CSDS 233 Assignment #3

Due October 27, 2024, before 11:59 pm. 100 points

Submission Instructions

- The submissions will be evaluated on completeness, correctness, and clarity.
- Please provide sufficient comments in your source code to help the TAs read it.
- Please generate a single zip file containing all your *.java files needed for this assignment (not .class) and optionally a README.txt file with an explanation about added classes and extra changes you may have done.
- Name your file P1_YourCaseID_YourLastName.zip for your **coding exercises**. Submit your zip file electronically to Canvas.
- Please submit a **PDF** for your answers to the written exercise **separate from the zip file**.

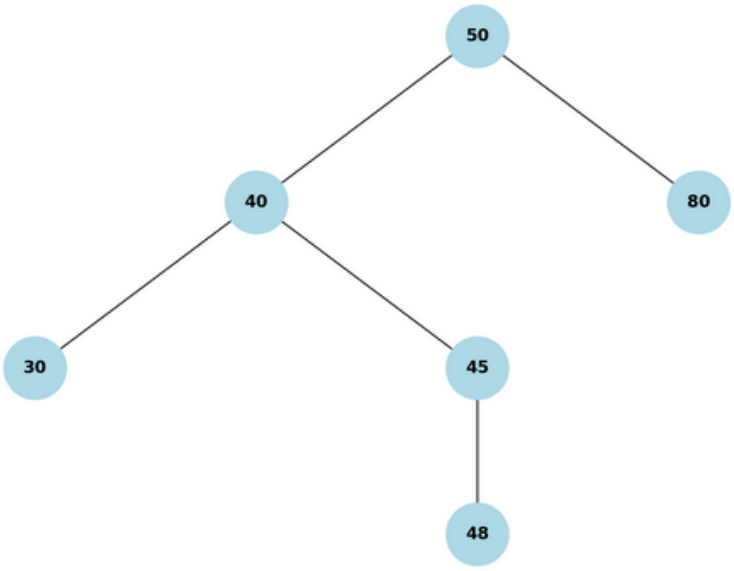
Office Hours for This Assignment
Written Assignment: October 23, 12-1pm Olin 404
Programming Assignment: October 18t, 5-6pm Olin 404

Contact Information
If you have questions outside of the scheduled office hours:

- For written questions, email: Quoc Trung Nguyen
- For coding problems, email: Vedant Gupta

Written Assignment (50 POINTS)

Question 1: AVL TREE OPERATION (15 points)



a) Perform rotations on the tree above to create balanced AVL tree (4 points)
b) Insert **49** into the tree above. What is the resulting tree? (3 points)
c) Delete **30** from the resulting tree from part (b). What is the resulting tree? (3 points)
d) Insert **40, 20, 10, 25, 50, 60, 70, 55, 30, 45** to an empty AVL Tree in the given order. Draw the final tree after all insertions (5 points)

Question 2: B-TREE (10 POINTS)

For each statement, indicate whether it is true or false. If false, explain why.

1. In a B-tree of order m, internal nodes must always have exactly m/2 children, regardless of the number of keys in the tree.
2. During the insertion process in a B-tree, if a node becomes full, it is always the root node that splits first.
3. In a B-tree, the height of the tree grows logarithmically as the number of keys increases.
4. The time complexity of searching for a key in a B-tree depends on both the height of the tree and the number of keys stored at each node.
5. In a B-tree, after a deletion operation, the tree always requires a merge operation between adjacent nodes to maintain its properties.

Question 3: TREE IN GENERAL (10 POINTS)

Answer these questions and make sure to explain your answer.

1. What is the primary difference between a binary tree and a binary search tree (BST)? (2 points)
2. What is the minimum number of nodes required to build an AVL tree with height 4? Provide your calculations and reasoning. (2 points)
3. In your own words, explain the concept of the balance factor in an AVL tree and what causes an imbalance. (2 points)
4. How does a balanced binary tree differ from an AVL tree? (2 points)
5. Provide an example of a binary tree that is balanced but not an AVL tree, and explain why it doesn't meet AVL tree requirements. (2 points)

Question 4: HEAP (15 POINTS)

