# CSE 222 HOMEWORK 7:

# Balanced Tree-based Stock Data Management

Gebze Technical University Computer Engineering Department

Project Report

Buket Gençer

210104004298

23.05.2024

## Introduction

The goal of the homework is to manage stock data using a balanced tree data structure, specifically an AVL tree. The AVL tree helps to store, retrieve, and manipulate stock information efficiently.

## Purpose and Scope of the Report

The purpose of this report is to explain the methods and results of the AVL tree-based stock data management system. This report covers:

* **The problem we are solving:** The efficient management of stock data using a balanced AVL tree structure.
* **The methods and tools we used:** Java programming language, Java Swing for the graphical user interface (GUI), and a random command generator to create test data.
* **How we collected, processed, and analyzed the data:** Generating random stock data commands, processing these commands using an AVL tree, and analyzing the time complexity of each operation.
* **The results of our analysis with visual aids:** Presenting graphs and charts that show the relationship between the size of the AVL tree and the time taken for operations like ADD, REMOVE, SEARCH, and UPDATE.
* **An interpretation of the results:** Discussing what the results mean in terms of performance and efficiency.

## Problem Definition

The problem is to efficiently manage stock data using an AVL tree. Each stock has a unique symbol, a price, a trading volume, and a market capitalization. The AVL tree must stay balanced after each operation to ensure optimal performance.

## Methods and Tools Used

**Java Swing**

Java Swing was used to create the graphical user interface (GUI). Swing provides a rich set of components for building GUIs and was suitable for visualizing the performance graphs of the AVL tree operations.

**AVL Tree**

An AVL tree is a self-balancing binary search tree where the difference in heights between the left and right subtrees of any node is at most one. This balance ensures that the time complexity for search, insertion, and deletion operations remains O(log n).

**Random Command Generator**

A custom Java program was developed to generate random stock data commands. This program creates commands like ADD, REMOVE, SEARCH, and UPDATE to simulate real-world operations on the AVL tree.

## Data Collection, Processing, and Analysis

**Data Collection**

We generated random stock data using a custom Java program. The program creates commands like ADD, REMOVE, SEARCH, and UPDATE for stocks. This simulated data was used to test the performance of the AVL tree implementation.

**Data Processing**

The generated commands were processed using the following steps:

1. **Insertion (ADD):** Adding new stocks to the AVL tree.
2. **Deletion (REMOVE):** Removing stocks from the AVL tree.
3. **Search (SEARCH):** Finding stocks in the AVL tree.
4. **Update (UPDATE):** Updating stock information in the AVL tree.

**Data Analysis**

We measured the time taken for each operation and recorded the size of the tree after each operation. We used this data to analyze the performance of the AVL tree. The results were visualized using graphs that show the relationship between the size of the tree and the time taken for each operation.

## AVL Tree and Time Complexity

**AVL Tree**

An AVL tree is a self-balancing binary search tree. In an AVL tree, the heights of the two child subtrees of any node differ by at most one. If at any time they differ by more than one, rebalancing is done to restore this property.

**Insertion Algorithm**

When inserting a new node into the AVL tree, the tree might become unbalanced. The insertion algorithm involves:

1. **Insert the node:** Follow the binary search tree insertion rule.
2. **Update the height:** Update the height of each node.
3. **Balance the tree:** Perform rotations if necessary to maintain the balance of the tree.

**Deletion Algorithm**

Deleting a node from an AVL tree might also unbalance the tree. The deletion algorithm involves:

1. **Delete the node:** Follow the binary search tree deletion rule.
2. **Update the height:** Update the height of each node.
3. **Balance the tree:** Perform rotations if necessary to maintain the balance of the tree.

**Search Algorithm**

Searching in an AVL tree follows the standard binary search tree rule. Given that the tree is balanced, the time complexity is O(log n).

**Update Algorithm**

Updating a node involves searching for the node, deleting it, and inserting the new node with updated values. This ensures the AVL properties are maintained.

