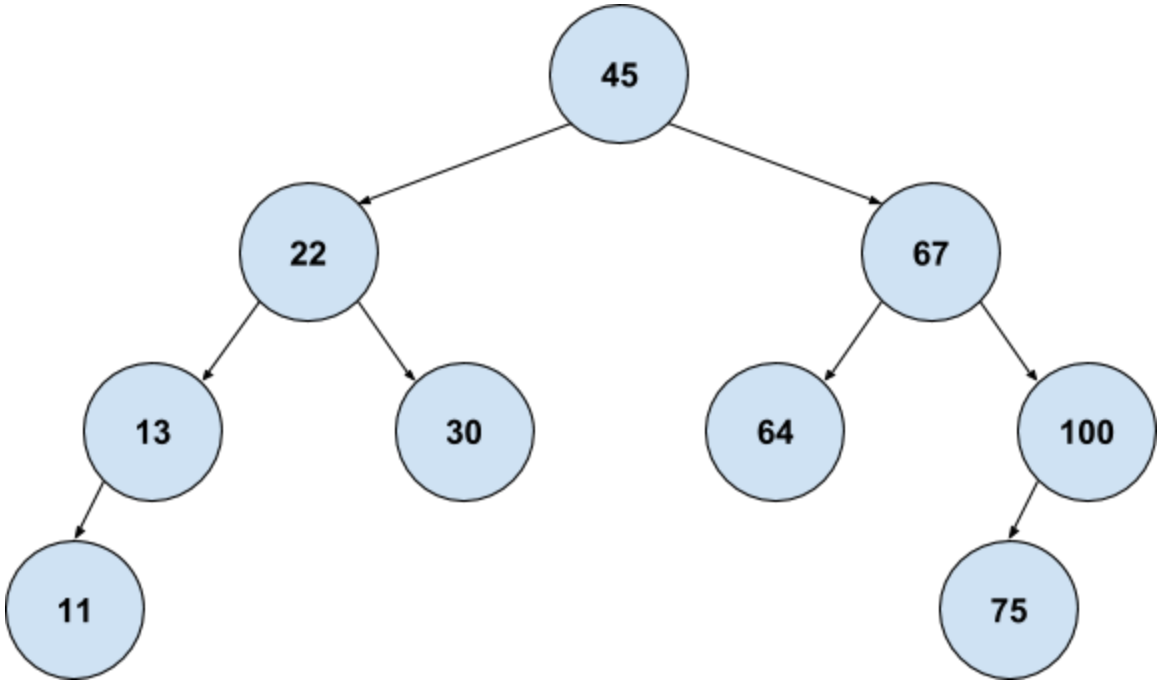
Worksheet 28: Binary Search Trees

Group Members:
Erin Alltop
Nathan Fraser
Joshua Kevin Groves
David Luke Hartman
Matthew Toro

In Preparation: Read Chapter 8 to learn more about the Bag data type, and chapter 10 to learn more about the basic features of trees. If you have not done so already, read Worksheets 21 and 22 for alternative implementation of the Bag.

In this worksheet we will practice the concepts of using a Binary Search Tree for the Bag interface. For each of the following problems, draw the resulting Binary Search Tree.

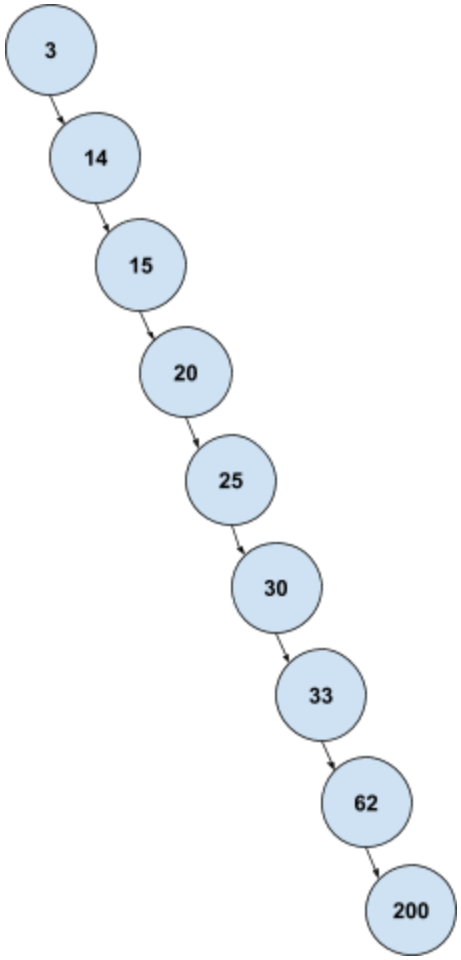
1. Add the following numbers, in the order given to a binary search tree. 45, 67, 22, 100, 75, 13, 11, 64, 30



