Lab 10 - Binary Search Trees

Due Sunday, October 25 at 23:55

Objective

Today's lab will help you get familiar with

- How to implement a generic binary search tree class.

More about Binary Search Trees

In class, you are learning about Binary Trees and Binary Search Trees, and discussed various operations, such as insertions and traversals that can be performed on these trees. Chapters 18 and 19 of your book give more information on Binary Search Trees, particularly on how to implement these trees and operations. In addition, in those chapters you can find very useful information regarding Binary Trees and traversals. Be sure to read the chapters. You might also need to use the material you covered in class. You are encouraged to use the algorithms in the chapters/from class for this assignment.

Assignment:

For this lab assignment, you will write a binary search tree class. The main idea is to write a **Tree** interface, a **BinaryTree** abstract class, and a **BinarySearchTree** class:

1. Write a generic interface called **Tree<E>** with the following methods:
    a. **boolean insert(E e)** - Inserts the given key to the tree, if it is not already present (duplicate keys are not allowed in the tree). Returns true if insertion was successful.
    b. **Boolean contains(E e)** - Returns true if the tree contains the specified key.

    c. **int numOfElementsDepth(int i)** - returns the number of elements at a certain depth i.
    d. **E findMax()** - Finds and returns the largest item in the tree.
    e. **E findMin()** - Finds and returns the smallest item in the tree.
    f. **String preOrderString()** - Returns a String representation using preorder traversal.
    g. **String postOrderString()** - Returns a String representation using postorder traversal.
    h. **String inOrderString()** - Returns a String representation using inorder traversal.
    i. **void empty()** - Removes all the items from the tree.
    j. **boolean isEmpty()** - Returns true if there are no items in the tree, false otherwise.
2. Write a **BinaryNode** class which you will use in your **BinaryTree** (next section). The binary node should have all the necessary methods and instance variables necessary (The book can be of help for that).
3. Write a generic abstract class called **BinaryTree** that implements the **Tree** interface. This class is abstract because this is just a general binary tree and therefore certain methods such as insert cannot be defined and will be left abstract. However other methods can be implemented here:
    a. All the traversal methods which return a String.
    b. The numOfElementsDepth() method.
    c. The empty/isempty() methods.

    The rest of the methods will be implemented in the BinarySearchTree (next section). Use recursion when appropriate.
4. Finally write the **BinarySearchTree** class for which all the rest of the necessary methods are implemented.
5. Write another class with a main method which creates a binary search tree (for data type Integers) and shows its capabilities.
    a. Create a BinarySearchTree of Integers
    b. The user should be able to pass a list of integers through the command line. **For example,** if calling the program in the following manner:

```
java MainClass 23 4 67 1 57 3 18 7
```

The numbers 23,4,67,1,57,3,18,7 should be inserted into the tree. Notice that the user should be able to make the tree as big as they'd like.

c. Try to insert a number which is already in the tree and print out that it was unsuccessful.
d. Search for one number in the list, and one number which is not in the list and print out the results.
e. Print out the number of elements at some depth.
f. Print out the largest element in the tree.
g. Print out the smallest element in the tree.
h. Print out the three strings for the three traversals.
i. Empty the tree, and print out that the tree is indeed empty.

Submission:

For this lab, you need to submit the code and lab notes. In the notes include a screenshot running part 6 from the command line.

Grading:

1. unit testing - 1
2. Tree - 1
3. BinaryNode - 1
4. BinaryTree- 2
5. BinarySearchTree - 2
6. main - 1
7. Lab notes - 1
8. Style/commenting - 1