CSE 222 HOMEWORK 7: Balanced Tree-based Stock Data Management

Gebze Technical University Computer Engineering Department

Submission Deadline: 23th May 2024 23:59

ÇAĞRI YILDIZ – 1901042630

Part 1: Detailed Explanation of the Program

Overview

This project involves implementing an AVL tree to manage stock data efficiently. The AVL tree will ensure balanced tree operations to maintain optimal performance for storing, retrieving, and manipulating stock information.

Class Hierarchy and Core Features

1. **Stock Class**
   - Represents a stock with attributes: symbol, price, volume, and marketCap.
   - Code Snippet:

```java
public class Stock {
    private String symbol;
    private double price;
    private long volume;
    private long marketCap;

    /**
     * Constructs a new Stock object.
     *
     * @param symbol the stock symbol
     * @param price the stock price
     * @param volume the trading volume
     * @param marketCap the market capitalization
     */
```

2. **AVLTree Class**
   - Implements an AVL tree to store and manage Stock objects.
   - Key operations: insertion, deletion, search, and tree rotations to maintain balance.
   - Code Snippet:

```java
public class AVLTree {
    private class Node {
        Stock stock;
        Node left, right;
        int height;

        Node(Stock stock) {
            this.stock = stock;
            height = 1;
        }
    }

    private Node root;
```

3. **StockDataManager Class**
   - Manages the AVL tree and provides methods for adding, removing, searching, and updating stocks.
   - Code Snippet:

```java
public class StockDataManager {
    private AVLTree avlTree;

    /**
     * Constructs a new StockDataManager.
     */
    public StockDataManager() {
        this.avlTree = new AVLTree();
    }
```

4. **GUIVisualization Class**
   - Provides a graphical interface to visualize the performance graphs.
   - Code Snippet:

```java
/**
 * A simple graphical user interface (GUI) to visualize the performance graphs.
 */
public class GUIVisualization extends JFrame {
    private List<Integer> dataPointsX;
    private List<Long> dataPointsY;
    private String title;

    /**
     * Constructs a new GUIVisualization.
     *
     * @param title the title of the graph
     * @param dataPointsX the x-axis data points
     * @param dataPointsY the y-axis data points
     */
```

5. **Main Class**
   - Handles generating random input files, processing commands, and performing performance analysis.
   - Code Snippet:

```java
public class Main {

    Run | Debug
    public static void main(String[] args) {
        StockDataManager manager = new StockDataManager();

        // Generate random input file
        String inputFilename = "random_input.txt";
        generateRandomInputFile(inputFilename, addCount:5, removeCount:3,

        // Process the generated input file
        processInputFile(manager, inputFilename);

        // Perform and print tree traversals
        System.out.println(x:"In-Order Traversal:");
        manager.inOrderTraversal();

        System.out.println(x:"Pre-Order Traversal:");
        manager.preOrderTraversal();

        System.out.println(x:"Post-Order Traversal:");
        manager.postOrderTraversal();

        // Perform performance analysis
        performanceAnalysis(manager);
    }
```

Design Decisions

- **Balancing Tree:** Used AVL tree to ensure operations maintain O(log n) time complexity.
- **Update on Duplicate**: If a stock with the same symbol exists, its attributes are updated instead of adding a new node.
- **Graphical Visualization:** Implemented GUI using Java Swing for performance visualization.

Challenges and Solutions

- **Tree Balancing:** Implemented complex rotations (left, right, left-right, right-left) to maintain AVL properties.
- **Performance Measurement:** Accurately measured and visualized performance of different operations (ADD, REMOVE, SEARCH, UPDATE).

Part 2: Compilation and Execution

1. **Compilation**
   - Use the provided Makefile to compile the program.
   - Command: *make*

2. **Running the Program**
   - After compilation, you can run the program using the run target in the Makefile.
   - Command: *make run*

3. **Generating Javadoc**
   - Use the javadoc target in the Makefile to generate the documentation.
   - Command: *make javadoc*

