GTU Department of Computer Engineering

CSE 222 / 505 – Spring 2022

Homework 7 - Report

Ahmet USLUOGLU

# 1801042602

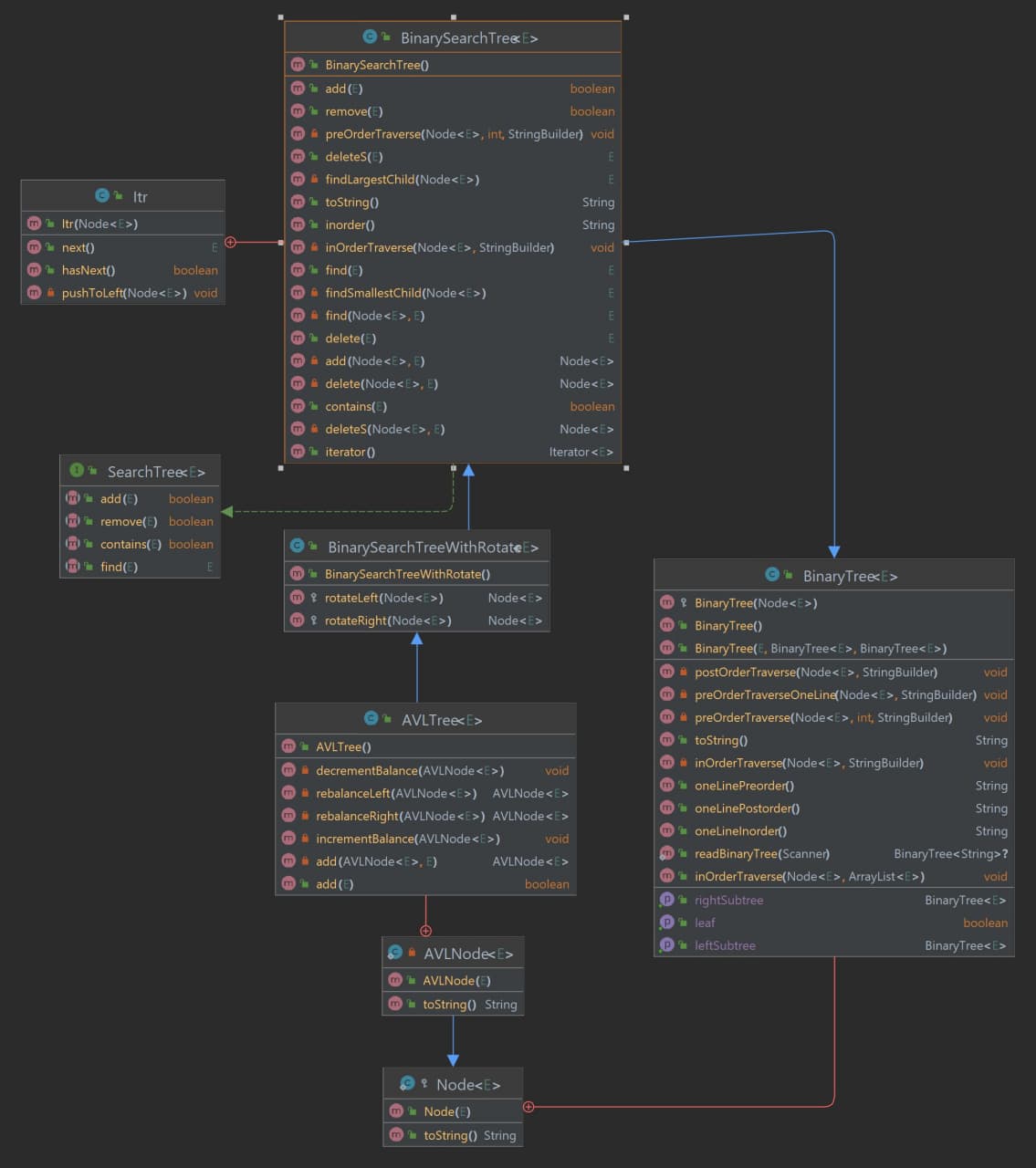
### – System Requirement

Operating System must have JDK (Java Development Kit) 11 and JRE (Java Runtime Environment) 11 or higher.

