1. Problem

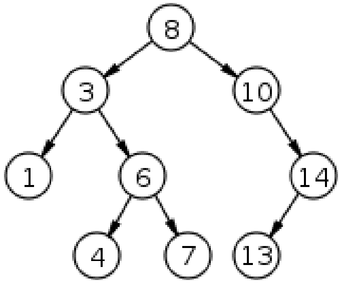
- Binary Tree: 1) isIdentical. 2) copy

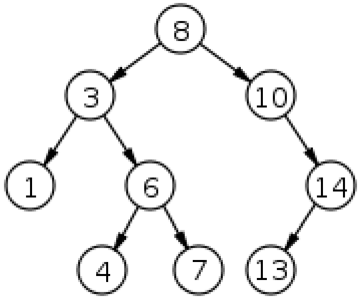
Input: 1) Roots of 2 BSTs. 2) Root of a BST.

Output: 1) A Boolean value

2) A deep-copied tree

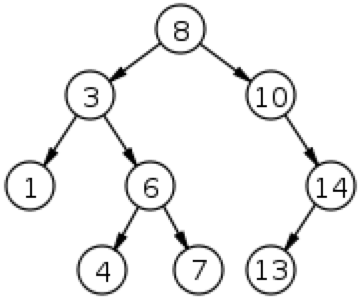
Example: 1)



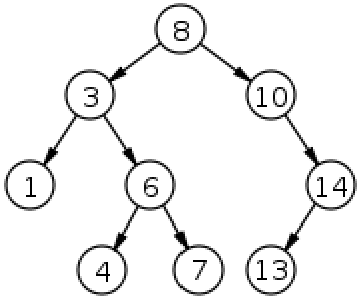


Output: True

2)



Output:



https://en.wikipedia.org/wiki/Binary_search_tree

2. Methods

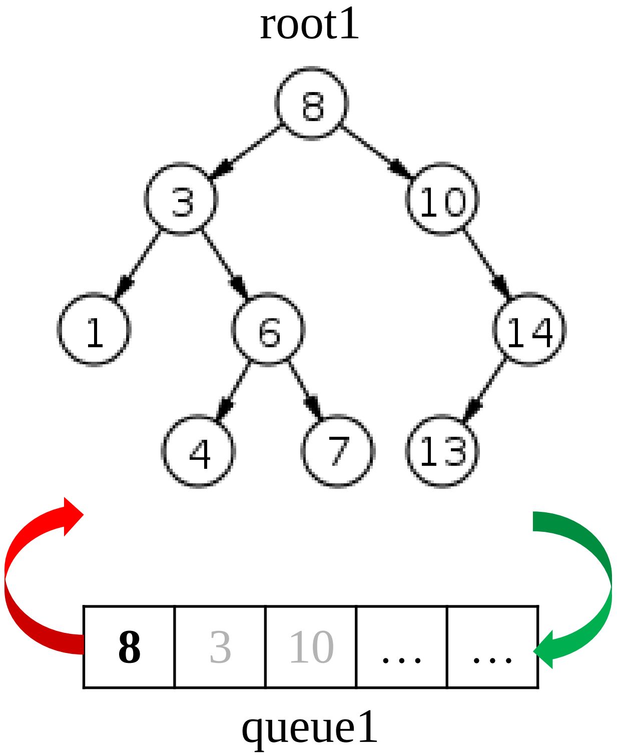
1) isIdentical. 2) copy

		Time complexity	Space complexity
isIdentical:	Iteratively	O(n)	O(n)
	Recursively	O(n)	O(n)
copy:	Iteratively	O(n)	O(n)
	Recursively	O(n)	O(n)

1) Iterative method

Pseudocode:

```
isIdentical (root1, root2)
Input: Roots of 2 BSTs
Output: A Boolean value
queue1.add(root1); queue2.add(root2)
while !queue1.isEmpty() and !queue2.isEmpty() do
    node1 ← queue1.poll(); node2 ←queue2.poll()
    if node1.val != node2.val do return False
    if node1.left != null and node2.left != null do
        queue1.add(node1.left); queue2.add(node2.left)
    else if (node1.left != null or node2.left != null) do
        return False
    if node1.right != null and node2.right != null do
        queue1.add(node1.right); queue2.add(node2.right)
    else if (node1.right != null or node2.right != null) do
        return False
return True
```



Space complexity: O(n)

Time complexity: O(n)