2. What is the height of the tree from #1? What is the height of the subtree rooted at the node holding the value 22? What is the depth of the node holding the value 22?

The height of the tree from # 1 is 3. The height of subtree rooted at the node holding 22 is 2. The depth of the node holding 22 is 1.

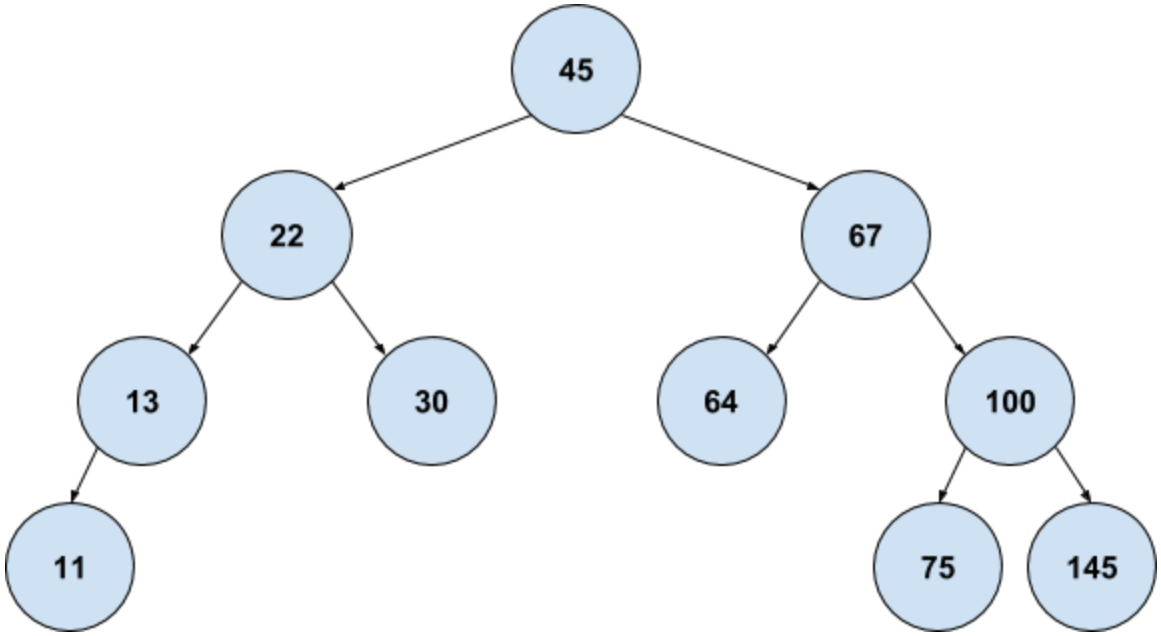
3. Add the following numbers, in the order given to a binary search tree. 3, 14, 15, 20, 25, 30, 33, 62, 200.



4. Is the tree from #3 balanced? Why not? What is the execution time required for searching for a value in this tree?

No, the tree is not balanced because the tree is not full, the entire left subtree is missing. The execution time for searching this tree would be O(n) as it is simply a line.

5. Add a new value, 145, to the tree from #1



6. Remove the value 67 from the tree from #1. What value did you replace it with and why?

67 was replaced with 75 because 75 was the leftmost child of the right subtree of the lowest level.