Binary Search Trees

Take-home ESC/Java2 problem, May 20, 2011

Deadline for solutions: June 3, 2011

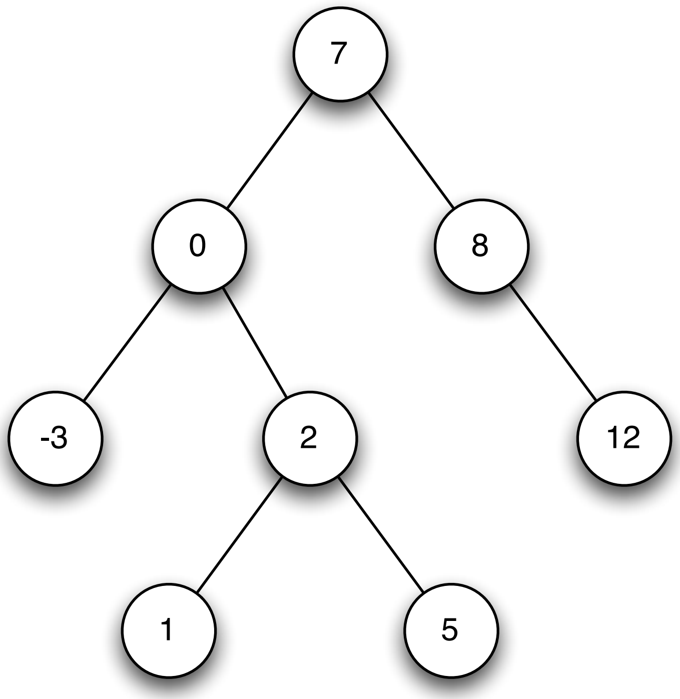
1. Introduction

Binary search trees (BST) are binary trees, not necessarily balanced, where an integer number is stored in every node, with the property that for every node *w*:

- every number stored in the left subtree of *w* is smaller than the number stored in *w*, and

- every number stored in the right subtree of *w* is bigger than the number stored in *w.*

Here is an example of a BST:



A class that implements BSTs should provide at least the following methods:

- creating an empty tree,

- checking if a given tree is empty,

- checking if a given number occurs in a tree,

- adding a given number to a tree (if the number already occurs in the tree, this method should not change the tree),

- finding the least number in the tree,

- removing the least number from the tree.

BSTs implemented in this fashion can be used to sort arrays of integer numbers with duplicate removal, in the following way:

1. Create an empty tree *D*,

2. Add all numbers from the given array to *D* one by one,

3. Remove the least number from *D* until it becomes empty.

2. Task

Write a Java class that implements BSTs as described above. Add a static method that will use BSTs to sort sequences of integer numbers.

Then, annotate your code with JML annotations and use ESC/Java2 to verify that the result of the sorting procedure is a sorted array of numbers. (You do not have to verify that the result contains the same numbers as the input array).

To this end, you may used all verification methods provided by ESC/Java2, including invariants, `ghost` variables etc. However, remember to avoid unjustified use of `assume` and `nowarn` annotations.

3. Emergency version (for 2 pts.)

If the above task turns out too difficult, you may omit the verification of the property described above, and ensure only that ESC/Java2 processes your class without generating any warnings. To do this, you must help it to verify (among other things) that the sorting procedure will never attempt to remove an element from an empty tree.