There should be enough space for storing data’s.

1. – Class Diagrams

\*Higher Resolution Version of the Class Diagram is in the files.



3. Problem - Solution Approach

Problem:

1 - ) Write a method that takes a binary tree and an array of items as input, and it returns a binary search tree (BST) as output. The binary tree contains n nodes and doesn’t need to be balanced. The array contains n unique items which are mutually comparable. The method should build a binary search tree of n nodes. The binary search tree should contain the items.

2-) Write a method that takes a binary search tree (BST) as a parameter and returns the AVL tree obtained by rearranging the BST. The method should convert the BST into an AVL tree by using rotation operations.

Solution:

1-) To create a Binary Search Tree with the same structure as Binary Tree using Array elements, I have sorted the array then traversed the Binary Tree with In-Order traversal method. While traversing set the first element in the array to first element found with traversal and removed that element from Array.

2-) To Convert the unbalanced Binary Search Tree to a balanced AVLTree , I have used AVLTree’s rotate left and rotate right functions. After each element is added AVLTree cheks the heightand Balance of the tree and rebalances the tree.

1. – Test Cases

a-) Testing the part1

1 – Adding random elements to the Binary tree and Array.

2 – Sorts the Array

3 – Making a Binary Search Tree with same structure.

b-) Testing the part2

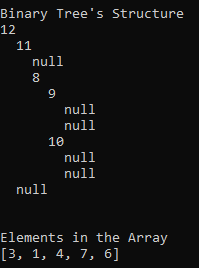
1 -) Make Unbalanced Binary Search Tree

2 -) Making a balanced AVLTree from Unbalanced Binary Search Tree

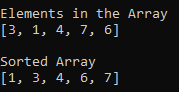
## 5 – Running Program and Results

a) Testing the part1

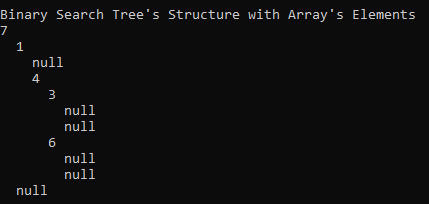
1 – Adding random elements to the Binary tree and Array.



2 – Sorts the Array

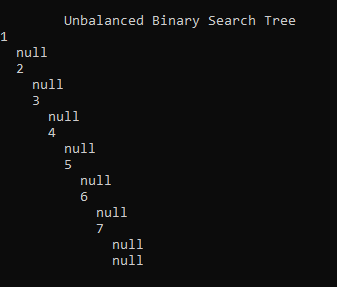


3 – Making a Binary Search Tree with same structure.

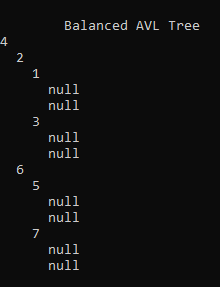


b-) Testing the part2

1 -) Make Unbalanced Binary Search Tree.