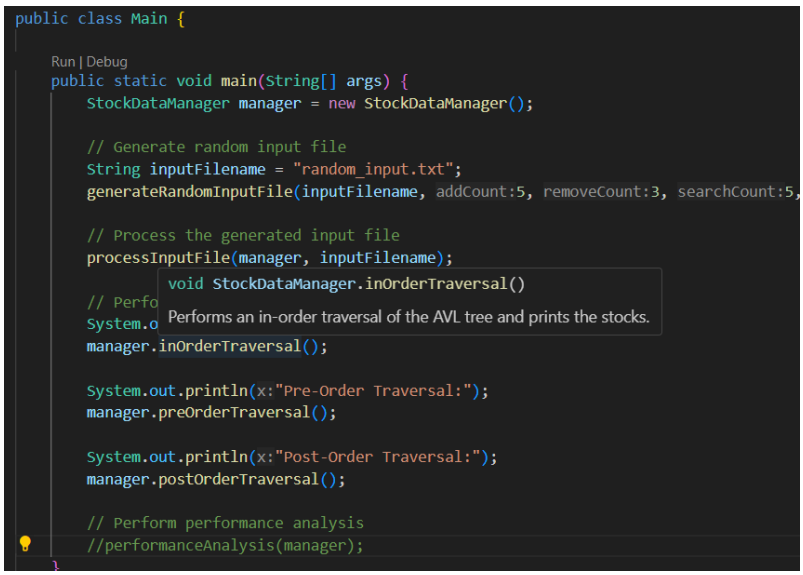
Part 3: Program Flow

1. **Generating Random Input File**
   - The generateRandomInputFile method generates a file with random commands for adding, removing, searching, and updating stocks.

2. **Processing Input File**
   - The processInputFile method reads the generated input file and executes the commands on the StockDataManager.
   -
3. **Performance Analysis**
   - The performanceAnalysis method measures the time taken for different operations (ADD, REMOVE, SEARCH, UPDATE) on varying tree sizes and visualizes the results.

Part 4: Terminal Outputs

1. **Processing input file:**
   - To see the results better in the terminal, we must comment the ***performanceAnalysis*** line. If we don't, it will be very difficult to understand anything from thousands of terminal lines.
   - I adjusted the code to create different inputs each time. That's why we don't need to enter any .txt file from the command line.



- After compile the code

```
PS C:\Users\cagri\OneDrive\Masaüstü\src> javac Main.java
PS C:\Users\cagri\OneDrive\Masaüstü\src> java -Xint  Main
Stock not found: FBRVB
Removed stock: AOEP
Added stock: VKRHE
Removed stock: AOEP
Added stock: FBRVB
Added stock: IWPBB
Added stock: AOEP
Found stock: Stock{symbol='FBRVB', price=366.79, volume=10379, marketCap=1022928}
Found stock: Stock{symbol='AOEP', price=837.56, volume=9959, marketCap=1025333}
Updated stock: FBRVB to TZVVA
Added stock: FVCPJ
Found stock: Stock{symbol='IWPBB', price=480.07, volume=4457, marketCap=1029408}
Found stock: Stock{symbol='IWPBB', price=480.07, volume=4457, marketCap=1029408}
Removed stock: FBRVB
In-Order Traversal:
Stock{symbol='AOEP', price=837.56, volume=9959, marketCap=1025333}
Stock{symbol='FVCPJ', price=300.89, volume=9647, marketCap=1060344}
Stock{symbol='TZVVA', price=113.07, volume=8411, marketCap=1055214}
Stock{symbol='IWPBB', price=480.07, volume=4457, marketCap=1029408}
Stock{symbol='TZVVA', price=113.07, volume=8411, marketCap=1055214}
Stock{symbol='VKRHE', price=213.04, volume=1583, marketCap=1019429}
Pre-Order Traversal:
Stock{symbol='IWPBB', price=480.07, volume=4457, marketCap=1029408}
Stock{symbol='FVCPJ', price=300.89, volume=9647, marketCap=1060344}
Stock{symbol='AOEP', price=837.56, volume=9959, marketCap=1025333}
Stock{symbol='TZVVA', price=113.07, volume=8411, marketCap=1055214}
Stock{symbol='VKRHE', price=213.04, volume=1583, marketCap=1019429}
Stock{symbol='TZVVA', price=113.07, volume=8411, marketCap=1055214}
Post-Order Traversal:
Stock{symbol='AOEP', price=837.56, volume=9959, marketCap=1025333}
Stock{symbol='TZVVA', price=113.07, volume=8411, marketCap=1055214}
Stock{symbol='FVCPJ', price=300.89, volume=9647, marketCap=1060344}
Stock{symbol='TZVVA', price=113.07, volume=8411, marketCap=1055214}
Stock{symbol='VKRHE', price=213.04, volume=1583, marketCap=1019429}
Stock{symbol='IWPBB', price=480.07, volume=4457, marketCap=1029408}
PS C:\Users\cagri\OneDrive\Masaüstü\src>
```

