UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science

Faculty of Engineering, Built Environment & IT

University of Pretoria

COS212 - Data structures and algorithms

Practical 3 Specifications:

Recursion, Simple Binary Search Tree traversals

Release Date: 14-03-2022 at 06:00

Due Date: 18-03-2022 at 23:59

Total Marks: 28

Contents

1 General instructions: 2

2 Plagiarism 3

3 Outcomes 3

4 Introduction 3

5 Task 1: Binary Search Tree Traversal 3

5.1 BSTNode 4

5.2 BST 4

6 Example figure: 7

7 Submission 7

1 General instructions:

- This assignment should be completed individually, no group effort is allowed.
- Be ready to upload your assignment well before the deadline as no extension will be granted.
- You may not import any of Java's built-in data structures. Doing so will result in a mark of zero. You may only make use of 1-dimensional native arrays where applicable. If you require additional data structures, you will have to implement them yourself.
- If your code does not compile you will be awarded a mark of zero. Only the output of your program will be considered for marks, but your code may be inspected for the presence or absence of certain prescribed features.
- All submissions will be checked for plagiarism.
- Read the entire specification before you start coding.
- You will be afforded five upload opportunities.

2 Plagiarism

The Department of Computer Science considers plagiarism as a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with consent) and copying material (such as text or program code) from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to http://www.library.up.ac.za/plagiarism/index.htm (from the main page of the University of Pretoria site, follow the Library quick link, and then choose the Plagiarism option under the Services menu). **If you have any form of question regarding this, please ask one of the lecturers, to avoid any misunderstanding.** Also note that the OOP principle of code re-use does not mean that you should copy and adapt code to suit your solution.

3 Outcomes

The aim of this practical is to learn how to traverse a binary search tree and practise the art of using recursive functions.

4 Introduction

Complete the task below. Certain classes have been provided for you alongside this specification in the Student files folder. You will have to craft your own main file to test your code. Remember to test boundary cases. Submission instructions are given at the end of this document.

5 Task 1: Binary Search Tree Traversal

A binary search tree (BST) is a variant of the binary tree that makes a binary search possible. For each node x of the tree, all values stored in its left subtree are less than the value of x, and all values stored in the right subtree are greater than the value of x. This allows a complexity between O(log n) and O(n) for search, insert and delete operations within a sequence of n elements. The shape of the tree determines the complexity, which can change when the tree is updated.You have been given a partially implemented binary search tree class and a binary search tree node class to use. Your task is to implement the following methods in the binary search tree class according to the given specification:
Please note that [] is square brackets

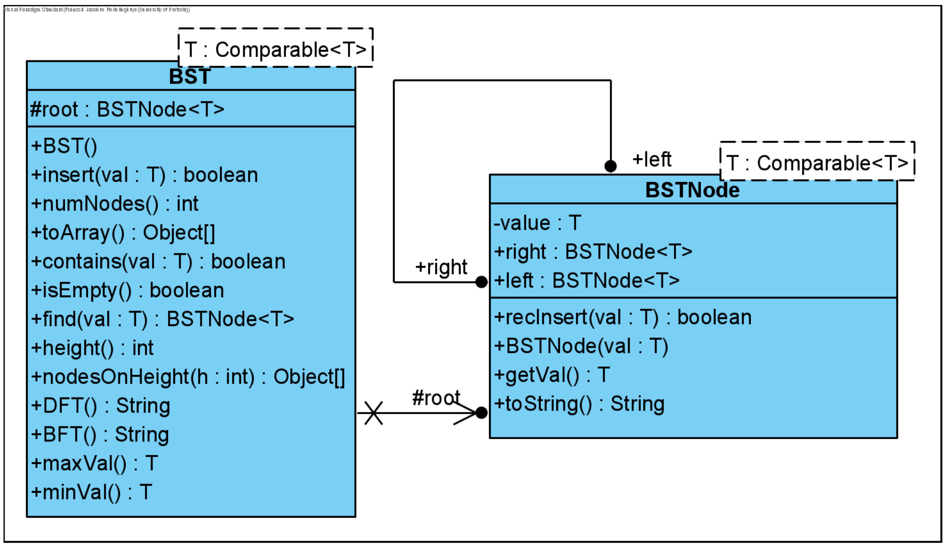

Figure 1: Complete class diagram

5.1 BSTNode

- getVal(): T
  - This function should return the value of the node.
- toString(): String
  - This function should return a string representation of the node. The format is as follows: L[*left's value*]V[*current node's value*]R[*right's value*]
    - If left is null use the following for left: L[]
    - If right is null use the following for right: R[]
  - Using the example figure the following nodes' toString() will be as follows:
    - Node with value of 73: L[11]V[73]R[95]
    - Node with value of 32: L[]V[32]R[60]
    - Node with value of 72: L[]V[72]R[]
- BSTNode(val: T)
  - The constructor should just initialize the value member with the passed in val parameter.

