Problem List

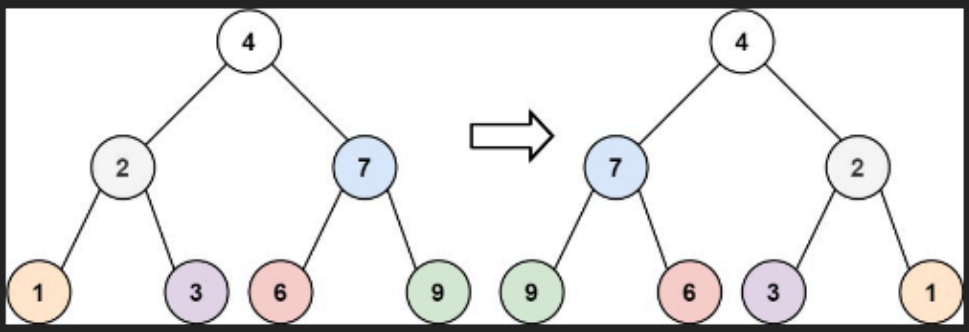
Description | Editorial | Solutions | Submissions

226. Invert Binary Tree

Easy | Topics | Companies

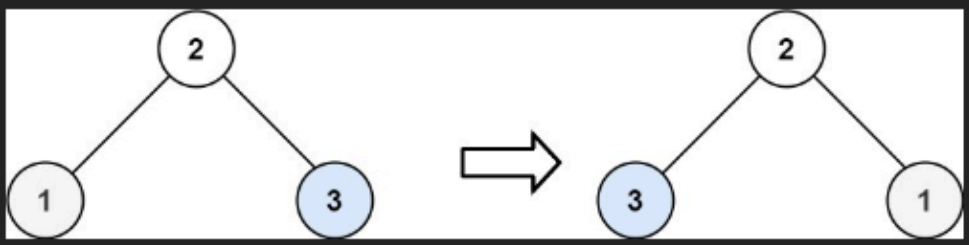
Given the `root` of a binary tree, invert the tree, and return *its root*.

Example 1:




```
Input: root = [4,2,7,1,3,6,9]
Output: [4,7,2,9,6,3,1]
```

Example 2:




```
Input: root = [2,1,3]
Output: [2,3,1]
```

Example 3:

```
Input: root = []
Output: []
```

13.6K | 108

Code

C Auto

```c
 *     struct TreeNode *right;
 * };
 */
struct TreeNode* invertTree(struct TreeNode* root) {
    if(root==NULL)
        return NULL;
        invertTree(root->left); //Call the left substree
        invertTree(root->right); //Call the right substree
        // Swap the nodes
        struct TreeNode* temp = root->left;
        root->left = root->right;
        root->right = temp;
        return root; // Return the root
}

```

Saved to local

Testcase | Test Result

Accepted Runtime: 6 ms

• Case 1 • Case 2 • Case 3

Input

root =

[4,2,7,1,3,6,9]

