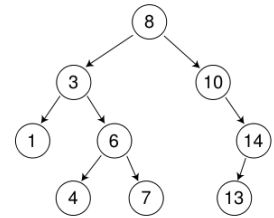


BINARY SEARCH TREE

Binary search trees are data structures that have low memory overhead and still allow fast (less than $O(N)$ in all but the worst case, $O(\log N)$ if the tree is balanced) searches.

I. Language Agnostic Overview

Binary trees, or trees where any node may have no more than two children, have a special property in that the average depth for any node is $O(\sqrt{N})$. Binary search trees, however, have an average depth of $O(\log N)$. This makes them an important data structure as they allow rapid access to any node with little memory overhead.



The included implementation of a binary search tree was taken from the data structures textbook “Data Structures and Algorithm Analysis in Java” by Mark Allen Weiss.

Implementations of binary search trees are typically required to satisfy the following constraints:

1. Each node must have a value (in this implementation values must be unique)
2. A total order is defined on these values.
3. The left subtree of a node contains only values less than the node's value.
4. The right subtree of a node contains only values greater than or equal to the node's value.

II. Java Specific Overview

Insert

- a. This inserts a new element into the tree.
- b. The precondition calls the repOK function (explained below) on the root node. The postcondition again calls the repOK function.

repOK

- a. This method performs a number of checks to make sure that the binary search tree is consistently structured. The first check sees if the current node is null, if so, the method returns true. The second check calls inRange (explained below) which verifies that the current value is within the proper range for the subtree. Finally, the repOK function recursively calls itself on the left and right subtrees.

Note: There are two versions of the repOK function. The other version (with signature `boolean repOK(BinaryNode t)`) simply calls this one with the entire tree as the range.

b. This method is used to check pre and postconditions, so it has no conditions of its own.

inRange

a. This method makes sure that a value is either infinity (in the positive or negative direction) or smaller than the largest element and larger than the smallest. This method is recursively called on the tree and all subtrees to verify that no element is out of order.

b. This method is used to check pre and postconditions, so it has no conditions of its own.

Metrics:

Lines of code for implementation: 130

Methods for implementation: 23

Lines of code for invariants: 28

Methods for invariants: 4

References:

Wikipedia's entry on binary search trees: http://en.wikipedia.org/wiki/Binary_search_trees

Data Structures and Algorithm Analysis in Java, 2nd Edition by Mark Allen Weiss. Section 4.3: “The Search Tree ADT – Binary Search Trees.” pgs 112 – 122.

Algorithms in Java, 3rd Edition by Robert Sedgewick. Section 12.6 “Binary Search Trees.” pgs 531 – 536.