5.2 BST

- numNodes(): int
  - This function should return the number of nodes in the BST.
  - If the root is null then the function should return 0.
  - Using the example figure the number of nodes should be: 9

- toArray(): Object[]
  - This function should return an array containing all the nodes in the BST.
  - The order of the nodes should be in ascending order from smallest value to highest value.
  - If the tree is empty the function should return null.
  - Using the example figure the values of the objects in the array should be the following: 7;11;32;60;72;73;82;87;95
- contains(val: T): boolean
  - This function should return true if the passed val is present in the BST. If it is not present the function should return false.
- isEmpty(): boolean
  - This function should return true if the BST is empty. If the BST is not empty it should return false.
- find(val: T) BSTNode<T>
  - This function should return the node that has the same value as the passed in val parameter.
  - If no node in the BST has the same value as the passed in val parameter the function should return null.
- height(): int
  - This function should return the height of the BST.
  - If the root is null then the function should return -1;
  - Using the example figure the height of the tree would be: 4
- nodesOnHeight(h: int): Object[]
  - This function should return all the nodes on the height specified by the passed in h parameter in an array.
  - If h < 0 or root is null or h is greater then the height of the tree then the function should return null.
  - The nodes in the resulting array should be sorted in ascending order from lowest value to highest value.
  - Using the example figure the values of the objects in the array will be as follows assuming h=2: 7;32;82

- DFT(): String
  - This function should return the toString() of all the nodes in the order of a depth first traversal as a semi-colon separated list.
  - If the root is null an empty string should be returned.
  - Use inorder traversal to construct the resulting string.
  - Using the example figure the returned string should be as follows:
    L[]V[7]R[];L[7]V[11]R[32];L[]V[32]R[60];L[]V[60]R[72];L[]V[72]R[];L[11]V[75]R[95];
    L[]V[82]R[87];L[]V[87]R[];L[82]V[95]R[]
- BFT(): String
  - This function should return the toString() of all the nodes in the order of a breath first traversal as a semi-colon separated list.
  - If the root is null an empty string should be returned.
  - Using the example figure the returned string should be as follows:
    L[11]V[73]R[95];L[7]V[11]R[32];L[82]V[95]R[];L[]V[7]R[];L[]V[32]R[60];L[]V[82]R[87];
    L[]V[60]R[72];L[]V[87]R[];L[]V[72]R[]
- maxVal(): T
  - This function should return the largest value that is present in the BST.
- minVal(): T
  - This function should return the smallest value that is present in the BST.

Implement the methods listed above. You may only use recursion to traverse the BST, but may use iterative loops when working with arrays. **The only function that may be solved using an iterative solition is BFT().** You may use your own helper functions to assist in implementing the specification. However you may not modify any of the given method signatures. Also do not modify any of the other code that you were given for this task.

6 Example figure:

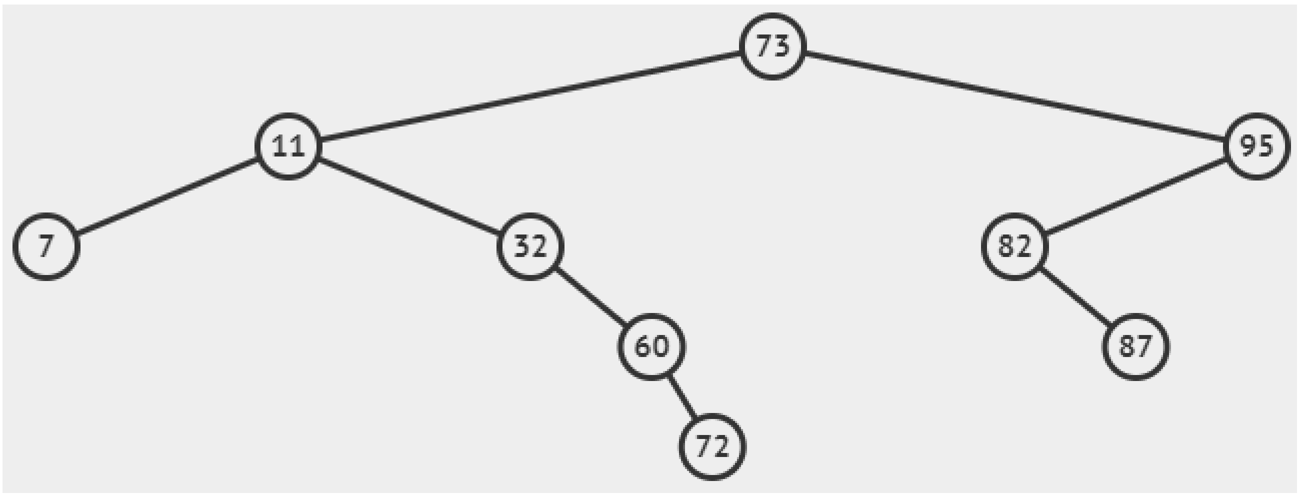

Figure 2: Binary Search Tree Example

7 Submission

You need to submit your source files on the Fitch Fork website (https://ff.cs.up.ac.za/).All methods need to be implemented (or at least stubbed) before submission. Place your BST.java and BSTNode.java file in a zip archive named uXXXXXXXX.zip where XXXXXXXX is your student number. There is no need to include any other files in your submission. Your code should be able to be compiled with the following command:

javac BST.java BSTNode.java

You have 5 submissions and your best mark will be your final mark. Upload your archive to the Practical 3 slot on the Fitch Fork website. Submit your work before the deadline. **No late submissions will be accepted!**