

CSCI 2110 Data Structures and Algorithms

Fall 2024

Laboratory No. 8

Week of November 18-22, 2024

Due (on Brightspace): Sunday, November 24, 11.59 PM

BINARY TREES

The objective of this lab is to help you to continue to get familiar with the Binary Search Tree Data Structure. Your task is to write, compile and run each program using an Integrated Development Environment (IDE). You may use any IDE of your choice – IntelliJ, Eclipse, NetBeans, VSCode, etc.

Marking Scheme: Throughout this course, we will be focusing on the complexity of algorithms and consequently, the efficiency of programs. In your first-year courses, the focus was on creating runnable code to specifications. This course builds on previous knowledge, but also focuses on efficiency. This will involve reducing unnecessary coding and designing or choosing good algorithms.

Each exercise will list the test cases that you need to test the code. Ensure that the output that you generate covers all the test cases.

Each exercise carries 10 points.

Your final score will be scaled down to a value out of 10. For example, if there are three exercises and you score 9/10, 10/10 and 8/10 on the three exercises, your total is 27/30 or 9/10.

Commenting your code: Your code should be clear and easy to understand. While extensive comments are not required, you are expected to include brief, meaningful comments to explain complex logic or any non-obvious sections of your code. Use inline comments to clarify key parts of the code. For methods and classes, use Javadoc to briefly describe their purpose, parameters, and return values. This ensures others can understand your code's intent quickly.

Error checking: Unless otherwise specified, you may assume that the user enters the correct data types and the correct number of input entries, that is, you need not check for errors on input.

Submission: One zip file containing the source codes (.java files) for all classes, and sample outputs. Please see details at the end of this document.

Submission Deadline: Sunday, November 24, 2024, 11.59 PM

Grace Time: Submissions will be accepted until 4.59 AM on Monday, November 25, 2024 without late penalty.

Late Penalty: Submissions received after the grace time will be subject to a 10% per day late penalty. The submission portal will close on Wednesday, November 27, 4.59 AM.

Dropping of Labs: Up to 2 labs can be dropped during the semester. No SDA submission is required.

Binary Search Trees

The objective of this lab is to help you get familiar with binary search trees. Download the example code/files provided along with this lab document. You will need the following files to complete your work:

BinaryTree.java (Generic BinaryTree Class)

BinarySearchTree.java (Generic BinarySearchTree Class)

The lab has one exercise and requires the completion of three methods, plus a driver program.

Exercise 1: Binary Search Tree (BST) Methods

You have been given the file BinarySearchTree.java. Complete the following methods (these are given as TODO methods in the BinarySearchTree.java code):

1. Complete the *findMax* method in the BinarySearchTree.java file. This method should return the maximum data value stored in the binary search tree (i.e., the largest integer or double, the String which is lexicographically (i.e., alphabetically) last).
2. Complete the *findMin* method in the BinarySearchTree.java file. This method should return the minimum data value stored in the binary search tree (i.e., the smallest integer or double, the String which is lexicographically first).
3. Complete the *recursiveSearch* method in the BinarySearchTree.java file. This method returns a BinaryTree whose data value matches the search key. Your solution must be recursive.
 - Before you begin coding, do a trace with pen and paper to understand how to recurse through the tree.
 - There are two recursive search methods in the starter code.
 - One to be completed
 - One which acts as a helper method by calling the second

Once your search method has been implemented and tested, write a program called **Exercise1.java** with a main method that does the following:

- Create a binary search tree which stores positive integers (>0) from user input. You will read in integer values from the user until the sentinel value 0 is entered.
- Construct the BST. The *insert* method is useful here.

Once your BST has been constructed, perform the following operations:

Print the max element

Print the min element

Prompt the user to search for an element in the tree:

- Search for an element that exists and print the result
- Search for an element in the tree that doesn't exist and print the result

Example input/output

```
Enter int or '0': 1
Enter int or '0': 2
Enter int or '0': 3
Enter int or '0': 4
Enter int or '0': 5
```

```
Enter int or '0': 6
Enter int or '0': 7
Enter int or '0': 8
Enter int or '0': 9
Enter int or '0': 0
The max data value in the BST is: 9
The min data value in the BST is: 1
What key would you like to search for? 3
Found!
```

Submit the input/output for at least three runs of the program. Do not use the same values as in the above example. There should be at least one positive test (where the search key is found), and one negative test (where the search key is not found).

What to submit:

Submit one ZIP file containing all source code (files with .java suffixes) and the text/word/similar document containing sample output. For the given exercise you will minimally have to submit a class containing your completed methods, a demo class, and sample program runs. You may combine your sample program runs into a single document via cut-and-paste or a similar method.

Your final submission should include the following files: **BinaryTree.java**, **BinarySearchTree.java**, **Exercise1.java**, document containing sample program runs.

Note 1: Although you may not have made any changes to BinaryTree.java, please include it in your submission package so that the TAs can execute your code easily.

Note 2: Please ensure that your full name and Banner ID appears on the document containing sample program runs.