Understand how inserting into a Binary Search Tree (BST) works.

Draw BSTs that are built by inserting numerical values in the following orders:

- Tree 1
  - 5 3 4 8 6 7 2 1 9
- Tree 2
  - 1 2 3 4 5 6 7 8 9
- Tree 3
  - 5 4 3 6 7 2 8 9 1
- Tree 4
  - 8 4 12 2 6 10 14 1 3 5 7 9 11 13 15

Compare your results to those of your fellow students. Make sure you all have the same understanding of how a BST works.

Which of these trees are full?

Now look at the last tree.
How many nodes are in the tree? (*also round that up to the nearest* 2^d)
How many comparisons are needed, at the most, to find any specific value in the tree?
Imagine you add a whole new level to the tree.
How many nodes are now in the tree?
What is that compared to the previous number of nodes? (*both rounded to nearest* 2^d)
How many comparisons are now needed to find any specific value in the tree?

What does this say about the time complexity of finding a value in a Binary Search Tree?