Camilla Bocciolone
Alba Ballarà

ASSIGNMENT 2

PROBLEM 5

The complexity of the algorithm is O(N) where N is the number of nodes that compose the tree. We can arrive to this conclusion by observing first of all that the 4 if statements (if (n1 == null && n2 != null), if (n1 != null && n2 == null) , if (n1 == null) e if (n1.d != n2.d)) have a constant time complexity since they are simple comparisons. The last ifs contain instead 4 recursive calls. All the recursive calls get as an input two subtrees of the nodes considered. For example in our call of isomorphic at the beginning n1 and n2 will be the root and the first recursive call will get as an input the subtree to the left of the root n1 and the subtree to the left of the root n2. This means that the recursive call will work on smaller trees with a number of nodes that depends on the structure of the tree. But since the recursive calls are made by pair we can see that in reality the two if containing recursive calls work every time on both the subtrees of both the trees. if ((isomorphic(n1.left, n2.left)) && isomorphic(n1.right, n2.right)) on its first call will so work on all the nodes of the two trees except the roots so 2*(N-1) nodes and on its second call on 2*(N-2). This means the time complexity of the 5th if will be O(N) and since the same operation is done by the 6th if, its time complexity will be as well O(N). the most significant factor affecting the time complexity is therefore the number of nodes visited during the recursive calls and the time complexity of isomorphic is O(N)**.**

PROBLEM 6

Output:

Time that it takes to add 10100 values in an AVL tree randomly created: 5254090
Time that it takes to add 10100 values in a BST tree randomly created: 4180719
Time that it takes to find 1000 values in a BST tree randomly created: 658023
Time that it takes to find 1000 values in an AVL tree randomly created: 550369
Height AVL tree: 11
Height BST: 20
Time that it takes to find 5000 values in an AVL tree (tree created with sequantial numbers) : 418827
Time that it takes to find 5000 values in a BST tree(tree created with sequantial numbers) : 87388374
Height AVL tree: 12
Height BST: 4998

Comment:

AVL tree are binary search tree with a balance condition. To compare them with binary search tree we created an AVL tree and a binary search tree by adding and deleting nodes, like that the binary search tree will have a unbalanced structure while the AVL tree should maintain the balance conditions. In the binary search tree in fact the delete favors making the left subtree deeper than the right because we are always replacing a deleted node with a node from the right subtree. We checked the height of the tree and as expected the AVL tree has a height of around log2(1000)=10 while the binary search tree is taller due to the more unbalanced structure. To compare them in terms of operations we firstly took the time taken to add values. We observed that when adding 10100 values the average time taken by the add operation was bigger in the AVL tree, the add operation required in fact the call to a balance operation that rebalance the tree when we add a new value. We then tried to use a simple operation as contains in both types of tree for 1000 values and take the time to do it. Since the cost of the operation should depend on the height of the tree, we expected the contains to take longer in the binary search

tree because of the depth of the tree. In general in different runs we observed that the binary search tree was more likely to take more time because it took longer to find a value in a non balanced binary search tree but in some cases we got the opposite result. To get a more accurate performance comparison we created the trees with consecutive numbers in a way that the binary search tree was degenerate . We checked the height and they matched the expectation, the binary search tree had in fact a height of N-1 while the AVL maintained a balance structure and was smaller. We searched the values in the tree by using the operation contains and as expected the time taken to do the contains operation in the binary search tree was much bigger than in the AVL.In summary, the AVL tree maintains balance, and therefore, the contains operation is more consistent and efficient but the add operation require more time in the AVL because of the effort to maintain te balance condition.