UNIVERSITY OF MISSOURI-COLUMBIA
COLLEGE OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

CS 8050 – DESIGN AND ANALYSIS OF ALGORITHMS II

Fall 2025

ASSIGNMENT 3: TREES

Due Date: Friday November 7th , 2025, at 5:00 pm

(100 points)

1 Goals

In this assignment, you will develop a JavaFX application that allows users to interact with and visualize various tree data structures. The objective is to help you understand the implementation and behavior of different tree types while gaining hands-on experience in creating graphical user interfaces.

By the end of this assignment, you should be able to:

- Write Java code to implement a simple tree system with functionalities to insert, delete, search, store and retrieve different tree types.
- Understand how to implement the Tree ADT (Abstract Data Type) using the Node ADT.
- Use JavaFX and Scene Builder to create a graphical user interface (GUI) for the application.

2 Description and Requirements

2.1. Tree Implementations

Implement the following tree data structures:

- Binary Search Tree (BST) **(given)**
- AVL Tree
- Red-Black Tree
- Min Heap
- Max Heap
- 2-4 Tree

Each tree should support the following operations:

- Insert
- Delete
- Search
- Clear
- In-order traversal

2.2 User Interface

Create a JavaFX application with the following components:

- A dropdown menu to select the tree type
- Text input for inserting/deleting values
- Buttons for insert, delete, search, and clear operations
- A new button for In-Order-Traversal.
- A canvas area to display the current tree structure
- A text area to show operation results and tree traversals

2.3. Tree Visualization

- Implement a method to draw the selected tree on the canvas
- Nodes should be represented as circles with the value inside
- Use different colors for different tree types (e.g., red and black for Red-Black trees)
- Implement proper scaling to fit large trees on the canvas

2.4. File Operations

- Implement functionality to save the current tree to a file
- Implement functionality to load a tree from a file
- Use Java's serialization mechanism for saving and loading

3 Objects Implementations/Structure

Your project should include the following components:

- A main application class (e.g., TreeVisualizerApp)
- A controller class to handle user interactions and update the UI
- Separate classes for each tree implementation
- A common interface or abstract class for all tree types

To help you get started, a starter application kit is provided, which includes a full implementation of the Binary Search Tree (BST). If you haven't done so already, download the file Trees.zip and import it into your IntelliJ IDE. You are welcome to implement additional classes as needed.

4 A Sample Application Run

You will be provided by a session in the class that demonstrates a sample run of the expected requirements of your application, please attend the lectures, and make sure you fulfil the application requirements before you submit code.

Notice If you have any clarification questions or seeking guidelines in building your code, please feel free to ask the instructor and/or your TA during the designated office hours.

5 Hand In

- You need to use IntelliJ IDEA to create a Java Project and name it as "yourGroup#_cs8050_assignment#" (e.g., if yourGroup# is and you are submitting assignment 3 hence the IntelliJ project name will be "group7_cs8050_assignment3"). Then create a Java package in your IntelliJ Java project and name it the same name as the project name. Under this package you will create the Java classes and interfaces.
- Create a Java package in your IntelliJ project using the same name as your project. All your Java classes and interfaces must be placed under this package.
- All code and documentation should be contained within this package. Once complete, ZIP the entire IntelliJ project folder and submit it electronically through Canvas.

6 Grading

AVL Tree	Insert	5
	Delete	3
	Search	2
Total for AVL		10

RBT Tree	Insert	10
	Delete	8
	Search	2
Total for RBT Tree		24

Min Heap	Insert	5
	Delete	3
	Search	2
Total for Min Heap Tree		10

Max Heap	Insert	5
	Delete	3
	Search	2
Total for Max Heap Tree		10

2-4 Tree	Insert	15
	Delete	12
	Search	3
Total for Max Heap Tree		30
For all trees	In-order traversal	5
	Clear	3
	Store	5
	Load	7

Total for all above		100

Please note that:

- A 15-mark deduction will be applied for any runtime errors in your implementation.
- A 5-mark deduction will be applied for there are no comments in your code.