Mandar Palkar **2140-6740** **mandar.palkar@ufl.edu**

COP 5536 Fall 2022

Programming Project
AVL Tree

Instructions to Run:
Navigate to the project directory and run the following set of commands.

```
- make
- java avltree <input_file>
```

The program will create a tree and perform the operations listed (Initialize/Search/Insert/Delete) in the input file. The output for the search file will be written to the filename called **output_file.txt.**

Program Structure:
The whole program is divided into 7 classes.

1. **TreeNode:**

This class represents the structure of the individual node of the AVL tree.
It encapsulates the following attributes and their getters setters:

1) **data:** Key associated with the node
2) **height:** Height of the subtree.
3) **left:** Pointer to the left child
4) **right:** Pointer to the right child.

2. **OpCode**

This is the Enum class that represents the operations to be done on the tree i.e programmatically in the form of enumeration. It provides enumerations for the following actions to be done on a tree.

1) ***Initialize***
2) ***Insert***
3) ***Delete***
4) ***Search***

3. **Operation:**

This class is the programmatic representation of the individual record that is being read from the input file provided. This class consists of the following attributes and their getters setters:

1) ***opcode*****:** Enum for the operations provided and is of type code.

2) ***val1*****:** First Integer value associated with the operation.
3) ***val2*****:** Second Integer value associated with the operation. Only Applicable for Search Range Operation

4. InputReader

This class contains functions that help with reading operations specified in the supplied input file. It contains two functions:

1. private static Operation readRecordFromLine(String line):

This is a helper function that takes a string as a parameter and parses it with the regular expression **^([A-Za-z]+)\(([\\-0-9,]*)\)$** to obtain pattern groups: the first group contains the operation name (Initialize/Insert/Delete/Search) and the second group contains the data associated with the operation.

For e.g: The text "Insert(28)" will be divided into two groups first being "Insert" and the second being "28". These groups will be then used to construct an Operations Object and that object will be returned from this function.

2. public List<Operation> getOperationsFromFile(String filename):

This function reads lines from the filename passed as the function parameter.

And then calls the function "**readRecordFromLine(String line)"** on every line to construct an Object of class Operation. This object is added to a list. And the list of such objects is returned from the function.

5. ResultWriter

This class contains the functionality to read the list operations supplied in the input file.

It contains the following functions:

1. public void initializeFile():

Creates a FileWriter Object that in turn creates a new file named **output_file.txt**

2. public void writeToFile(String txt):

Takes a string as a parameter and writes that string as a newline to the file created earlier.

3. public void close(String txt):

Closes the file object created by the initializeFile() function.

4. **AVLTreeUtil:**

This class maintains and takes care of the operations on the Actual Tree. It contains the following functions:

1. **public void insert(int data):**

An exposed method that calls the internal recursive function for insert.

2. **private TreeNode insert(TreeNode treeNode, int data):**

A private recursive method that performs the insert for a key recursively in an AVL tree.

3. **public void delete(int data):**

An exposed method that calls the internal recursive function for delete.

4. **private TreeNode delete(TreeNode treeNode, int data):**

A private method function that performs the deletion of a key recursively in an AVL tree.

5. **private int getHeight(TreeNode treeNode):**

A private method that returns the node's height for non-null nodes. Returns 0 for null nodes.

6. **private int getBalanceFactor(TreeNode treeNode):**

A private method that calculates the difference of height between left subtree and right subtree.

7. **private TreeNode findReplaceableNode(TreeNode treeNode):**

An internal method that finds the predecessor node for the node that is marked for deletion.

8. **private TreeNode rotateLeft(TreeNode treeNode):**

An internal method that performs left rotation on the Node.

9. **private rotateRight(TreeNode treeNode):**

An internal method that performs right rotation on the Node.

10. **public String search(int val):**

An exposed method that calls the internal recursive search for a single value.

11. private String search(TreeNode treeNode, int val):
An exposed method function that calls the search for a single value.

12. public String search(int val1, int val2):
An internal method function that recursively search values that are in range val1 <= values <= val2.

13. public String search(TreeNode treeNode, int val1, int val2, List<String> result):
internal method function that recursively search values that are in range val1 <= values <= val2.

5. **avltree:**
This is the main class that acts as a driver/wrapper for calling functions of classes mentioned above. It instantiates objects of other classes to drive the operations.

1. **public void initializeTree():**
Initalizes the new instance of AVLTreeUtil Class.

2. **public void writeResult(String result):**
Wrapper call for the ResultWriter's writeToFile(String text) method of AVLTreeUtil class .

3. **public void closeWriter():**
Wrapper function close() method of AVLTreeUtil class.

4. **public void insert(int val):**
Wrapper function for insert(int val) method of AVLTreeUtil class instance.

5. **public void delete(int val):**
Wrapper function for delete(int val) method of AVLTreeUtil class instance.

6. **public void search(int val):**
Wrapper function for search(int val) method of AVLTreeUtil class instance.

7. **public void search(int val1, int val2):**
Wrapper function for search(int val1,int val2) method of AVLTreeUtil class instance.

8. public void readOperations(String filename):

Wrapper function for readOperations(String filename) method of InputReader class instance. The InputReader's readOperations call returns a List of Objects of class Operations, which is then assigned to local attribute named operations.

9. public void performOperations():

Iterates over objects from list and perform operations on the basis of the opcode of the operations and the data associated with each operation.

Program Structure: