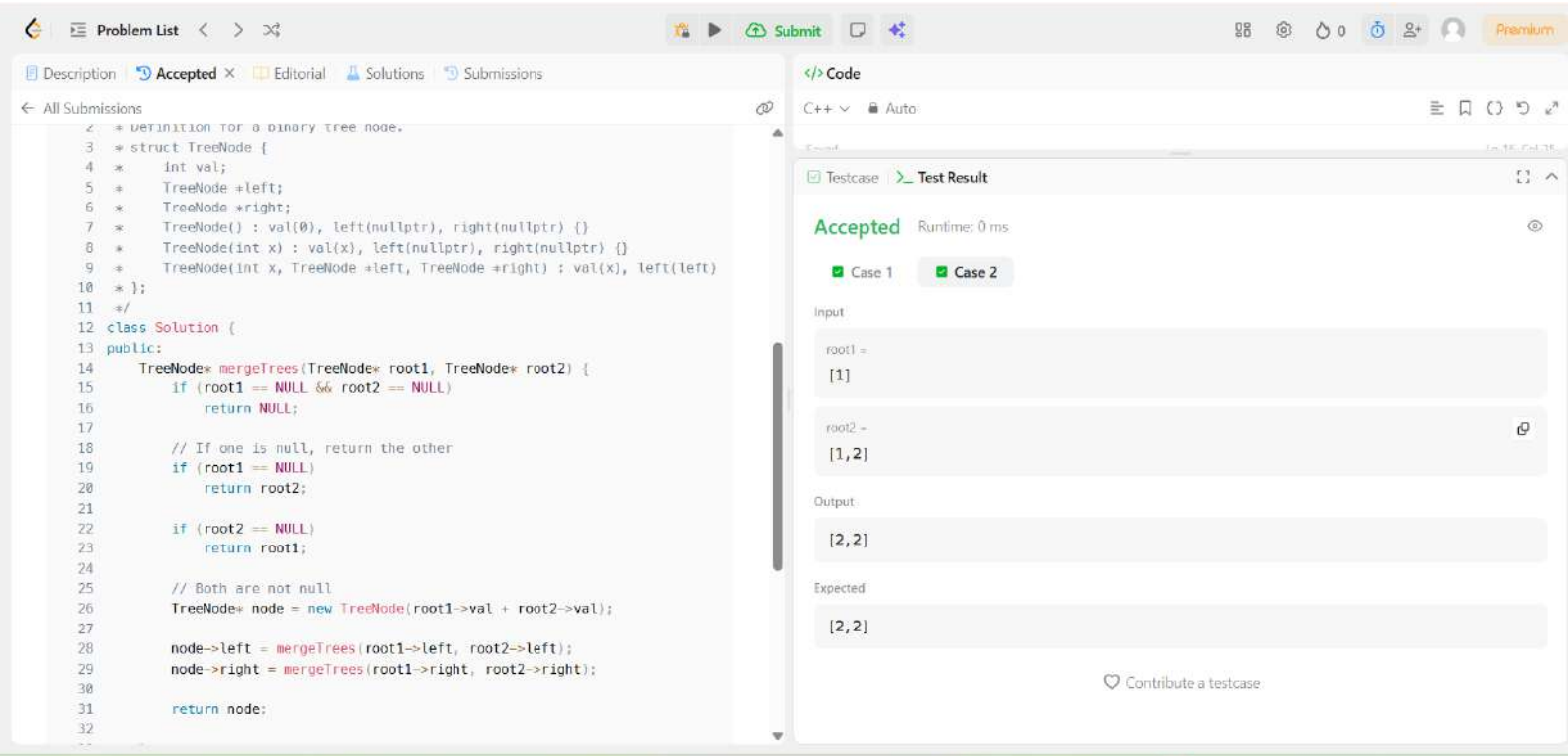
```cpp
 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
 * };
 */
class Solution {
public:
    TreeNode* mergeTrees(TreeNode* root1, TreeNode* root2) {
        if (root1 == NULL && root2 == NULL)
            return NULL;

        // If one is null, return the other
        if (root1 == NULL)
            return root2;

        if (root2 == NULL)
            return root1;

        // Both are not null
        TreeNode* node = new TreeNode(root1->val + root2->val);

        node->left = mergeTrees(root1->left, root2->left);
        node->right = mergeTrees(root1->right, root2->right);

        return node;

    }
};
```

Saved

Ln 16, Col 25

Testcase | Test Result


Problem List
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Code
C++
Auto
/**
* Definition for a binary tree node.
* struct TreeNode {
*   int val;
*   TreeNode *left;
*   TreeNode *right;
*   TreeNode() : val(0), left(nullptr), right(nullptr) {}
*   TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
*   TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left)
* };
*/
class Solution {
public:
TreeNode* mergeTrees(TreeNode* root1, TreeNode* root2) {
if (root1 == NULL && root2 == NULL)
return NULL;
// If one is null, return the other
if (root1 == NULL)
return root2;
if (root2 == NULL)
return root1;
// Both are not null
TreeNode* node = new TreeNode(root1->val + root2->val);
node->left = mergeTrees(root1->left, root2->left);
node->right = mergeTrees(root1->right, root2->right);
return node;
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
root1 =
[1,3,2,5]
root2 =
[2,1,3,null,4,null,7]
Output
[3,4,5,5,4,null,7]
Expected
[3,4,5,5,4,null,7]
Contribute a testcase


Problem List
Submit
Premium
Description
Accepted
Editorial
Solutions
Submissions
All Submissions
Code
C++
Auto
/**
* Definition for a binary tree node.
* struct TreeNode {
*   int val;
*   TreeNode *left;
*   TreeNode *right;
*   TreeNode() : val(0), left(nullptr), right(nullptr) {}
*   TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
*   TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left)
* };
*/
class Solution {
public:
TreeNode* mergeTrees(TreeNode* root1, TreeNode* root2) {
if (root1 == NULL && root2 == NULL)
return NULL;
// If one is null, return the other
if (root1 == NULL)
return root2;
if (root2 == NULL)
return root1;
// Both are not null
TreeNode* node = new TreeNode(root1->val + root2->val);
node->left = mergeTrees(root1->left, root2->left);
node->right = mergeTrees(root1->right, root2->right);
return node;
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Input
root1 =
[1]
root2 =
[1,2]
Output
[2,2]
Expected
[2,2]
Contribute a testcase