- Created txt file

random_input.txt

```
SEARCH FBRVB
REMOVE AOEP
ADD VKRHE 213.04 1583 1019429
REMOVE AOEP
ADD FBRVB 366.79 10379 1022928
ADD IWPBB 480.07 4457 1029408
ADD AOEP 837.56 9959 1025333
SEARCH FBRVB
SEARCH AOEP
UPDATE FBRVB TZVVA 113.07 8411 1055214
ADD FVCPJ 300.89 9647 1060344
SEARCH IWPBB
SEARCH IWPBB
REMOVE FBRVB

```

- As you see, you can check the trevelsals in the terminal output , too.

```
In-Order Traversal:
Stock{symbol='AOEP', price=837.56, volume=9959, marketCap=1025333}
Stock{symbol='FVCPJ', price=300.89, volume=9647, marketCap=1060344}
Stock{symbol='TZVVA', price=113.07, volume=8411, marketCap=1055214}
Stock{symbol='IWPBB', price=480.07, volume=4457, marketCap=1029408}
Stock{symbol='TZVVA', price=113.07, volume=8411, marketCap=1055214}
Stock{symbol='VKRHE', price=213.04, volume=1583, marketCap=1019429}
Pre-Order Traversal:
Stock{symbol='IWPBB', price=480.07, volume=4457, marketCap=1029408}
Stock{symbol='FVCPJ', price=300.89, volume=9647, marketCap=1060344}
Stock{symbol='AOEP', price=837.56, volume=9959, marketCap=1025333}
Stock{symbol='TZVVA', price=113.07, volume=8411, marketCap=1055214}
Stock{symbol='VKRHE', price=213.04, volume=1583, marketCap=1019429}
Stock{symbol='TZVVA', price=113.07, volume=8411, marketCap=1055214}
Post-Order Traversal:
Stock{symbol='AOEP', price=837.56, volume=9959, marketCap=1025333}
Stock{symbol='TZVVA', price=113.07, volume=8411, marketCap=1055214}
Stock{symbol='FVCPJ', price=300.89, volume=9647, marketCap=1060344}
Stock{symbol='TZVVA', price=113.07, volume=8411, marketCap=1055214}
Stock{symbol='VKRHE', price=213.04, volume=1583, marketCap=1019429}
Stock{symbol='IWPBB', price=480.07, volume=4457, marketCap=1029408}
PS C:\Users\cagri\OneDrive\Masaüstü\src>
```

- If an input that has not been added yet is deleted, there is no problem in deleting it since it does not exist.

```
private Node deleteNode(Node root, String symbol) {
    if (root == null)
        return root;
```

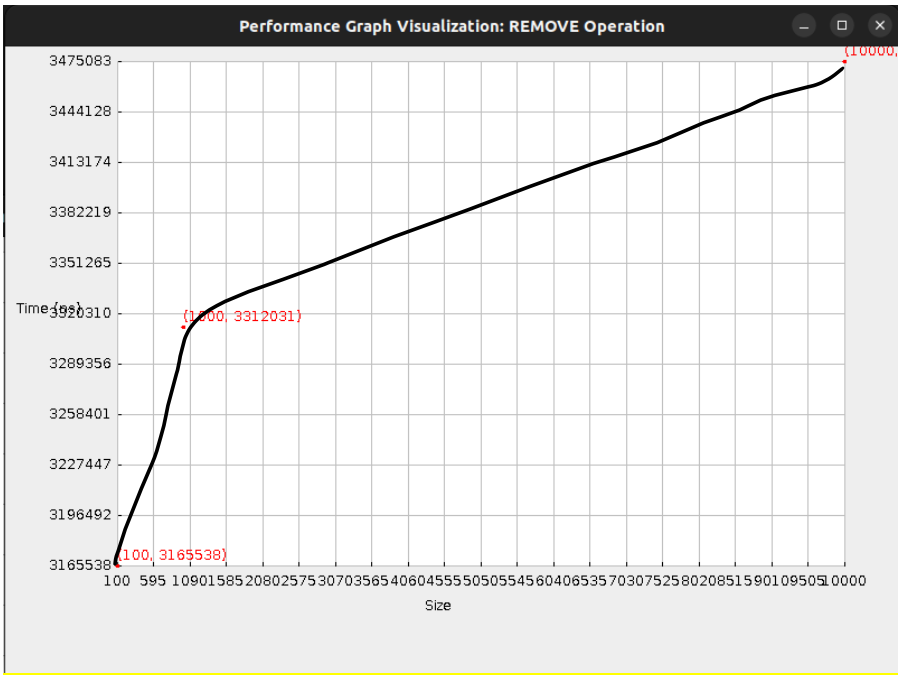
2. **Performing Analysis:**
   - In the performance analysis part of the program, we aim to measure and visualize the time complexity of various operations (ADD, REMOVE, SEARCH, UPDATE) on the AVL tree with different tree sizes. Here's how we conducted the performance analysis:
     (a) Define Tree Sizes for Testing
       - We tested with different tree sizes: 100, 1000, and 10,000 nodes to observe how the AVL tree operations scale.
     (b) Generate Input Files
       - For each tree size, we generated random input files containing a mix of ADD, REMOVE, SEARCH, and UPDATE commands.
     (c) Measure Execution Timeü
       - For each operation type, we measured the time taken to process the commands in the generated input files.
     (d) Store Execution Times
       - We stored the average execution times for each operation type to visualize later.
     (e) Visualize Performance Data
       - Using the GUIVisualization class, we created graphs to visualize the relationship between tree size and average execution time for each operation.

