# Discussion Week 5 - Binary Search Trees

**Last updated Spring 2018**

## Topics Covered

- Binary search tree characteristics
- Binary search tree terminology
- Important binary search tree operations
- Binary search tree performance

## Mini-lecture

A *binary search tree* is a binary tree that maintains the magic *Binary Search Tree Property*, which states that for any given node:

- Its left child's value is less than or equal to that of the node's
- Its right child's value is greater than that of the node's
- Each of its children is *also* a valid binary search tree

This last point is the one most often forgotten. Be certain to check for this part, too.

Also, note that this assumes that a left/right child both exist. If they don't, then you're good to go.

### Definitions

*Node, Left Child, Right Child*: These are the same as for ordinary trees. We're just putting these here to remind you of the words that show up a lot when we talk about binary trees.

*Full Binary Search Tree*: A BST in which every node has 0 or 2 children. Note that this does not mean that the tree has good performance.

*Degenerate Binary Search Tree*: A BST in which every node has at most 1 child. These are linked lists.

*Balanced Binary Search Tree*: A BST in which the difference between the maximum and minimum height of leaf nodes is at most one. These will have a height of `O(log n)`, which is what we want.

*Complete Binary Search Tree*: A balanced BST that also has all of its leaves as far left as is possible.

*Perfect Binary Search Tree*: A BST for the fastidious among us. A complete binary search tree where all of the leaves are the same height.

## Worksheet

You've been given a `BST` class for making binary trees that store integers. Implement the `static` methods `Maximum`, `Minimum`, `Contains`, and `Valid`. For this class. Unit tests have been provided for you for all four.

## Microquiz

1. Which traversal order prints the elements of a binary search tree from least to greatest? (2 words, if you don't hyphenate)
2. What is the worst case time complexity for minimum, maximum, insertion, deletion, and search of a binary search tree? What about a *balanced* binary tree? (20 words max.)
3. For checking that the binary search tree property is held by a given tree, why is it inadequate to simply check if each node's children are less than/greater than the node itself? (10 words max.)

*Once you've submitted your microquiz at the link provided you, you're good to go.*