**Rotations**

Rotations are used to maintain the balance of the tree:

1. **Right Rotation:** Used in the Left-Left case.
2. **Left Rotation:** Used in the Right-Right case.
3. **Left-Right Rotation:** Used in the Left-Right case.
4. **Right-Left Rotation:** Used in the Right-Left case.

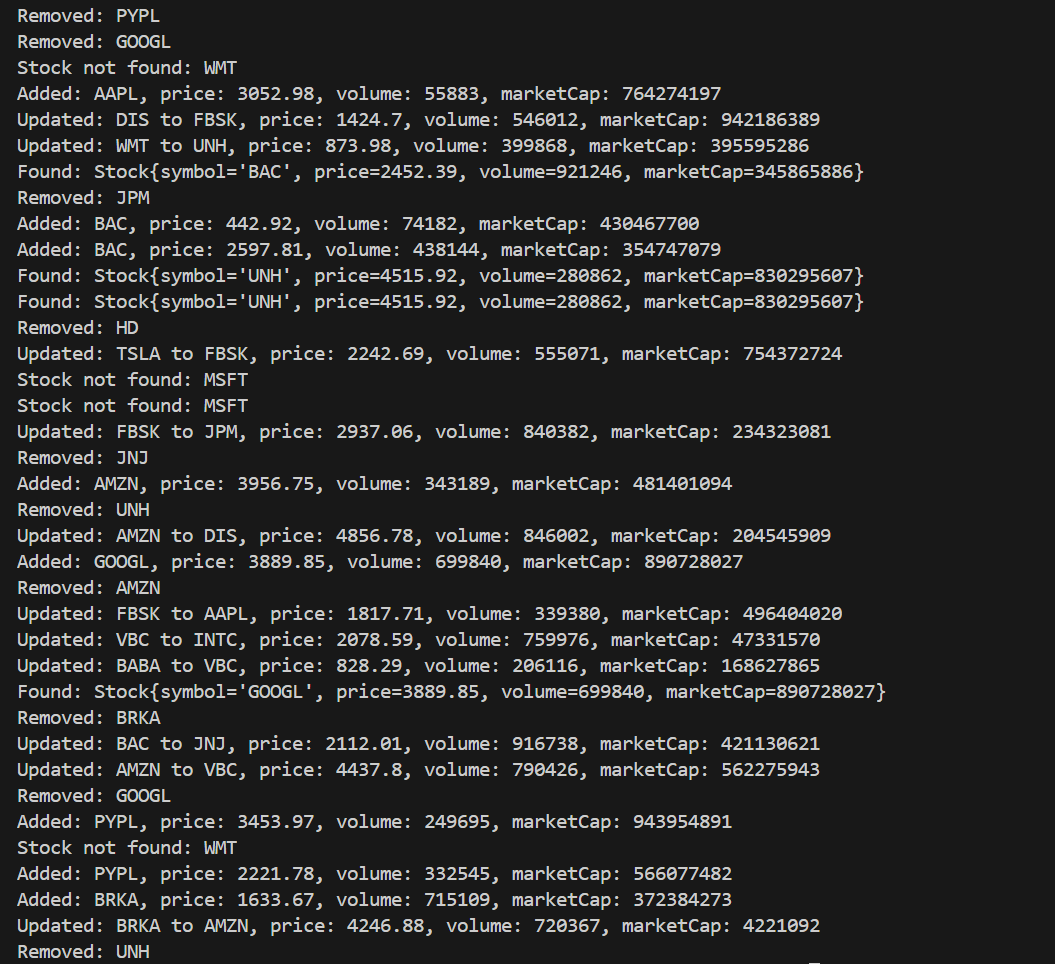
**Time Complexity**

The time complexity for insertion, deletion, and searching operations in an AVL tree is O(log n), where n is the number of nodes in the tree. This ensures efficient performance for managing stock data.