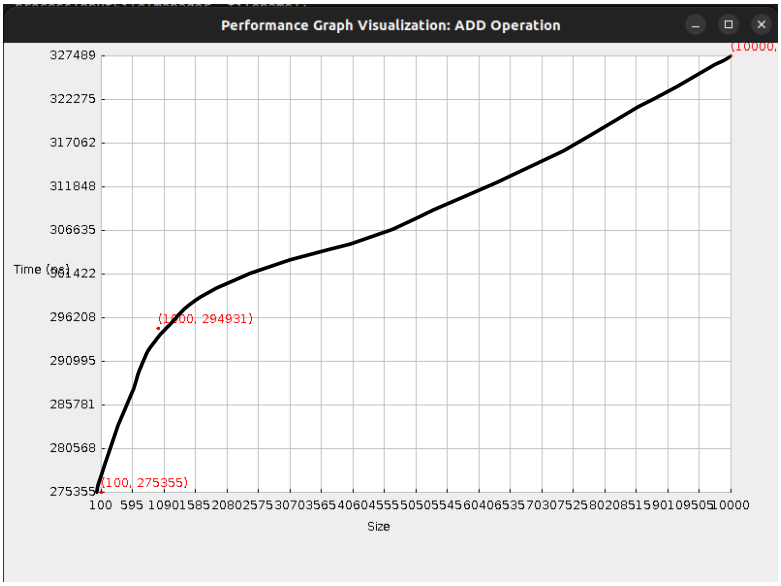
   - To see the results better in the terminal, we must uncomment the ***performanceAnalysis*** line.

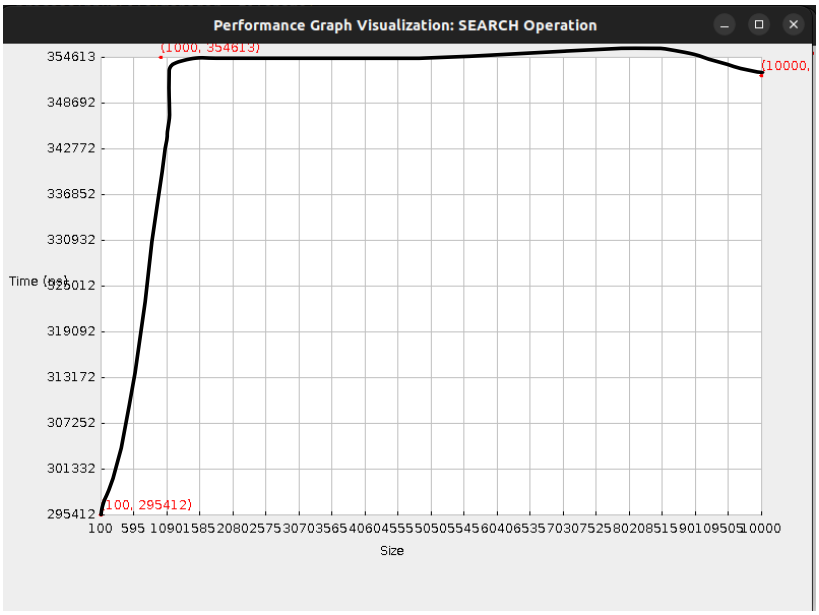
```
public class Main {

    Run | Debug
    public static void main(String[] args) {
        StockDataManager manager = new StockDataManager();

        // Generate random input file
        String inputFilename = "random_input.txt";
        generateRandomInputFile(inputFilename, addCount:5, removeCount:3,

        // Process the generated input file
        processInputFile(manager, inputFilename);

        // Perform and print tree traversals
        System.out.println(x:"In-Order Traversal:");
        manager.inOrderTraversal();

        System.out.println(x:"Pre-Order Traversal:");
        manager.preOrderTraversal();

        System.out.println(x:"Post-Order Traversal:");
        manager.postOrderTraversal();

        // Perform performance analysis
        performanceAnalysis(manager);
    }
```