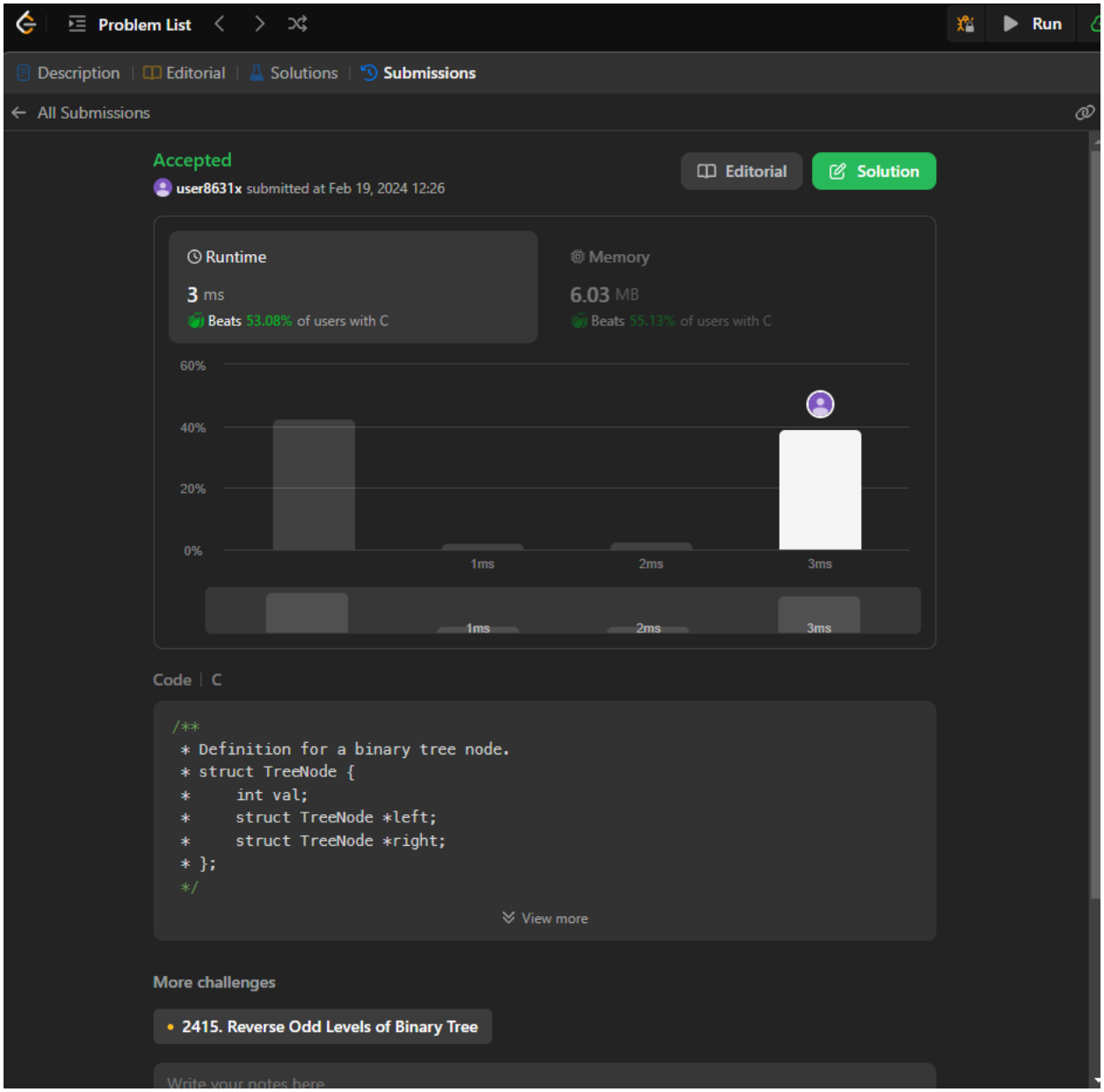
Output

[4,7,2,9,6,3,1]

Expected

[4,7,2,9,6,3,1]

Contribute a testcase


Problem List
Run
Description
Editorial
Solutions
Submissions
All Submissions
Accepted
user8631x submitted at Feb 19, 2024 12:26
Editorial
Solution
Runtime
3 ms
Beats 53.08% of users with C
Memory
6.03 MB
Beats 55.13% of users with C
60%
40%
20%
0%
1ms
2ms
3ms
Code | C
/**
* Definition for a binary tree node.
* struct TreeNode {
*   int val;
*   struct TreeNode *left;
*   struct TreeNode *right;
* };
*/
View more
More challenges
2415. Reverse Odd Levels of Binary Tree
Write your notes here

Code:

```c
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     struct TreeNode *left;
 *     struct TreeNode *right;
 * };
 */
struct TreeNode* invertTree(struct TreeNode* root) {
    if(root==NULL)
        return NULL;
        invertTree(root->left); //Call the left substree
        invertTree(root->right); //Call the right substree
        // Swap the nodes
        struct TreeNode* temp = root->left;
        root->left = root->right;
        root->right = temp;
        return root; // Return the root
}
```