CS61B, Spring 2003 **Discussion #11** **Amir Kamil**
UC Berkeley **3/31/03**

Topics: Binary Search Trees

1 Binary Search Trees

If we are most interested in searching for individual items in a collection, or of obtaining a sorted list of all the items, then we use a *binary search tree* (BST). Such a tree introduces only a single constraint among the data items, that an inorder traversal on the tree visits the values in their natural order. This requires the tree to have two properties:

- The subtree to the left of a value may only contain values less than it.
- The subtree to the left of a value may only contain values greater than it.

It is easy to see that an inorder traversal on a tree that obeys the two properties will visit values in order. Thus we can obtain an ordered list of elements by executing an inorder traversal on the tree.

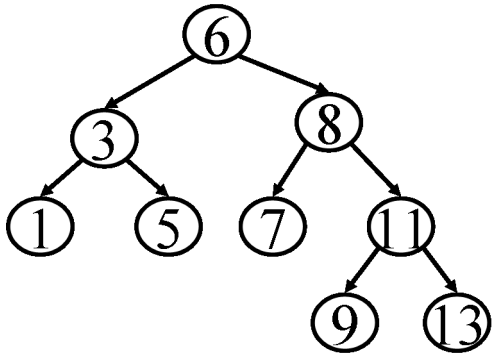

Figure 1: A binary search tree, with integer values.

Using the two BST properties, searching for an element is straightforward. Starting at the root, go left if the target is less than the value at the current node, right if it is greater. If it is equal, then of course you've found the element. To insert an item in a BST, use the search algorithm until you've hit a *leaf node* (a node with no children), and insert the item properly to its left or right.

Removal from a BST is a little trickier, since removing a node from a tree also removes its descendants. What if the value you wish to remove is not a leaf node? Then you must replace the node rather than just

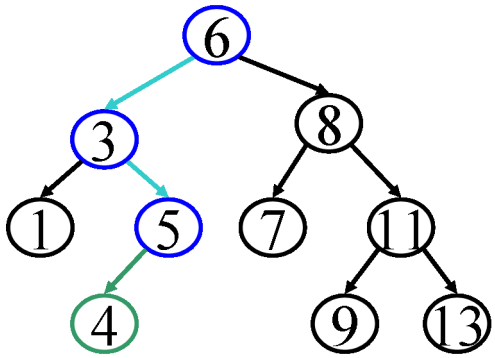

Figure 2: Insertion of the value 4 into the BST.

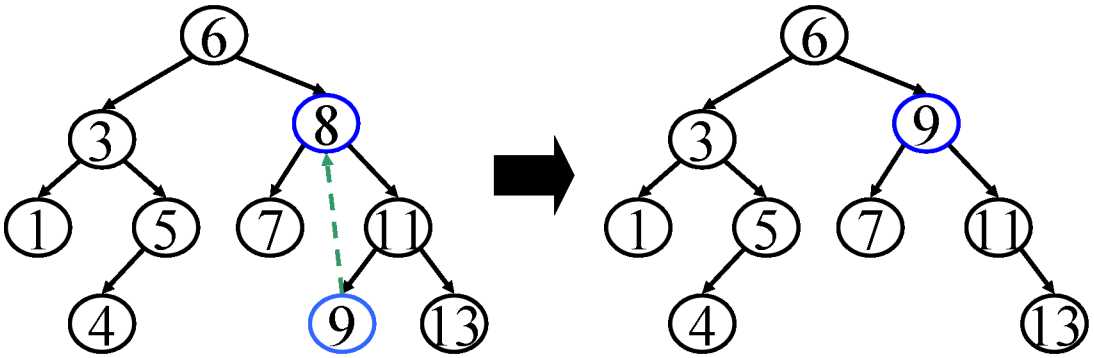

Figure 3: Removal of the value 8 from the BST.

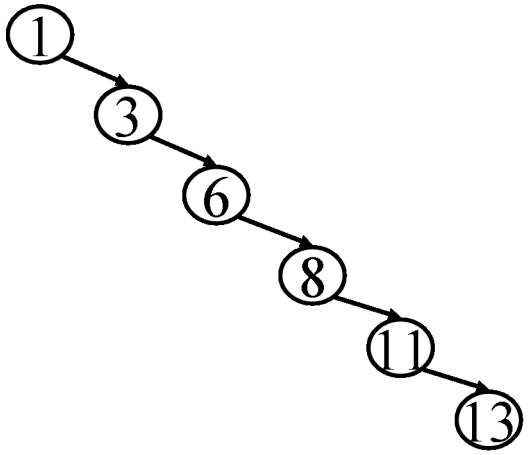

Figure 4: A binary search tree constructed in order.

remove it. Inspection of a BST reveals that both the leftmost node in the right subtree or the rightmost node in the left subtree will always work as replacements, so either may be used.

Analysis of the above operations reveals that retrieving an ordered list of values is linear in the number of values in the tree, and the other three operations are linear in the height of the tree. But what is the height of the tree? A balanced tree has about $\lg n$ height were n is the number of elements, but a BST is not necessarily balanced. Consider a BST constructed by adding elements in order. Searching, insertion, and removal all take $O(n)$ time in this case! But in the average case, we would expect them to take $O(\lg n)$.