COMP 210 – Data Structures and Analysis (Sec 1&2)

Assignment #6 – AVL Trees

Issue Date: November 8th, 2023
Due Date: Wednesday, November 15th, 2023, 11:55PM
Marks: 5

A. Tasks to be completed

This is the second to last assignment for the semester! Congratulations on making it this far, but make sure you devote time to these final two assignments since many students will regard these as being more difficult than our previous assignments.

Your task in this assignment is to implement an AVL tree as an implementation of the interface ***SelfBalancingBST***. The method signatures and explanations of what you need to implement are provided in the **SelfBalancingBST** Interface. Every method you need to implement is commented on in the ***AVLTree*** class, but you will also need to understand/use the methods given to you. The starter code containing these 3 files is attached and you should unzip them and place them in the appropriate *assn06* folder as done in past assignments.

We have specified the **fields,** and have implemented the following methods: **constructor, isEmpty, height, size, findMin, findMax getValue, getRight, getLeft**. The constructor takes no arguments and creates an empty AVLTree. You can either use a null element value to indicate that a tree is empty or explicitly keep track of whether a tree is empty with a boolean field (you would have to add this field).

You need to implement the following methods:

- **insert, remove, contains** (as required in the interface)
- **rotateLeft, rotateRight** (helper functions)

Inserting and removing elements to a self-balancing tree may result in a different root object after the insertion or removal because of self-balancing operations. This is why these methods as declared in the interface return the potentially *different post-operation root* of the tree. In your AVLTree implementation, this means you will need to recapture the result whenever you recursively do something to a child. [Note: For the **remove** method with 2 children, you must replace with minimum from the ***right*** *subtree* and not with the maximum from the left subtree.]

Self-balancing trees are one of the more difficult Data Structures to implement so we highly recommend making use of the debugger to help with any issues you may run into.

You are being provided with the **main method** as an example of how to test your AVL tree. As usual, *you should be writing additional incremental tests for your code* instead of only relying on the autograder. We are of course happy to help you with this in the office hours (via course.care) as well.

B. Submit to Grade Scope for Grading

Creating a Zip Archive in IntelliJ: Let's return to IntelliJ to prepare your zip archive for submission:

Mac Users
Along the bottom of your window, you should see an option to open a terminal integrated into IntelliJ. Type the following command (all on a single line, including the '.'):

zip -r assn06_submission.zip assn06 . (Your path should be set to the src file)

In the file explorer pane, look for the zip file named "assn06_submission.zip". If you right click on this file "Open in -> Finder" on Mac, the zip file's location on your computer will open. Upload this file to Gradescope to submit your work for this assignment.

Windows Users
Please navigate to your course workspace in the File Explorer window. Then right click on the **src** folder in your exercises directory and compress the **src directory** into a zip folder. You can name it "assn06_submission.zip" Please note, the level of the folder needs to include the src folder. Compression software on Windows may introduce additional folder structure, and the file naming must be the same.

Before uploading the zip file to Gradescope, *please delete any other files that showed up in the src/* folder of the zip file that were not actually part of assn06. Specifically, in this assignment, the files that need to be kept are those ending in .java. Then:

1. Log in to Gradescope.
2. Choose the assignment titled "**Assignment 6 – AVL Trees**".
3. You will find an option to upload a zip file.

Autograding will require a short while to process. For this exercise, there isn't a "human graded" component. Consequently, you should aim to achieve the full score of 100 points. If you encounter any reported issues, you should try to resolve these and then continue to resubmit until all issues are cleared.