**PS5 - Problem 2. Prove that any n-node BST can transformed into any other n-node BST (with the same set of keys) using at most O(n) rotations.**

Suppose we want to transform tree A into tree B using only rotations. We can transform A and B into a "common" form in linear time, annotating the rotations made to obtain the "common" form from tree B and perform on the "common" form obtained from tree A the reverse of these rotations.
Let's consider a right vine (a degenerate tree where nodes can only have a right child) as a candidate "common" form. A recursive algorithm to construct a right vine from an arbitrary tree is the following:

```
construct_vine(root):
   node = root
   while node is not None:
      if node.left is not None:
         right_rotate(node)
      else:
         node = node.right
```

This algorithm ensures that, if the current node has a left child, it performs a right rotation such that the current node becomes a right child of its left child. In this way the node must be a right child (if there is a value which is smaller) and cannot have a left child, compyling with the vine definition.

Correctness: When we are able to arrive at the common form from tree A, reversing the rotations of B from end to start will necessairly bring us to tree B. That is, given tree B at i+1 and the node which right rotated from tree B at i, we can construct tree B at i by performing a left rotation on the same node.

Performance: Now consider what happens when we construct a vine from an arbitrary tree. A rotation is an operation which is constant in time since it involves only changing pointers. Every rotation decreases the nodes that have to be ordered on the vine by 1. Since there are n nodes and the smallest value of the bst does not need a right rotation, in the worst case (when the tree is a left vine), the algorithm takes n-1 operations.

Reconstructing tree B from the vine obtained from tree A involves simply starting from the last performed rotation in tree B and perform the inverse, a left rotation. Once you do that with every rotation you have performed to obtain a vine from tree B, you obtain tree B. You can convert any tree into another using at most 2n-2 operations -> O(n)