SymmetricTree

Joel Boynton

October 22, 2025

1 Problem Description

In this problem we are given a **root** of a binary tree. We are to write a function that determines whether the tree mirrors itself. That is to say if a parent **1** splits into two children **2** they should read as [2,2] if these two children split into two children each **3** and **4** the two sides should read as [3,4] and [4,3]; directly mirroring the other side. As the name implies, a parent in a binary tree system should only have 1 or 2 children.

2 Solution Description

There are a few things that need to be checked when going through the tree. The approach that I want to take is recursive in nature. First we check if the root of the tree is empty. If so then the tree is symmetric by definition. Otherwise, we will need to check for a couple of other identifiers. To do this I will use a function to check if both sides of the tree mirror each other. This will check a few things first. whether they are both empty or if just one is empty (indicating whether it is symmetric or not). Finally it will check (while both nodes exist) if the current nodes mirror each other. This should yield a time complexity of O(n).

3 Initial Attempt to Code the Solution

The code is a rather simple recursive program. I first start by reading a check on our tree root, saying that if it is false (meaning empty) then it returns true. This tells us that the binary tree satisfies the symmetric rules. I then move on to a lambda function I call *isMirror*. This recursively checks if our left and right tree nodes are mirrors of eachother. The first two basic checks are as follows. if left AND right are both null then, again we return true. Since they are both empty, this is obviously symmetric. The next checks if only one or the other is null; if so return false. These would obviously not be considered symmetric as one does not reflect the other (one is empty, one is not). Then the function returns the final check to determine if both sides of the tree are mirrors of eachother. I do this by checking three things while we know that both nodes exist. First, the current noes must hold the same value. Second, the outer subtrees must be mirros of the other. Third, the inner subtrees must also, be mirrors of each other. If all are true. the the pair are considered symmetric. The recursive checking process begins with the left and right subtrees of the root. This is done until the program is

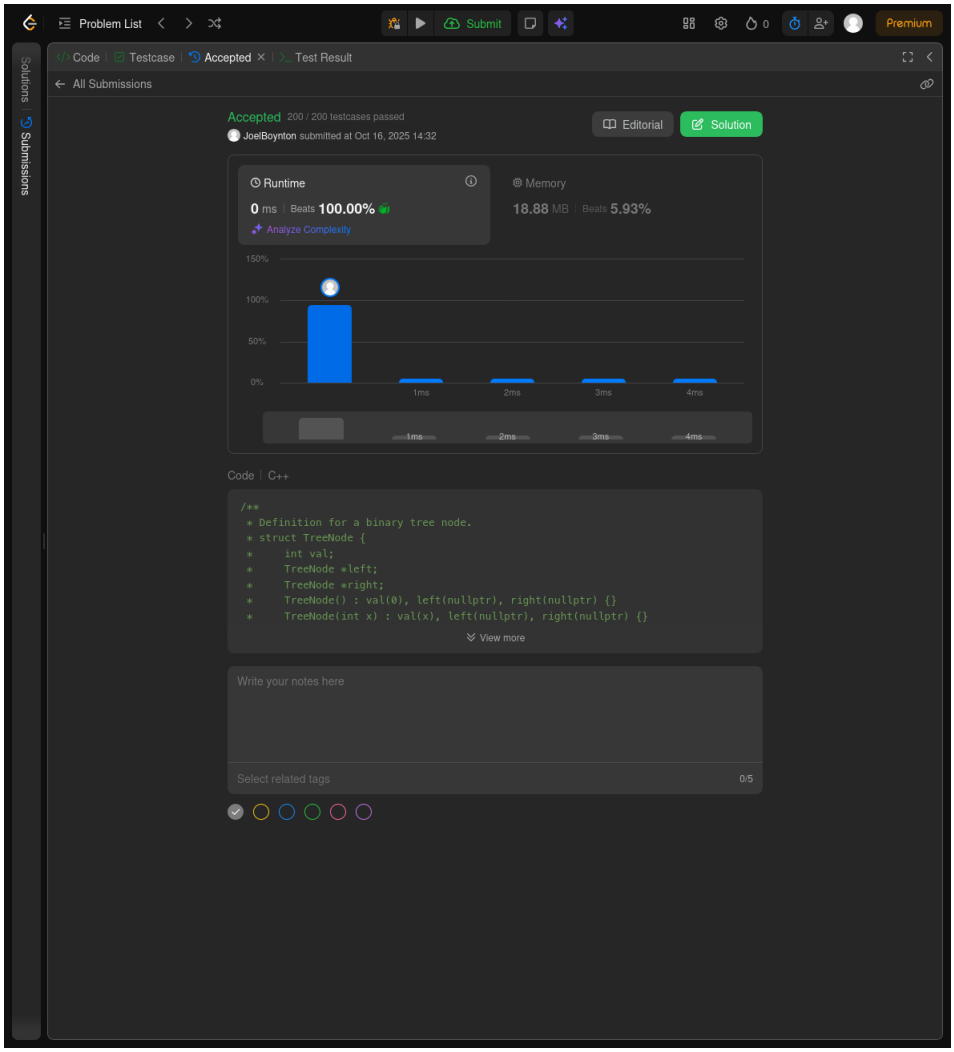
complete.

