Hacettepe University

Department of Computer Engineering

BBM204 Software Laboratory II

Spring 2019

Experiment 2

Subject : *Binary Search Tree*

Submission Date : *01.04.2019*

Due Date : *19.04.2019*

Programming Language: *Java SE - JDK8*

Background

A binary search tree (BST) is a rooted binary tree, whose internal nodes each store a key (and optionally, additional values) and each has two distinguished sub-trees, commonly denoted left and right. The tree additionally satisfies the binary search property, which states that the key in each node must be greater than or equal to any key stored in the left sub-tree, and less than or equal to any key stored in the right sub-tree. Equal keys cannot be stored in both left and subtrees and they must be stored in the left subtrees or in the right subtrees. The leaves of the tree do not have any children.

Frequently, the information represented by each node is a record rather than a single data element. However, for sequencing purposes, nodes are compared according to their keys rather than any part of their associated records. The major advantage of binary search trees over other data structures is that the related sorting algorithms and search algorithms can be very efficient; they are also easy to code. [1]

Experiment

In this assignment, you will implement a Java program that reads a list of commands from the input file and executes the job described by these commands. Some of these commands will create a new binary search tree and they will set the current tree to this newly created binary search tree. Some of them can perform certain operations on the current (lastly created) binary search tree. Note that you will have a single binary search tree for which you perform the operations during the main function and the initial binary search tree should be an empty binary search tree. The program should read the commands from the input file and write the results of all the commands to the output file, so that each command has one row. Input and output file names will be given as arguments to the program. You should use object-oriented architecture for binary search trees.

Your program should work for the following commands.

COMMAND	MEANING
CreateBST *n1,n2, …* **(10pt)**	Insert *N* (where *N* > 0) distinct integers (*n1*, *n2*, …) into an empty binary search tree. For instance, if the command line includes "*CreateBST 6,3,8,1,9* ", your function should insert the integers 6, 3, 8, 1, and 9 into an empty binary search tree in the given order. Set the current binary search tree to this binary search tree after the execution of this command. Your program should output is "*BST created with elements n1,n2, …*". The elements should written with the inorder traversal.
CreateBSTH *H* **(10pt)**	Create a full binary search tree with a height of *H* (*H* > 0) using the integers 1, 2, .., 2^H-1. Set the current binary search tree to this binary search tree after the execution of this command. Your program should output is "A full *BST created with elements n1,n2 …*". The elements should written with the inorder traversal.
FindHeight **(10pt)**	Find and write the height of the current binary search tree.
FindWidth **(10pt)**	Find and write the width of the current binary search tree. The width of a tree is the number of nodes at a level that contains the maximum number of nodes.
Preorder **(5pt)**	Write all nodes of the current binary search tree using preorder traversal.
LeavesAsc **(10pt)**	Write all leaves of the current binary search tree in ascending order without using a sorting algorithm.
DelRoot **(10pt)**	Delete the root of the current binary search tree if it exists.
DelRootLc **(5pt)**	Delete the left child of root of the current binary search tree if it exists.
DelRootRc **(5pt)**	Delete the right child of root of the current binary search tree if it exists.