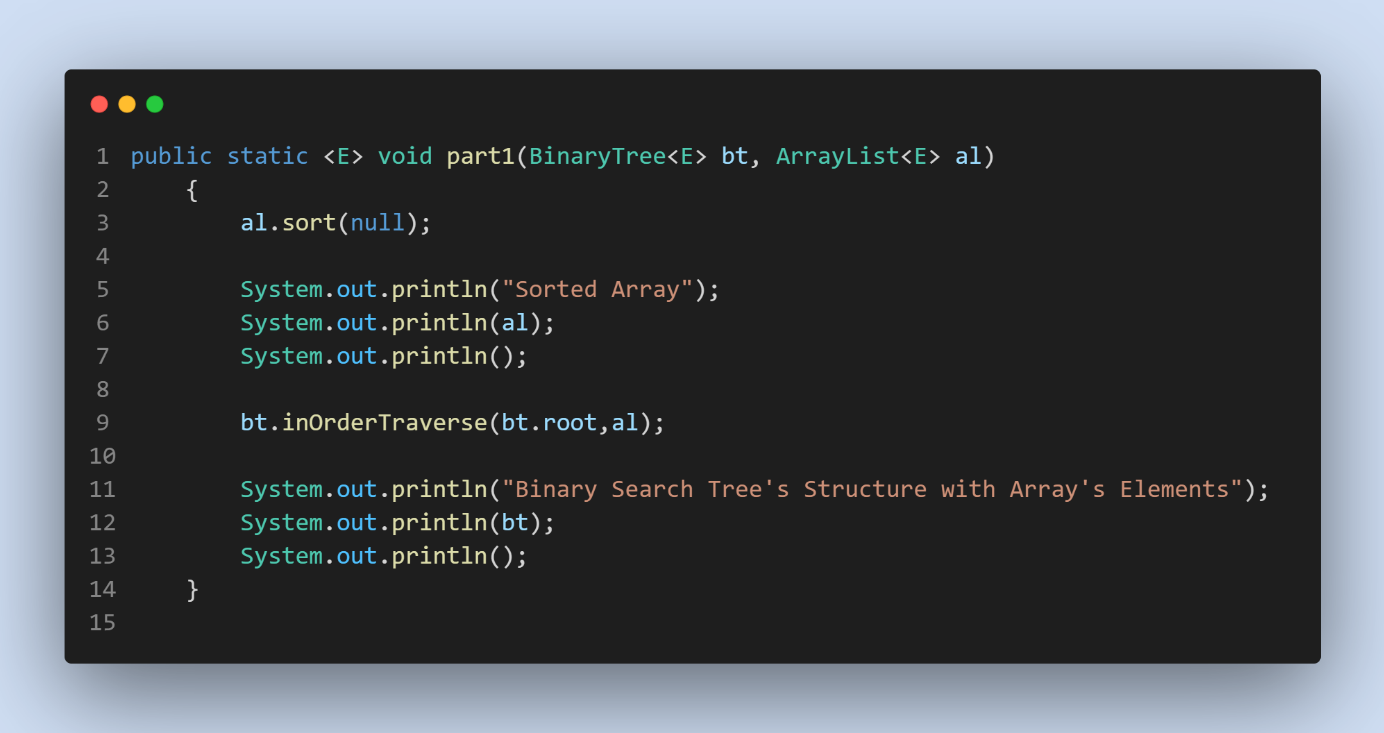


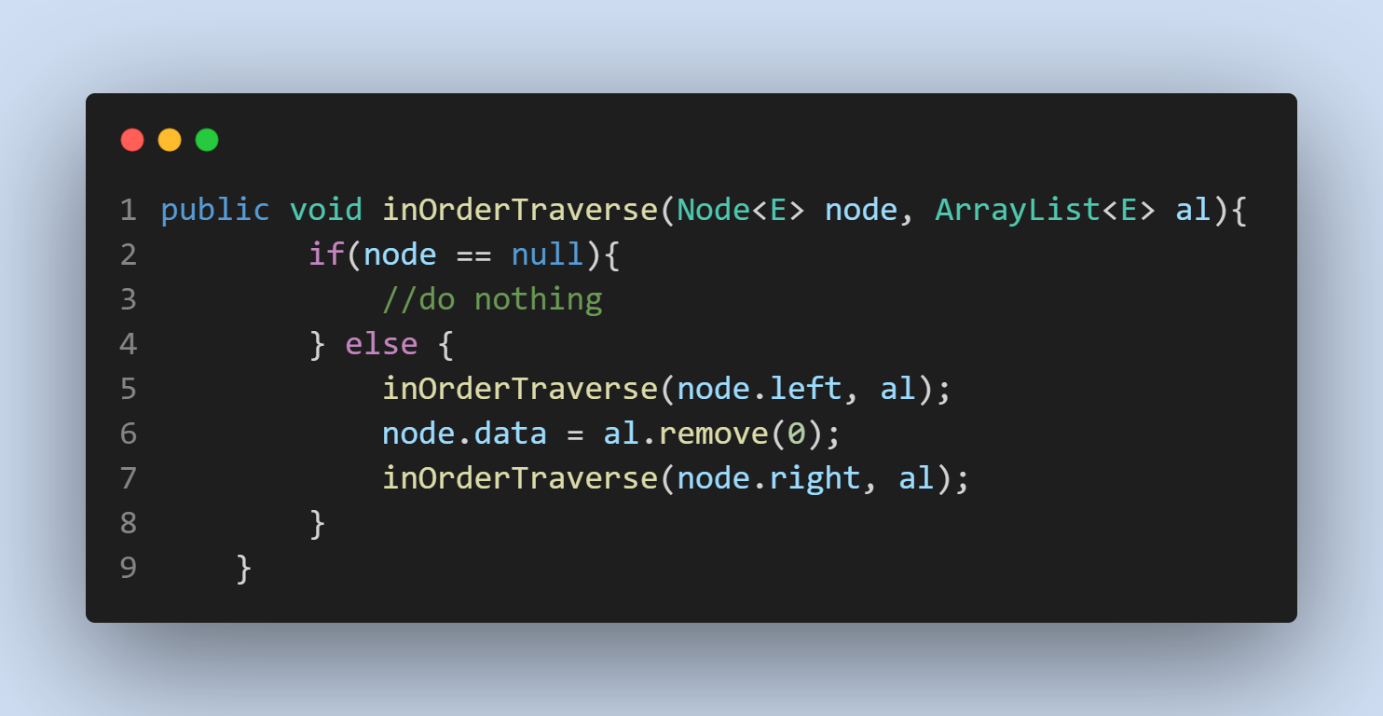
2 -) Making a balanced AVLTree from Unbalanced Binary Search Tree



