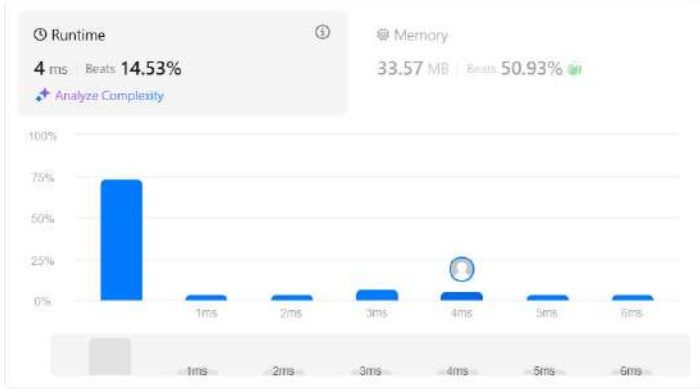


< All Submissions

Accepted 182 / 182 testcases passed
Shiwani_Singh submitted at Dec 22, 2025 00:15

Editorial Solution



Code C++

```
1 /**
2  * Definition for a binary tree node.
3  * struct TreeNode {
4  *     int val;
5  *     TreeNode *left;
```

```
1 /**
2  * Definition for a binary tree node.
3  * struct TreeNode {
4  *     int val;
5  *     TreeNode *left;
6  *     TreeNode *right;
7  *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
8  *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
9  *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
10  * };
11 */
12 class Solution {
13 public:
14     TreeNode* mergeTrees(TreeNode* root1, TreeNode* root2) {
15         if (root1 == NULL && root2 == NULL)
16             return NULL;
17
18         // If one is null, return the other
19         if (root1 == NULL)
20             return root2;
21
22         if (root2 == NULL)
23             return root1;
24
25         // Both are not null
26         TreeNode* node = new TreeNode(root1->val + root2->val);
27
28         node->left = mergeTrees(root1->left, root2->left);
```

Problem List

Accepted

Editorial

Solutions

Submissions

All Submissions

75%

60%

25%

0%

1ms

2ms

3ms

4ms

5ms

6ms

1ms

2ms

3ms

4ms

5ms

6ms

Code

C++

```
1 /**
2  * Definition for a binary tree node.
3  * struct TreeNode {
4  *     int val;
5  *     TreeNode *left;
6  *     TreeNode *right;
7  *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
8  *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
9  *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
10  * };
11  */
12 class Solution {
13 public:
14     TreeNode* mergeTrees(TreeNode* root1, TreeNode* root2) {
15         if (root1 == NULL && root2 == NULL)
16             return NULL;
17
18         // If one is null, return the other
19         if (root1 == NULL)
20             return root2;
21         if (root2 == NULL)
22             return root1;
23
24         // Both are not null
25         TreeNode* node = new TreeNode(root1->val + root2->val);
26         node->left = mergeTrees(root1->left, root2->left);
27         node->right = mergeTrees(root1->right, root2->right);
28
29         return node;
30     }
31 };
32
33
34
```

Code

C++

```
8  *   TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
9  *   TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left), right(right) {}
10 * };
11 */
12 class Solution {
13 public:
14     TreeNode* mergeTrees(TreeNode* root1, TreeNode* root2) {
15         if (root1 == NULL && root2 == NULL)
16             return NULL;
17
18         // If one is null, return the other
19         if (root1 == NULL)
20             return root2;
21         if (root2 == NULL)
22             return root1;
23
24         // Both are not null
25         TreeNode* node = new TreeNode(root1->val + root2->val);
26         node->left = mergeTrees(root1->left, root2->left);
27         node->right = mergeTrees(root1->right, root2->right);
28
29         return node;
30     }
31 };
32
33
34
```

Testcase

Test Result

Problem List

Description

Accepted

Editorial

Solutions

Submissions

Submit

0

Premium

All Submissions

```
2 * Definition for a binary tree node.
3 * struct TreeNode {
4 *     int val;
5 *     TreeNode *left;
6 *     TreeNode *right;
7 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
8 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
9 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left)
10 * };
11 */
12 class Solution {
13 public:
14     TreeNode* mergeTrees(TreeNode* root1, TreeNode* root2) {
15         if (root1 == NULL && root2 == NULL)
16             return NULL;
17
18         // If one is null, return the other
19         if (root1 == NULL)
20             return root2;
21
22         if (root2 == NULL)
23             return root1;
24
25         // Both are not null
26         TreeNode* node = new TreeNode(root1->val + root2->val);
27
28         node->left = mergeTrees(root1->left, root2->left);
29         node->right = mergeTrees(root1->right, root2->right);
30
31         return node;
32     }
33 }
```

Code

C++

Auto

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

📄

🚀

🔧 ⚙️ 🔥 0 🔄 👤 Premium

Description Accepted ✕ Editorial Solutions Submissions

< All Submissions

```
1 * Definition for a binary tree node.
2 * struct TreeNode {
3 *     int val;
4 *     TreeNode *left;
5 *     TreeNode *right;
6 *     TreeNode() : val(0), left(nullptr), right(nullptr) {}
7 *     TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
8 *     TreeNode(int x, TreeNode *left, TreeNode *right) : val(x), left(left)
9 * };
10 */
11
12 class Solution {
13 public:
14     TreeNode* mergeTrees(TreeNode* root1, TreeNode* root2) {
15         if (root1 == NULL && root2 == NULL)
16             return NULL;
17
18         // If one is null, return the other
19         if (root1 == NULL)
20             return root2;
21
22         if (root2 == NULL)
23             return root1;
24
25         // Both are not null
26         TreeNode* node = new TreeNode(root1->val + root2->val);
27
28         node->left = mergeTrees(root1->left, root2->left);
29         node->right = mergeTrees(root1->right, root2->right);
30
31         return node;
32     }
33 }
```

</> Code

C++ ▾ 📄 Auto

🔍 📖 🔄 ↺ 🚀

📄 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