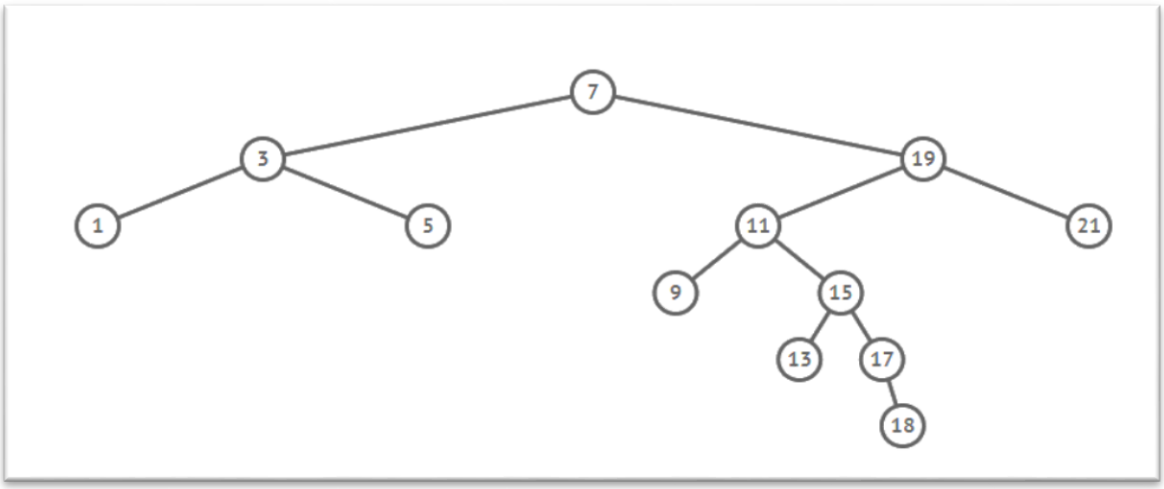
Input-Output Format

Your program will read input file, construct a binary search tree and perform certain operations on the binary search tree. Input file contains several **command lines** that your program will execute. Input file will be error-free and integer values will be unique. Neither extra characters nor spaces will be included in input file. A sample input file and its corresponding tree representation can be shown below:

Input file:

```
CreateBST 7,3,1,5,19,11,9,15,13,17,18,21
```

Visualization of tree



Your program will read input file and perform certain operations on the current (lastly created) binary search tree.

Example Input file:

```
CreateBST 7,3,1,5,19,11,9,15,13,17,18,21
FindHeight
FindWidth
Preorder
LeavesAsc
DelRoot
Preorder
DelRootLc
LeavesAsc
DelRootRc
Preorder
CreateBSTH 3
Preorder
```

You must split input file into commands by using New Line character as delimiter. According to the given example input file, after the commands are executed program should write output lines to the output file should be as follows.

Output File:

```
BST created with elements:1 3 5 7 9 11 13 15 17 18 19 21
Height:6
Width:4
Preorder:7 3 1 5 19 11 9 15 13 17 18 21
LeavesAsc:1 5 9 13 18 21
Root Deleted:7
Preorder:9 3 1 5 19 11 15 13 17 18 21
Left Child of Root Deleted:3
LeavesAsc:1 13 18 21
Right Child of Root Deleted:19
Preorder:9 5 1 21 11 15 13 17 18
A full BST created with elements:1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Preorder:8 4 2 1 3 6 5 7 12 10 9 11 14 13 15
```

Note that every command requires the tree objects to be updated. Avoid replacing a node's (node to be deleted) integer value with another (related neighbour's) integer value. Your program must manage and dominate tree objects. **(10pt)**

Arguments and Execution

>cd n010101010/src

>javac Exp2.java **(5pt)**

>java Exp2 input.file output.file **(5pt)**

NOTES

- Use Eclipse while development
- Use UNDERSTANDABLE names for your variables, functions, and classes (please be sure you are obeying name convention).
- Write READABLE SOURCE CODE blocks.
- Use EXPLANATORY COMMENTS in your source codes.
- Don't miss the deadline.
- CAUTION: Don't start thinking about assignment lately. Experiment requires less code but more algorithmic approach.
- Save all your work until the assignment is graded.
- The assignment must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating.
- You can ask your questions through course's piazza group and you are supposed to be aware of everything discussed in the piazza group. General discussion of the problem is allowed, but DO NOT SHARE answers, algorithms, source codes.
- You will submit your work from https://submit.cs.hacettepe.edu.tr with the file hierarchy as below:

|--src

-- Exp2.java

-- *.java

REFERENCES

[1] https://en.wikipedia.org/wiki/Binary_search_tree