Write Code to Determine if Two Trees are Identical

Two trees are identical when they have same data and arrangement of data is also same.

To identify if two trees are identical, we need to traverse both trees simultaneously, and while traversing we need to compare data and children of the trees.

Algorithm:

```
sameTree(tree1, tree2)
1. If both trees are empty then return 1.
2. Else If both trees are non -empty
     (a) Check data of the root nodes (tree1->data ==  tree2->data)
     (b) Check left subtrees recursively  i.e., call sameTree(
          tree1->left_subtree, tree2->left_subtree)
     (c) Check right subtrees recursively  i.e., call sameTree(
          tree1->right_subtree, tree2->right_subtree)
     (d) If a,b and c are true then return 1.
3  Else return 0 (one is empty and other is not)
```

C/C++

```
#include <stdio.h>
#include <stdlib.h>

/* A binary tree node has data, pointer to left child
   and a pointer to right child */
struct node
{
    int data;
    struct node* left;
    struct node* right;
};

/* Helper function that allocates a new node with the
   given data and NULL left and right pointers. */
struct node* newNode(int data)
{
    struct node* node = (struct node*)
                            malloc(sizeof(struct node));
    node->data  = data;
    node->left  = NULL;
    node->right = NULL;

    return(node);
}

/* Given two trees, return true if they are
 structurally identical */
int identicalTrees(struct node* a, struct node* b)
{
    /*1. both empty */
    if (a==NULL && b==NULL)
        return 1;

    /* 2. both non-empty -> compare them */
    if (a!=NULL && b!=NULL)
    {
        return
        (
            a->data == b->data &&
            identicalTrees(a->left, b->left) &&
            identicalTrees(a->right, b->right)
```

```
            identicalTrees(a->right, b->right)
        );
    }

    /* 3. one empty, one not -> false */
    return 0;
}

/* Driver program to test identicalTrees function*/
int main()
{
    struct node *root1 = newNode(1);
    struct node *root2 = newNode(1);
    root1->left = newNode(2);
    root1->right = newNode(3);
    root1->left->left  = newNode(4);
    root1->left->right = newNode(5);

    root2->left = newNode(2);
    root2->right = newNode(3);
    root2->left->left = newNode(4);
    root2->left->right = newNode(5);

    if(identicalTrees(root1, root2))
        printf("Both tree are identical.");
    else
        printf("Trees are not identical.");

    getchar();
  return 0;
}
```

Java

```
// Java program to see if two trees are identical

// A binary tree node
class Node
{
    int data;
    Node left, right;

    Node(int item)
    {
        data = item;
        left = right = null;
    }
}

class BinaryTree
{
    Node root1, root2;

    /* Given two trees, return true if they are
       structurally identical */
    boolean identicalTrees(Node a, Node b)
    {
        /*1. both empty */
        if (a == null && b == null)
            return true;

        /* 2. both non-empty -> compare them */
        if (a != null && b != null)
            return (a.data == b.data
                    && identicalTrees(a.left, b.left)
                    && identicalTrees(a.right, b.right));

        /* 3. one empty, one not -> false */
        return false;
    }

    /* Driver program to test identicalTrees() function */
    public static void main(String[] args)
    {
        BinaryTree tree = new BinaryTree();

        tree.root1 = new Node(1);
        tree.root1.left = new Node(2);
        tree.root1.right = new Node(3);
        tree.root1.left.left = new Node(4);
        tree.root1.left.right = new Node(5);

        tree.root2 = new Node(1);
        tree.root2.left = new Node(2);
        tree.root2.right = new Node(3);
        tree.root2.left.left = new Node(4);
        tree.root2.left.right = new Node(5);

        if (tree.identicalTrees(tree.root1, tree.root2))
            System.out.println("Both trees are identical");
        else
            System.out.println("Trees are not identical");

    }
}
```

Python

```
# Python program to determine if two trees are identical

# A binary tree node has data, pointer to left child
# and a pointer to right child
class Node:
    # Constructor to create a new node
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None


# Given two trees, return true if they are structurally
# identical
def identicalTrees(a, b):

    # 1. Both empty
    if a is None and b is None:
        return True

    # 2. Both non-empty -> Compare them
    if a is not None and b is not None:
        return ((a.data == b.data) and
                identicalTrees(a.left, b.left)and
                identicalTrees(a.right, b.right))

    # 3. one empty, one not -- false
    return False

# Driver program to test identicalTress function
root1 = Node(1)
root2 = Node(1)
root1.left = Node(2)
root1.right = Node(3)
root1.left.left = Node(4)
root1.left.right = Node(5)

root2.left = Node(2)
root2.right = Node(3)
root2.left.left = Node(4)
root2.left.right = Node(5)

if identicalTrees(root1, root2):
    print "Both trees are identical"
else:
    print "Trees are not identical"

# This code is contributed by Nikhil Kumar Singh(nickzuck_007)
```

Time Complexity:

Complexity of the identicalTree() will be according to the tree with lesser number of nodes. Let number of nodes in two trees be m and n then complexity of sameTree() is O(m) where m < n. Iterative function to check if two trees are identical.