Search Tree Implementation

A binary search tree relies on the property that keys that are less than the parent are found in the left subtree, and keys that are greater than the parent are found in the right subtree. We will call this the bst property. As we implement the Map interface as described above, the bst property will guide our implementation. Figure 1 illustrates this property of a binary search tree, showing the keys without any associated values. Notice that the property holds for each parent and child. All of the keys in the left subtree are less than the key in the root. All of the keys in the right subtree are greater than the root.



Now that you know what a binary search tree is, we will look at how a binary search tree is constructed. The search tree in Figure 1 represents the nodes that exist after we have inserted the following keys in the order shown: 70,31,93,94,14,23,7370,31,93,94,14,23,73. Since 70 was the first key inserted into the tree, it is the root. Next, 31 is less than 70, so it becomes the left child of 70. Next, 93 is greater than 70, so it becomes the right child of 70. Now we have two levels of the tree filled, so the next key is going to be the left or right child of either 31 or 93. Since 94 is greater than 70 and 93, it becomes the right child of 93. Similarly 14 is less than 70 and 31, so it becomes the left child of 31. 23 is also less than 31, so it must be in the left subtree of 31. However, it is greater than 14, so it becomes the right child of 14.

To implement the binary search tree, we will use the nodes and references approach similar to the one we used to implement the linked list, and the expression tree. However, because we must be able create and work with a binary search tree that is empty, our implementation will use two classes. The first class we will call BinarySearchTree, and the second class we will call TreeNode. The BinarySearchTree class has a reference to the TreeNode that is the root of the binary search tree. In most cases the external methods defined in the outer class simply check to see if the tree is empty. If there are nodes in the tree, the request is just passed on to a private method defined in the BinarySearchTree class that takes the root as a parameter. In the case where the tree is empty or we want to delete the key at the root of the tree, we must take special action.