### 6 – Calculate Time complexity

--Example Part1 complexity



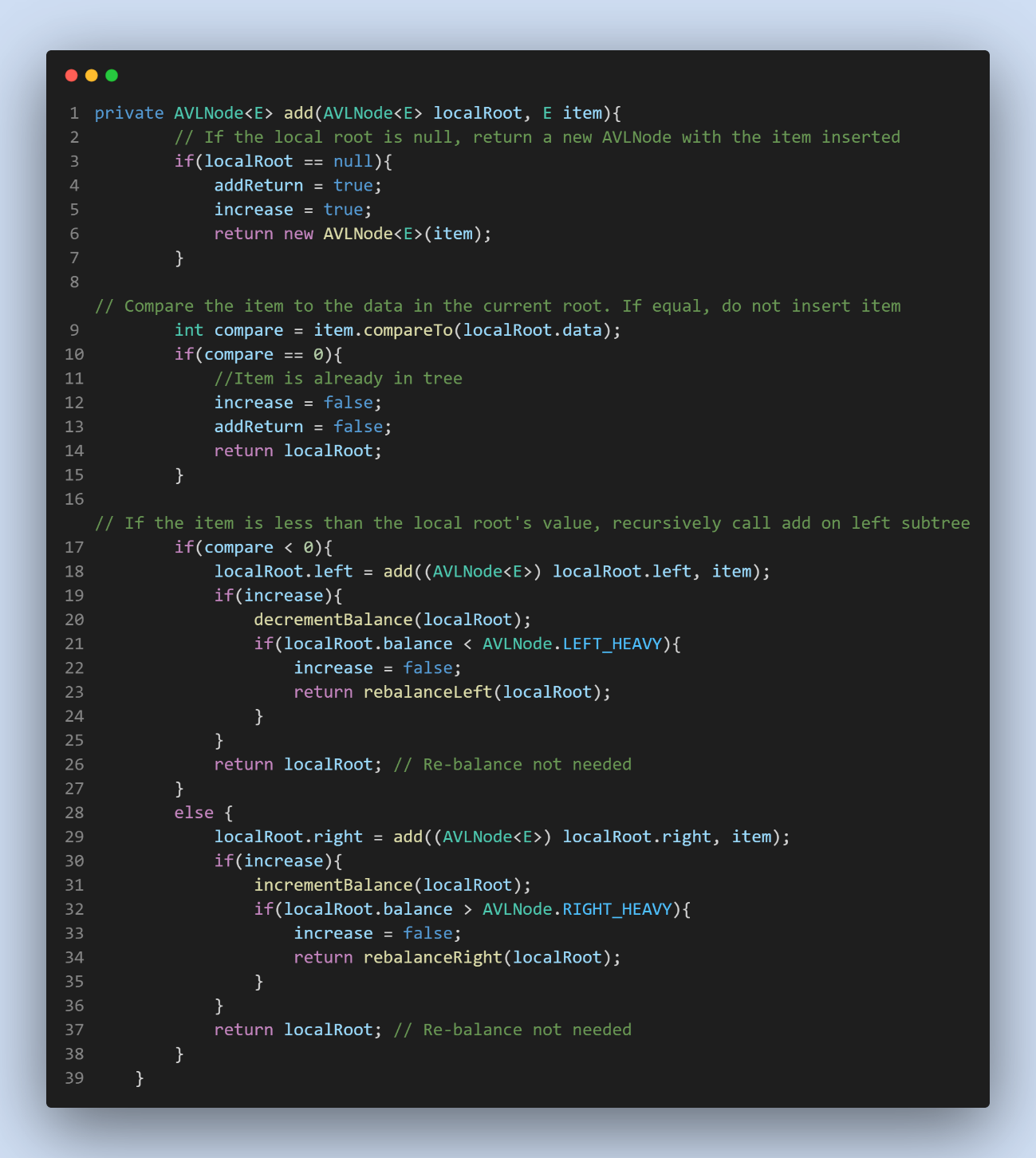


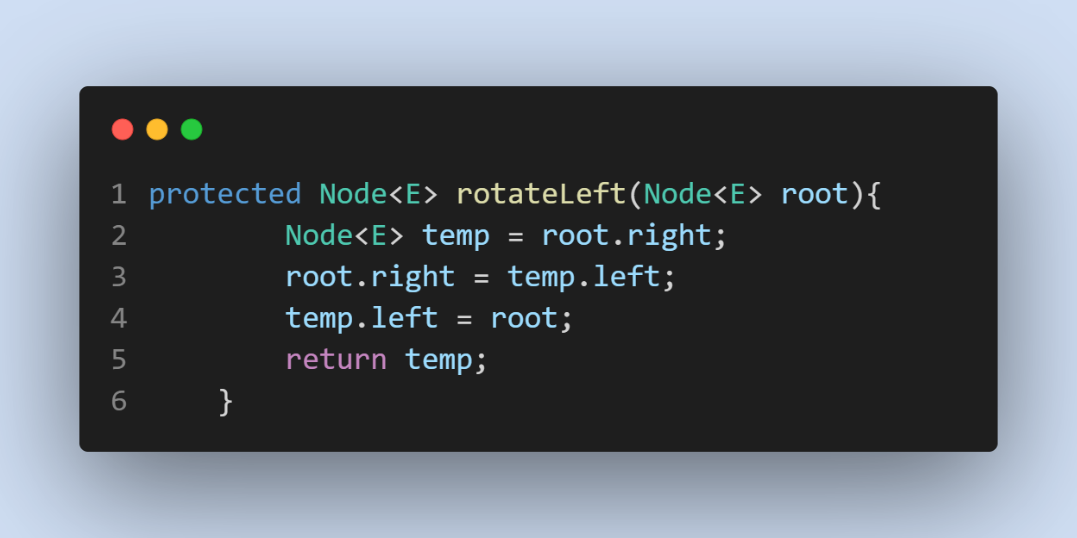
- Sorting the array complexity (Merge Sort) = O(nlogn)

- Traversing the Binary Tree takes = O(n)

- Part1 Complexity T(n) = Tetha(nlogn)

--Example Part2 Complexity







Traversing Binary Search Tree Complexity = O(n)

AVLTree insertion Complexity = O(logn)

RotateLeft, RotateRight complexity = O(1)

RebalanceLeft, RebalanceRight complexity = O(1)

Part2 complexity = Tetha (nlogn)