## Analysis Results and Visuals

I added diffrent output from different input (commands )file

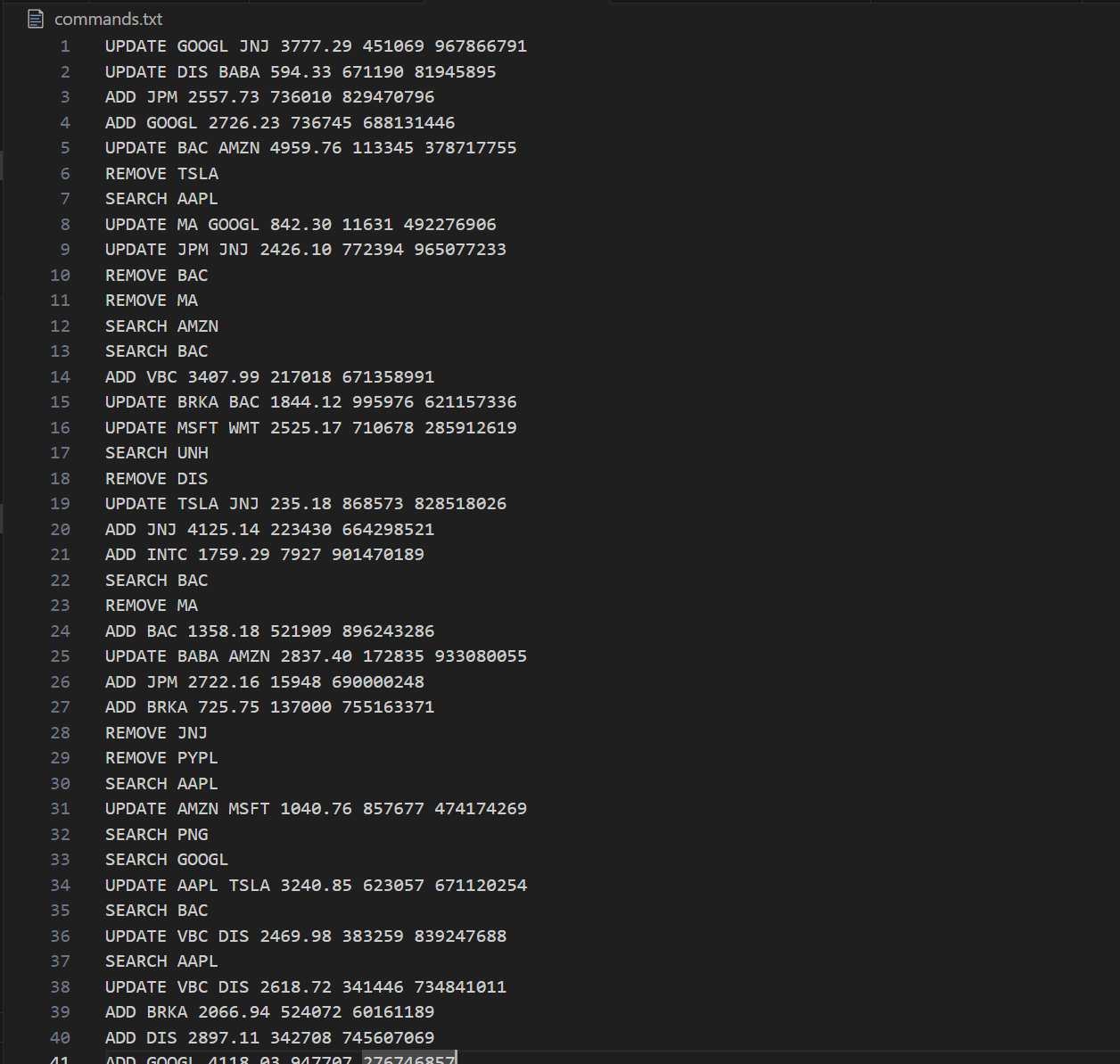
**Some Output of Terminal:**

****

**metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Examples of input file:**

****

**Add Operation**

metin, ekran görüntüsü, sayı, numara, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, sayı, numara, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**metin, ekran görüntüsü, sayı, numara, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Remove Operation**

metin, ekran görüntüsü, sayı, numara, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