2) Recursive method

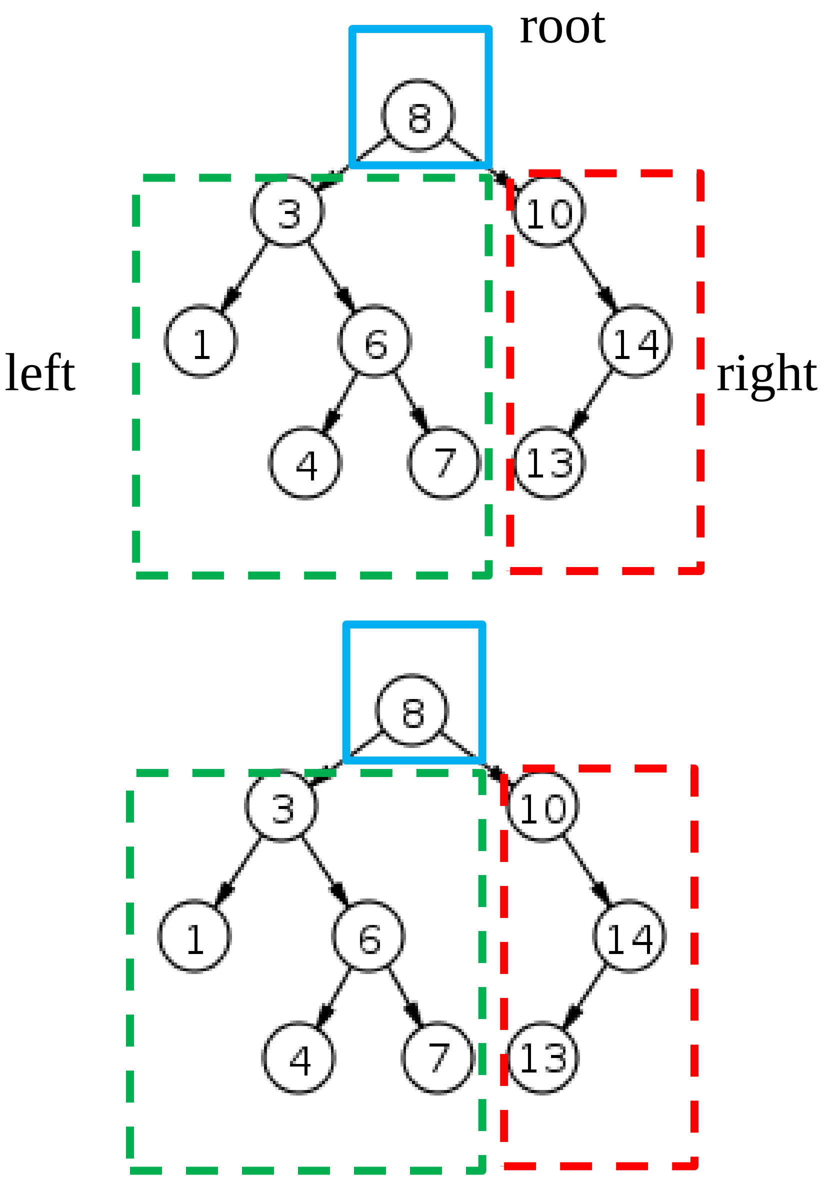
Pseudocode:

```
isIdentical(root1, root2)

Input: Roots of 2 BSTs
Output: A Boolean value
if root1 = null and root2 = null do
    return True
else if root1 != null and root2 != null do
    if (root1.val = root2.val and
         isIdentical(root1.left, root2.left) and
         isIdentical(root1.right, root2.right)): do
        return True
return False
```

Time complexity: O(n)

Space complexity: O(n)

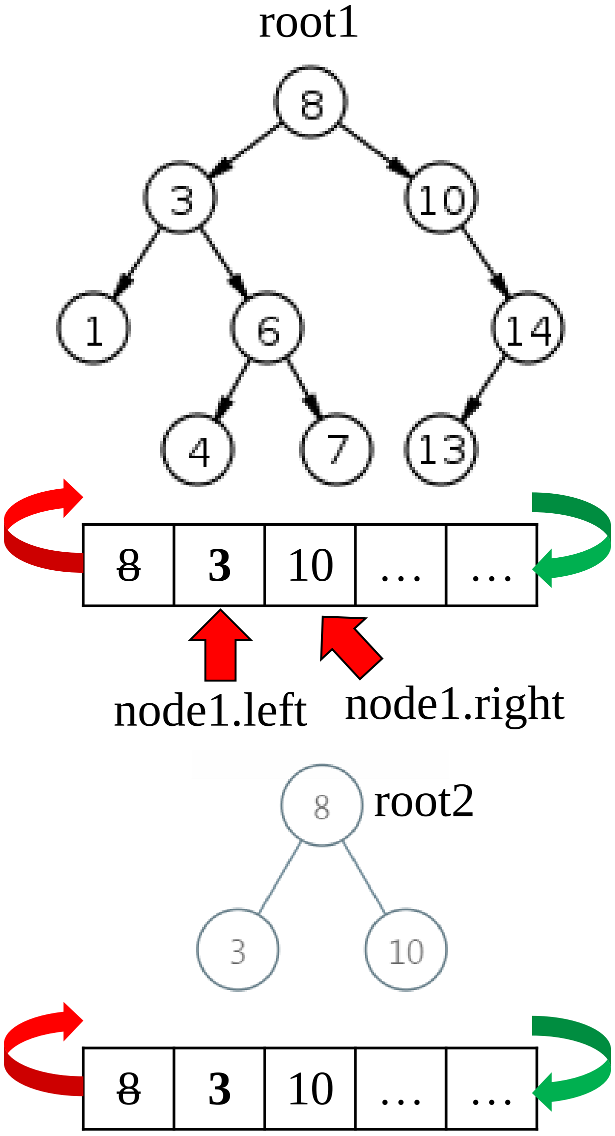


2. Algorithms copy

1) Iterative method

Pseudocode:

```
copy(root1)
Input: Root of a BST
Output: A deep-copied tree
root2  ← Node(root1.val)
queue1.add(root1); queue2.add(root2)
while !queue1.isEmpty() do
    node1 ← queue1.poll(); node2 ←queue2.poll()
    if node1.left != null do
        node2.left ← Node(node1.left.val)
        queue1.add(node1.left); queue2.add(node2.left)
    if  node1.right != null do
        node2.right ← Node(node1.right.val)
        queue1.add(node1.right); queue2.add(node2.right)
return root2
```

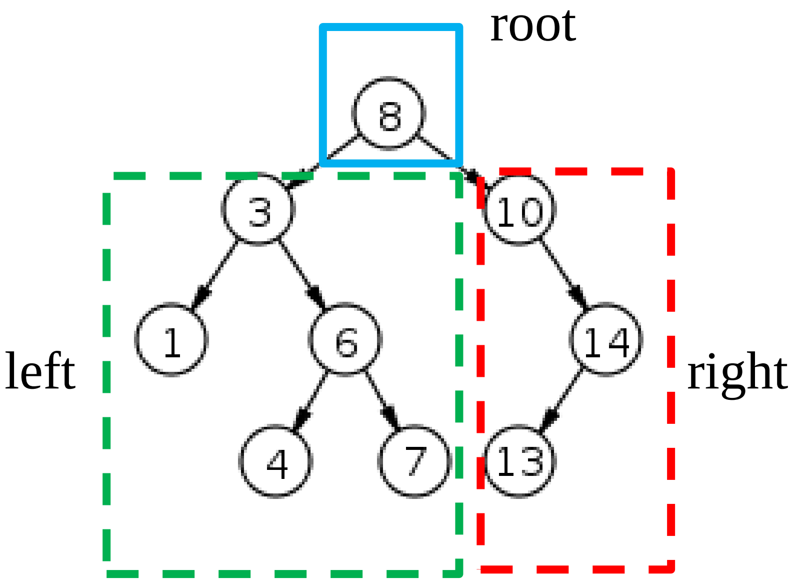


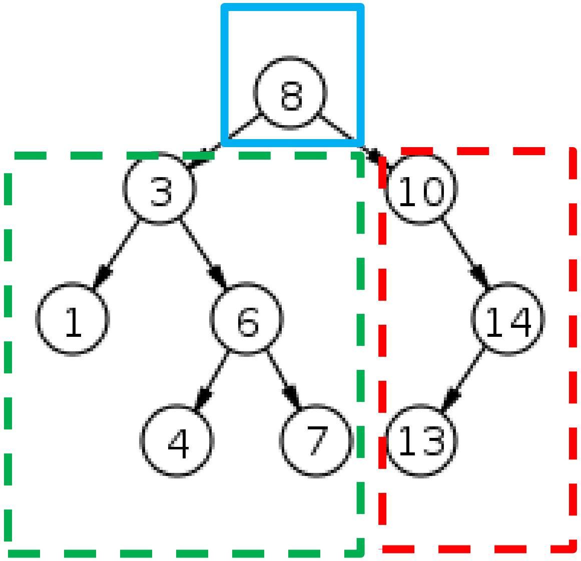
Time complexity: O(n)   Space complexity: O(n)

2) Recursive method

Pseudocode:

```
copy(root1)
Input: Root of a BST
Output: A deep-copied tree
if root1 = null do
    return null
newNode ← Node(root1.val)
newNode.left ← copy(root1.left)
newNode.right ← copy(root1.right)
return newnode
```





Time complexity: O(n)

Space complexity: O(n)

3. Summary

		Time complexity	Space complexity
isIdentical:	Iteratively	O(n)	O(n)
	Recursively	O(n)	O(n)
copy:	Iteratively	O(n)	O(n)
	Recursively	O(n)	O(n)