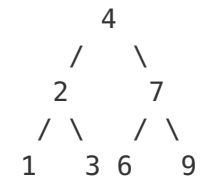
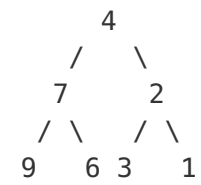
Invert Binary Tree

Invert a binary tree.



to



Trivia:
This problem was inspired by this original tweet by Max Howell:

> Google: 90% of our engineers use the software you wrote (Homebrew), but you can’t invert a binary tree on a whiteboard so fuck off.

Solution 1

As in many other cases this problem has more than one possible solutions:

Lets start with straightforward - recursive DFS - it's easy to write and pretty much concise.

```
public class Solution {
    public TreeNode invertTree(TreeNode root) {

        if (root == null) {
            return null;
        }

        final TreeNode left = root.left,
                right = root.right;
        root.left = invertTree(right);
        root.right = invertTree(left);
        return root;
    }
}
```

The above solution is correct, but it is also bound to the application stack, which means that it's no so much scalable - (you can find the problem size that will overflow the stack and crash your application), so more robust solution would be to use stack data structure.

```java
public class Solution {
    public TreeNode invertTree(TreeNode root) {

        if (root == null) {
            return null;
        }

        final Deque<TreeNode> stack = new LinkedList<>();
        stack.push(root);

        while(!stack.isEmpty()) {
            final TreeNode node = stack.pop();
            final TreeNode left = node.left;
            node.left = node.right;
            node.right = left;

            if(node.left != null) {
                stack.push(node.left);
            }
            if(node.right != null) {
                stack.push(node.right);
            }
        }
        return root;
    }
}
```

Finally we can easly convert the above solution to BFS - or so called level order traversal.

```java
public class Solution {
    public TreeNode invertTree(TreeNode root) {

        if (root == null) {
            return null;
        }

        final Queue<TreeNode> queue = new LinkedList<>();
        queue.offer(root);

        while(!queue.isEmpty()) {
            final TreeNode node = queue.poll();
            final TreeNode left = node.left;
            node.left = node.right;
            node.right = left;

            if(node.left != null) {
                queue.offer(node.left);
            }
            if(node.right != null) {
                queue.offer(node.right);
            }
        }
        return root;
    }
}
```

If I can write this code, does it mean I can get job at Google? ;)

written by jmnarloch original link here

Solution 2

Recursive

```
TreeNode* invertTree(TreeNode* root) {
    if (root) {
        invertTree(root->left);
        invertTree(root->right);
        std::swap(root->left, root->right);
    }
    return root;
}
```

Non-Recursive

```
TreeNode* invertTree(TreeNode* root) {
    std::stack<TreeNode*> stk;
    stk.push(root);

    while (!stk.empty()) {
        TreeNode* p = stk.top();
        stk.pop();
        if (p) {
            stk.push(p->left);
            stk.push(p->right);
            std::swap(p->left, p->right);
        }
    }
    return root;
}
```

written by chammika original link here

Solution 3

```
def invertTree(self, root):
    if root:
        root.left, root.right = self.invertTree(root.right), self.invertTree(root
.left)
        return root
```

Maybe make it four lines for better readability:

```
def invertTree(self, root):
    if root:
        invert = self.invertTree
        root.left, root.right = invert(root.right), invert(root.left)
        return root
```

And an iterative version using my own stack:

```
def invertTree(self, root):
    stack = [root]
    while stack:
        node = stack.pop()
        if node:
            node.left, node.right = node.right, node.left
            stack += node.left, node.right
    return root
```

written by StefanPochmann original link here