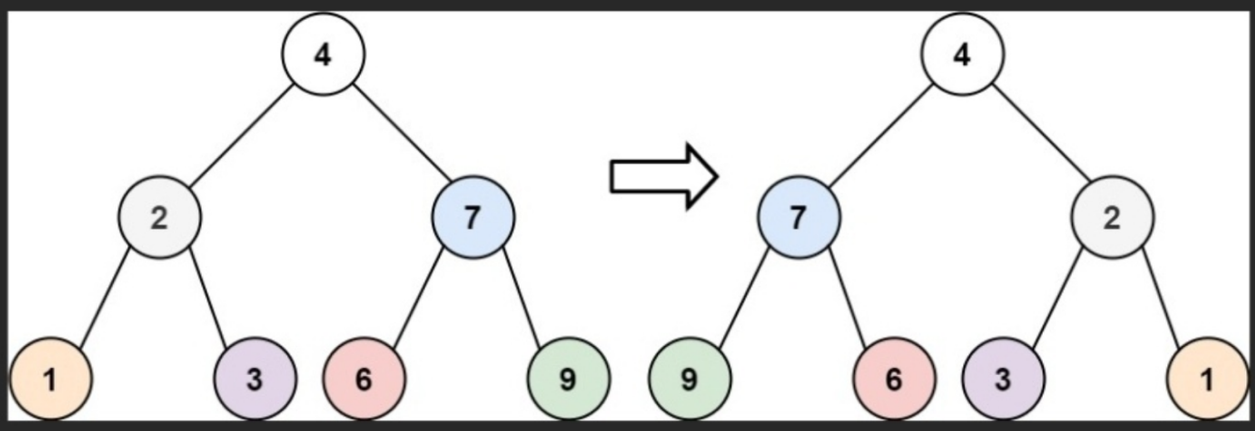
226. Invert Binary Tree

Easy 13K 182

Companies

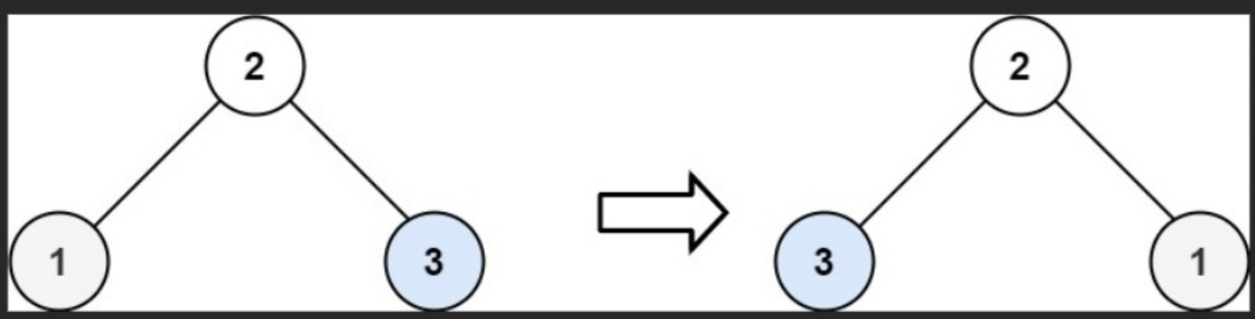
Given the `root` of a binary tree, invert the tree, and return *its root*.

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:

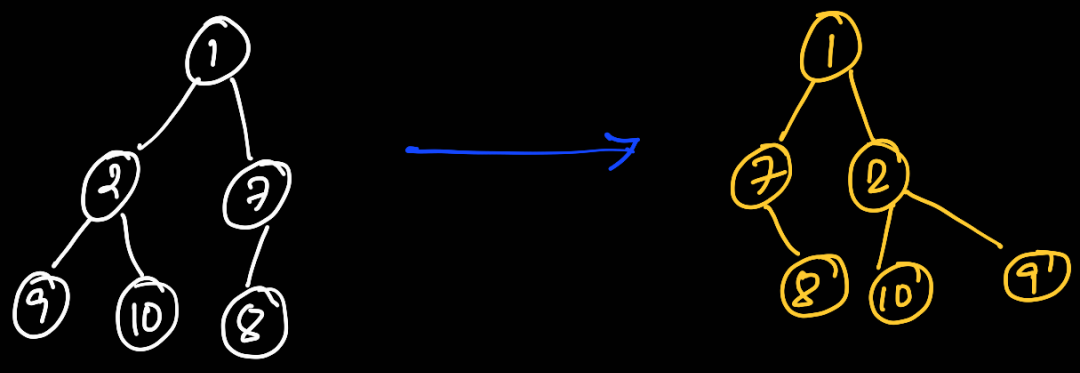
```
Input: root = []
Output: []
```

Constraints:

