Binary Search Trees

A binary search tree (BST) is a node-based binary tree data structure where each node has a comparable key (and an associated value).

Each node has no more than two child nodes. The left sub-tree contains only nodes with keys less than the parent node; the right sub-tree contains only nodes with keys greater than the parent node.

Binary search trees are a fundamental data structure used to construct more abstract data structures such as sets, multisets, and associative arrays.

A BST supports the following operations:

Search: We begin by examining the root node. If the key equals that of the root, we return the node.

If the key is less than that of the root, we search the left subtree. Else, we search the right subtree.

This process is repeated until the key is found or the remaining subtree is *null*.

On average, this means that each comparison allows the operations to skip over half of the tree.

Insert: In order to insert a new node in the tree, its key is first compared with that of the root.

If its key is less than the root's, it is then compared with the key of the root's left child. If its key is greater, it is compared with the root's right child.

This process continues, until the new node is compared with a leaf node, and then it is added as this node's right or left child, depending on its key.

Delete: There are three possible cases to consider:

Deleting a node with no children: simply remove the node from the tree.

Deleting a node with one child: remove the node and replace it with its child.

Deleting a node with two children: call the node to be deleted *N*. Do not delete *N*. Instead, choose either its in-order successor node or its in-order predecessor node, *R*.

Copy the value of *R* to *N*, then recursively call delete on *R* until reaching one of the first two cases.