From here, we will prove the efficiency of a balanced binary search tree as it compares to an unbalanced binary search tree using by building trees using a list of integers.

---

5. **(15 points) Constructing Trees**

   (a) (5 points) (*You must submit code for this question!*) Use your **recursive implementations** of your AVL Tree and BST from Parts 1 and 2 to construct trees using `getRandomArray(10,000)`. Both trees must be made from the same array. In other words, **do not call the method twice - store the output of the method from `getRandomArray(10,000)` once and use it to construct both trees.**

   (b) (5 points) Did you run into any issues? Test your code on a smaller input (say, `getRandomArray(10)`, and see if you're still running into the same error. If it works on inputs of size 10 but not size 10,000, your code is probably fine and this is expected! Explain why you're running into issues (or might run into issues), using concepts we covered in class.

b) I did not run into any issues trying to insert 10,000 numbers/nodes. I tried to increase the number to 500,000 and I immediately noticed that the amount of time to just generate the array of numbers takes a very long time. I reduced the number of nodes to 100,000 and was able to generate the array within 1 minute. Inserting the nodes into the BST and AVL Trees were much faster. It took 0.2128 seconds.

c) Iterative was much faster than Recursion in BST. This time, 10000 nodes took 0.15 seconds to insert.

   (c) (5 points) (*You must submit code for this question!*) Use your **iterative implementations** of your AVL Tree and BST from Parts 1 and 2 to construct trees from the input of your implementation of `getRandomArray(10,000)`. Both trees must be made from the same array. In other words, **do not call the method twice - store the output of the method from `getRandomArray(10,000)` once and use it to construct both trees.**

6. **(15 points) Compare Implementations**

   (a) (5 points) (*You must submit code for this question!*) Modify your iterative implementations of your methods in `AVLTree` and `BinarySearchTree` by keeping track of how many times you traverse one level down in the tree. In other words, if you go from a node to its child, add 1 to the counter.

   (b) (5 points) (*You must submit code for this question!*) Construct a BST and AVLTree iteratively using `getRandomArray(10000)`. Compare how many levels we have to traverse in the two trees. You can include a screenshot of your code's output or write it out by hand.

b) I got the same number of level traversals = 151,283 for 10000 nodes inserted.

   (c) (5 points) (*You must submit code for this question!*) Construct a BST and AVLTree iteratively using `getSortedArray(10000)`. Compare how many levels we have to traverse in the two trees. You can include a screenshot of your code's output or write it out by hand.

This time the program traversed = 49995000 levels for both BST and AVL