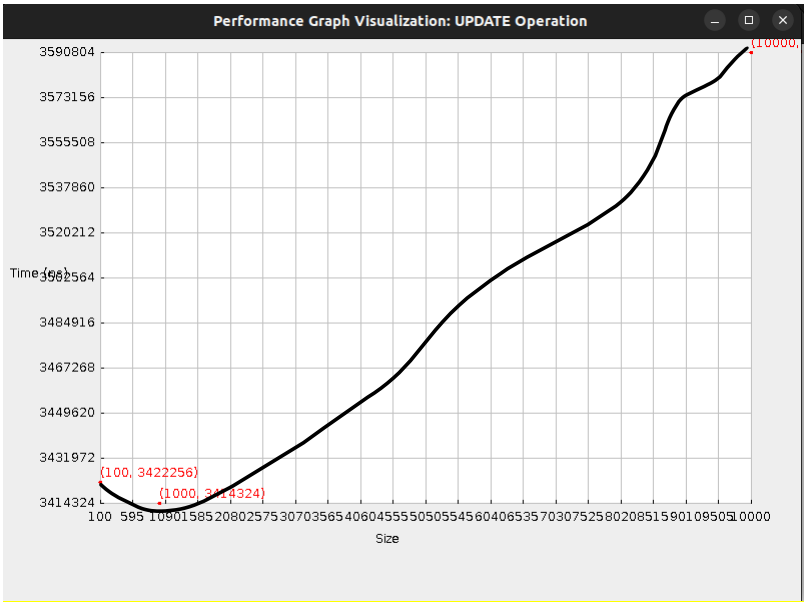
- After compilation:
    - Since there are hundreds of thousands of commands in the terminal, I cannot show them all when performing performance analysis.
    - I got different graphics every time I ran the code, even though the functions were Log(n).

```
Added stock: TXBWZ
Added stock: WDDK
Found stock: Stock{symbol='OAYJW', price=204.25, volume=8668, marketCa
p=1091432}
Stock not found: LMVK
Found stock: Stock{symbol='RCZ', price=441.72, volume=3270, marketCap=
1047267}
Found stock: Stock{symbol='XIVK', price=87.21, volume=5443, marketCap=
1054758}
Found stock: Stock{symbol='OIK', price=435.53, volume=10167, marketCap
=1029009}
Added stock: XGLJ
Found stock: Stock{symbol='RHYK', price=396.09, volume=7226, marketCap
=1086250}
Updated stock: UXLY to ETE
Added stock: PUU
Added stock: TRH
Found stock: Stock{symbol='LNXJ', price=718.0, volume=7070, marketCap=
1061521}
Found stock: Stock{symbol='LXCKI', price=798.3, volume=9229, marketCap
=1020384}
Removed stock: MXL
Found stock: Stock{symbol='CGHXN', price=208.64, volume=4803, marketCa
p=1063371}
Added stock: AMGKI
Stock not found: JAD
Found stock: Stock{symbol='QGJ', price=432.93, volume=7019, marketCap=
1053100}
Found stock: Stock{symbol='TZWX', price=930.85, volume=3941, marketCap
=1053259}
Added stock: WSP
Added stock: XSS
Found stock: Stock{symbol='IJH', price=471.16, volume=2812, marketCap=
1063715}
Found stock: Stock{symbol='IGPC', price=651.57, volume=6090, marketCap
=1061960}
Found stock: Stock{symbol='NZGXY', price=402.61, volume=8769, marketCa
p=1041359}
```




Performance Graph Visualization: ADD Operation
Time (ns)
Size
(100, 275355)
(1000, 294931)
(10000,


Performance Graph Visualization: SEARCH Operation
Time (ns)
Size
(100, 295412)
(1000, 354613)
(10000,


Performance Graph Visualization: UPDATE Operation
Time (ns)
Size
(100, 3422256)
(1000, 3414324)
(10000,