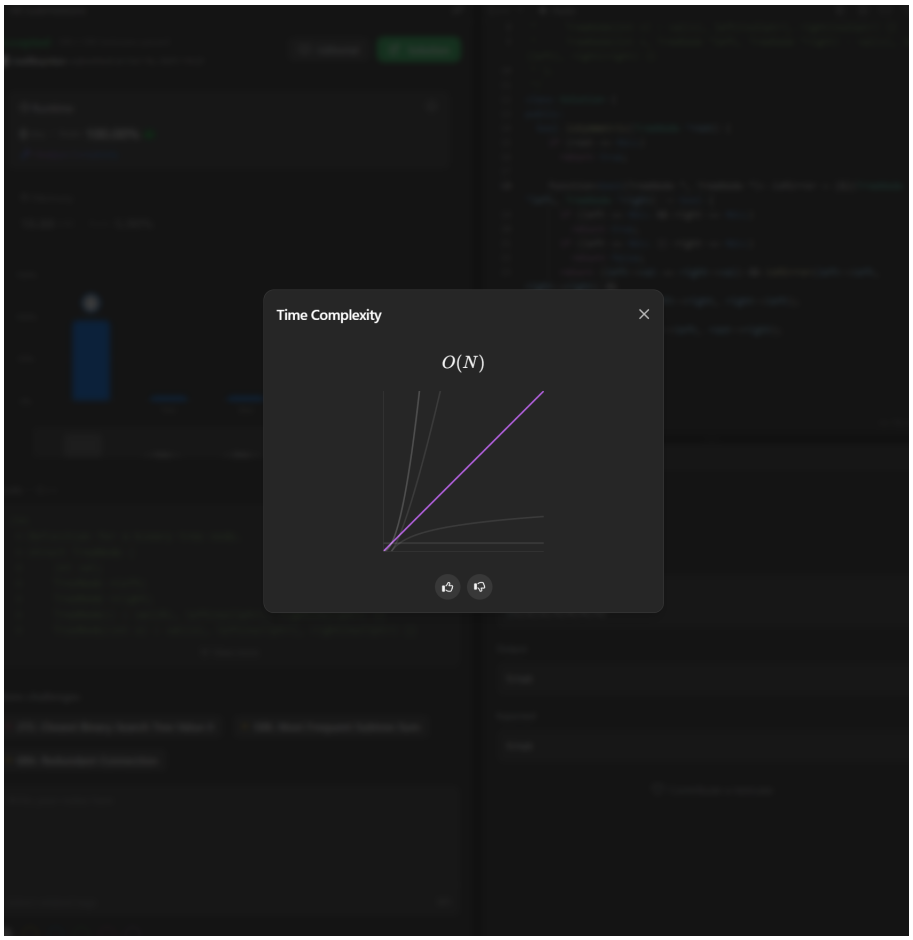
4 Testng Description

Leet Code problem–Testing is done with built in test cases and submission tests. *Results in images below*

5 Code & Testing

```
class Solution {
  public:
  bool isSymmetric(TreeNode* root){
    if (root == NULL) return true;
    function<bool(TreeNode*, TreeNode*)> isMirror =
      [&](TreeNode* left, TreeNode* right) -> bool {
      if(left == NULL && right == NULL) return true;
      if(left == NULL || right == NULL) return false;
      return(left->val == right->val) &&
        isMirror(left->left, right->right) &&
        isMirror(left->right, right->left);
    };
    return isMirror(root->left, root->right);
  }
};
```


Problem List
Submit
Premium
Code
Testcase
Accepted
Test Result
All Submissions
Solutions
Submissions
Accepted 200 / 200 testcases passed
JoelBoynton submitted at Oct 16, 2025 14:32
Editorial
Solution
Runtime
0 ms | Beats 100.00%
Analyze Complexity
Memory
18.88 MB | Beats 5.93%
150%
100%
50%
0%
1ms
2ms
3ms
4ms
Code | C++
/**
* 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) {}
View more
Write your notes here
Select related tags
0/5


Time Complexity
$O(N)$