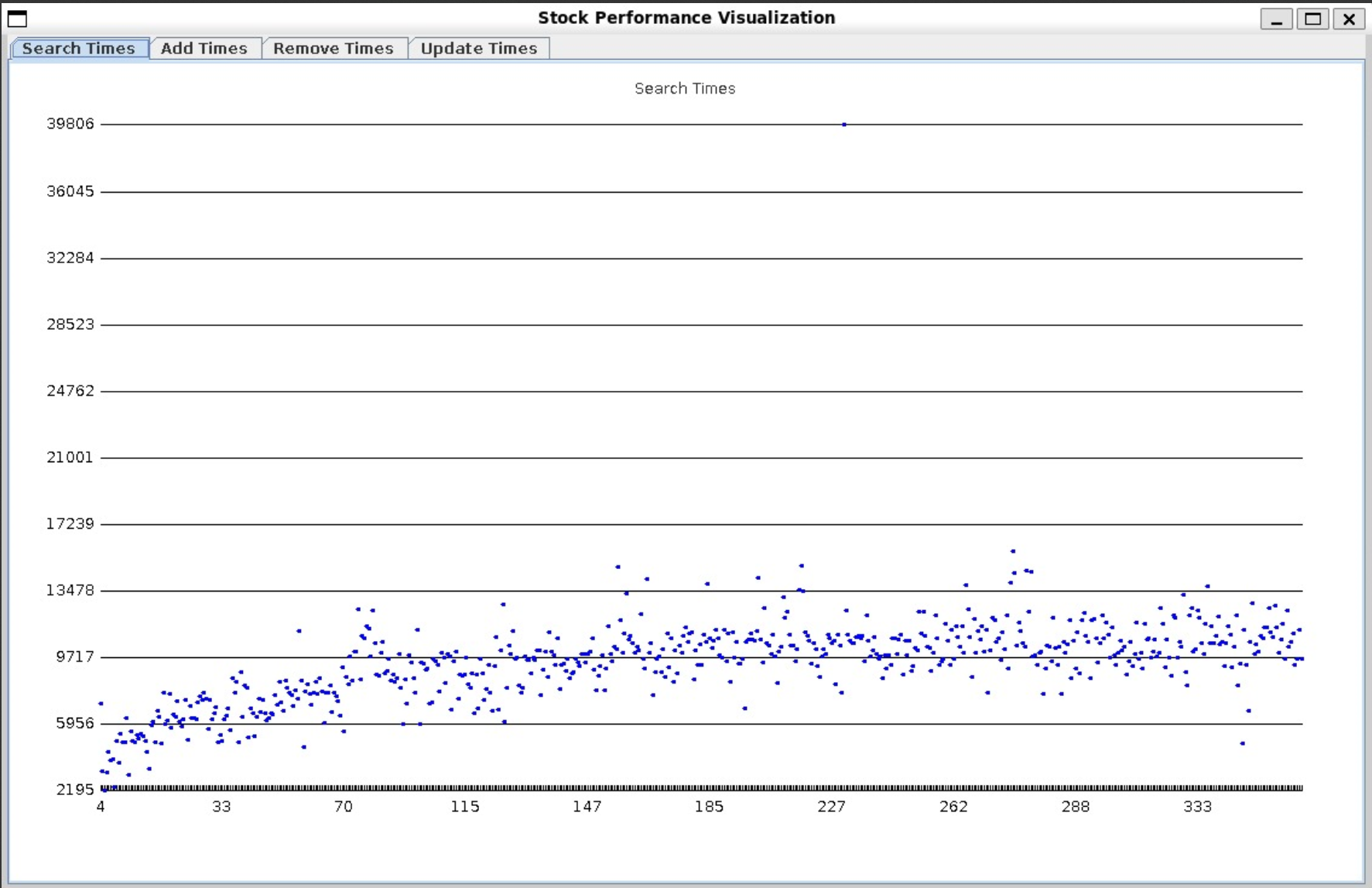
metin, ekran görüntüsü, sayı, numara, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, sayı, numara, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Search Operation**

****

**metin, ekran görüntüsü, sayı, numara, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**metin, ekran görüntüsü, sayı, numara, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Update Operation**

metin, ekran görüntüsü, sayı, numara, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, sayı, numara, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin, ekran görüntüsü, çizgi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu