**Question 3 :**

**3 a> Algorithm to check if two binary search trees are same:**

**Steps:**

1. Check if the root node of the two given trees have same data
2. If they are then recursively call the same function on left child of the root node. Compare left child node data of the trees
3. Now do the same on the right child of the root node. Compare their data. This is done until we reach leaf nodes.
4. If all of them return true then the two trees are same else they are not same

**Time Complexity :**

This algorithm involves visiting each node in the tree to compare their data through recursive calls. In order to check if two trees are equal, we are traversing the tree in preorder. We can think of it as edge traversals, during preorder traversal each node is visited exactly twice.

Number of edges in the binary search tree with n nodes = (n -1)

maximum time taken = 2(n -1) = 2n- 2

* T(n) = 2n-2

By ignoring small terms, we have

**Time complexity T(n) = O(n)**

Since this involves pre-order traversal, the space complexity = h (height of the tree) as max size of the recursion stack can be h at any point of time

**Space Complexity = O(n)**

**3 b> Algorithm to create a copy of binary search tree:**

**Steps:**

1. Create an empty binary search tree to create a copy
2. Traverse the nodes of the given tree in pre-order. i.e first insert the root node to the new BST .
3. Then recursively call the same function on left node and then on right node.
4. This ensures the cloned copy created has the nodes in the same order and structure as the given BST.

**Time Complexity :**

This algorithm involves visiting each node in the tree to add the node to the new BST through recursive calls. we are traversing the tree in preorder. We can think of it as edge traversals, during preorder traversal each node is visited exactly twice.

Number of edges in the binary search tree with n nodes = (n -1)

maximum time taken = 2(n -1) = 2n- 2

* T(n) = 2n-2

By ignoring small terms, we have

**Time complexity T(n) = O(n)**

Since this involves pre-order traversal, the space complexity = h (height of the tree) as max size of the recursion stack can be h at any point of time