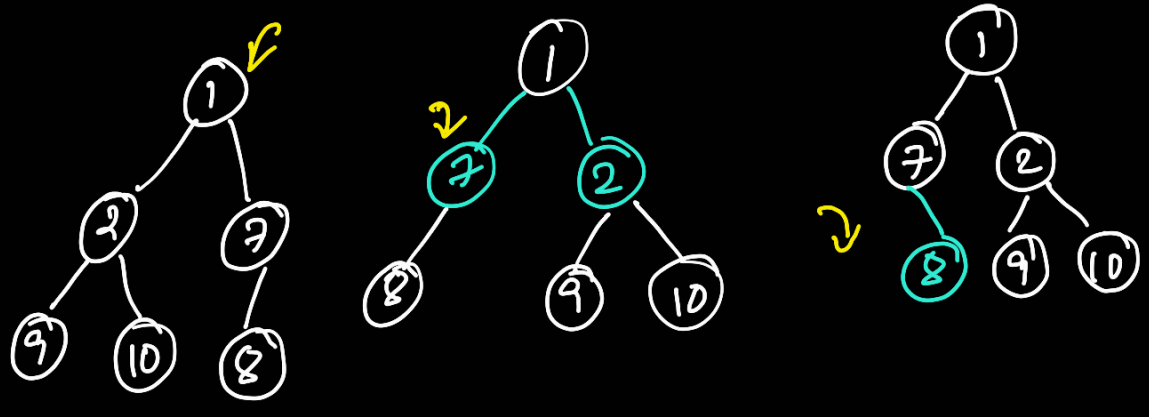
- The number of nodes in the tree is in the range `[0, 100]`.
- `-100 <= Node.val <= 100`

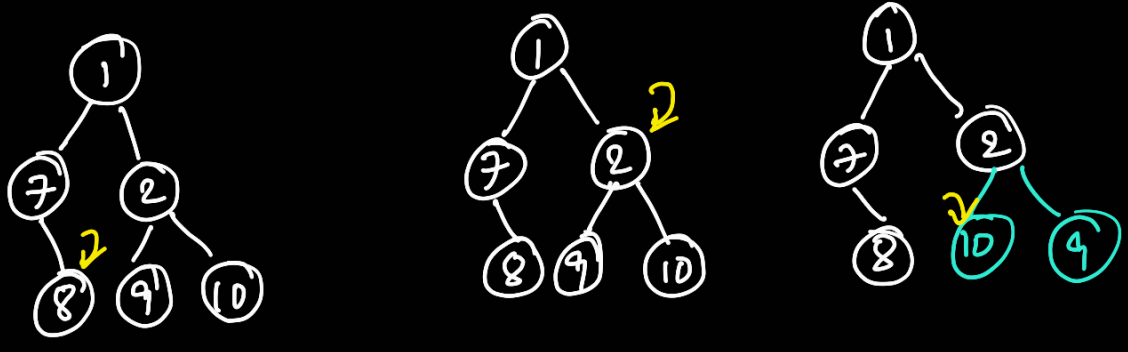
Accepted **1.7M** | Submissions **2.3M** | Acceptance Rate **75.6%**

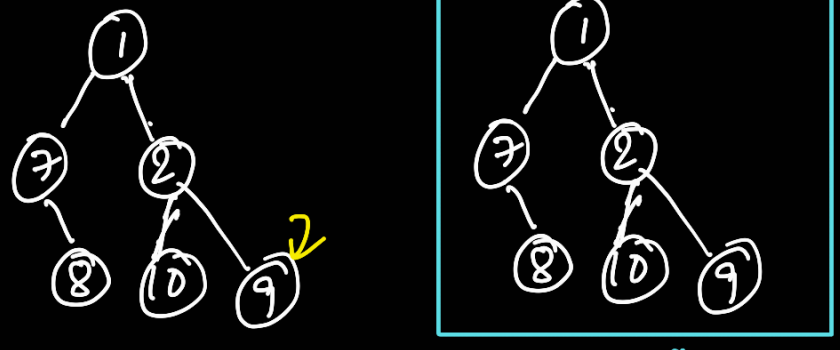
Approach 1:



If we see carefully at every node it is swapping left and right address.







inverted tree

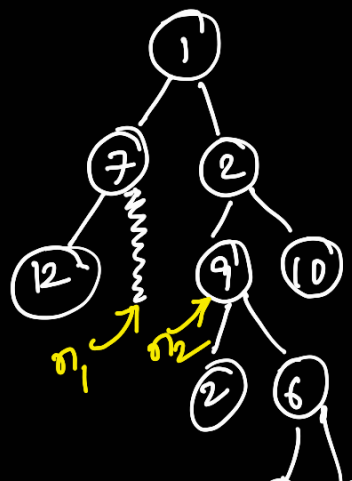
```
invert (node)
{
    if (node is null)
        return

    Swap (node->left , node->right)

    invert (node->left)
    invert (node->right)
}
```

$T(n) : O(n)$

note: we can also solve by swapping values but that would make the code really complicated and big to handle edge cases like
one is null and other is non null node



Approach 2:

Iterative implementation.

We simulate recursion using explicit stack and here the order of visiting nodes is not a thing to worry about as long as all nodes are visited.

```
stack <treenode*> S
S.push(root)
```

note: queue also can be used.

```
while (S is not empty)
{
    auto node = S.top()
    S.pop()

    if (node is null)
        don't do anything
    else
    {
        Swap(node->left, node->right)
        S.push(node->left)
        S.push(node->right)
    }
}
```

T(n) : O(n)

$S(n) : O(n)$