BST insertion

To insert a new value into a BST, we will insert it as a leaf. It will have no children. We only have to determine its parent.

So, there is only one possible location: where it would have been if we had looked for it.

Insert runs in O(h)

Delete from a BST

Three cases:

1. No children (a.k.a leaf)
2. Has one child
3. Has two child

Case 0): deleting a leaf is easy: just set its parent’s child pointer to NULL and freeit

Case 1): to delete a node with one child, set its parent’s child pointer to its child and free it

Case 2): to delete a node with two children: DON’T!

Actually, we replace case 2 with either case 0 or 1: replace the value in the node to delete with either the largest value in its left subtree or the smallest value in its right subtree.

This will not break the search order.

Now just delete the original copy. It will be either case 0 or 1 because it cannot have 2 children.

Insert, delete, search, min, max all run in O(h)

On average, BSTs have a height that is O(logn)

In the worst case, all operations would be O(n)