1. For each statement, indicate whether it is true or false. If false, explain why.
    a. In a min-heap, every parent node must have a value less than or equal to the values of its children. (2 points)

b. The worst-case time complexity for inserting an element into a heap is O(logn). (2 points)
c. The height of a heap with n nodes is always O(logn), regardless of whether it's a min-heap or a max-heap. (2 points)
d. Performing the "heapify" operation on a heap is an O(n) process (2 points)
e. A min-heap can be used to find the second smallest element in O(1) time (2 points)

2. Given the sequence [14, 3, 21, 9, 8, 5].
    a. Insert the above sequence one by one into a min heap. Show the heap structure after each insertion in array form (3 points)
    b. Delete the root element from the heap. Perform the "heapify" operation and show the resulting heap structure. (2 points)

`

Programming Assignment: Implementing a Binary Search Tree in Java [50 Points]

In this assignment, you will implement a Binary Search Tree (BST) in Java. You are not allowed to use any built-in libraries for data structures. The purpose of this assignment is to manually implement the binary search tree and understand the fundamentals of tree operations and traversal techniques. You will implement a class named BinarySearchTree with all of the following methods. In addition you will create another java class (instructions below). Make sure to structure your code clearly and include comments for clarity.

1. Class and Node Structure [5 Points]

You will implement two main components:

1. Private Node Class:

- A private inner class Node that represents a node in the binary search tree. It should include:

- int key: Stores the key value for the node.

- Node left: Points to the left child node.

- Node right: Points to the right child node.

- The Node class should also have an appropriate constructor to initialize the key and children (set them to null).

2. BinarySearchTree Class:

- A class named BinarySearchTree that contains:

- Node root: The root of the binary search tree.

- All required methods

2. Tree Operations [40 Points]

The following methods must be implemented in your BinarySearchTree class. The method stubs have been provided to you.

1. Insert Operation [5 Points]

- void insert(int key): Inserts a new node with a given key into the binary search tree. The method should correctly place the node according to the binary search tree rules.

2. CreateTree Operation [5 Points]

- void createTree(int[] values): Creates a binary search tree from an array of integers by inserting each value into the tree using the insert() method.

3. Search Operation [5 Points]

- boolean search(int key): Searches for a node with a given key in the binary search tree. Returns true if the key is found, false otherwise.

4. Delete Operation [10 Points]

- Node delete(int key): Deletes a node with the specified key from the binary search tree and returns it. This method must correctly handle all possible cases:

    - The node to be deleted is a leaf (has no children).

    - The node to be deleted has one child.

    - The node to be deleted has two children (Hint: replace it with its inorder successor).

5. Min and Max Value Operations [5 Points]

- int findMin(): Finds and returns the minimum key value in the binary search tree.

- int findMax(): Finds and returns the maximum key value in the binary search tree.

6. Tree Height Operation [4 Points]

- int height(): Returns the height of the binary search tree.

7. Traversals [6 Points]

- In-order Traversal:

    - void inorderTrav(): Traverses the tree in in-order and prints the elements.

- Pre-order Traversal:

    - void preorderTrav(): Traverses the tree in pre-order and prints the elements.

- Post-order Traversal:

    - void postorderTrav(): Traverses the tree in post-order and prints the elements.

3. Test Cases [10 Points]

In a new .java file, create a main class named BinarySearchTreeMain to test the functionality of your binary search tree. The test class should:

1. Insert Nodes and Display Traversals [1 Points]

- Insert a set of elements into the tree.

- Display the in-order, pre-order, and post-order traversals after inserting nodes.

2. Test CreateTree Method [2 Points]

- Create a binary search tree from an array of integers and display the in-order traversal.

3. Test Search Method [2 Points]

- Search for a few key values (three present and three absent) in the tree and print the result.

4. Test Delete Method [1 Point]

- Delete three nodes from the tree and display the in-order traversal after each deletion.

5. Test Find Min and Max Operations [2 Points]

- Test the findMin() and findMax() methods and display the minimum and maximum values found in the binary search tree.

6. Test Tree Height Method [2 Points]

- Test the height() method and display the height of the binary search tree.

Submission Instructions

Submit two Java files:

1. BinarySearchTree.java: The file containing your BinarySearchTree class with all the methods required.

2. BinarySearchTreeMain.java: The file containing your main class for testing the binary search tree.

Ensure that you have thoroughly tested all of the functions as per the requirements. Please include detailed comments in your code explaining the purpose and logic of each method. At the top Please have an author tag and add your name. Cite any sources used in comments at the top.