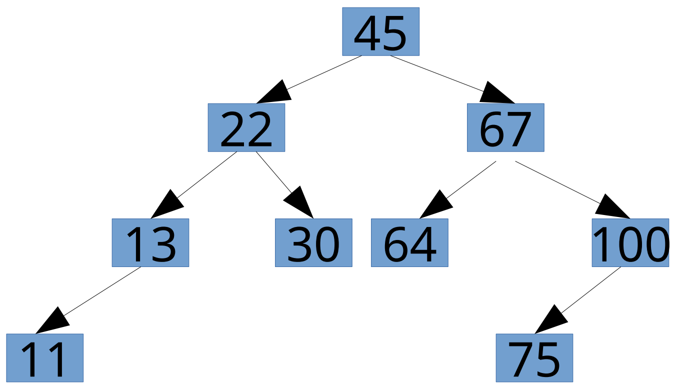
Worksheet 28: Binary Search Trees

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.

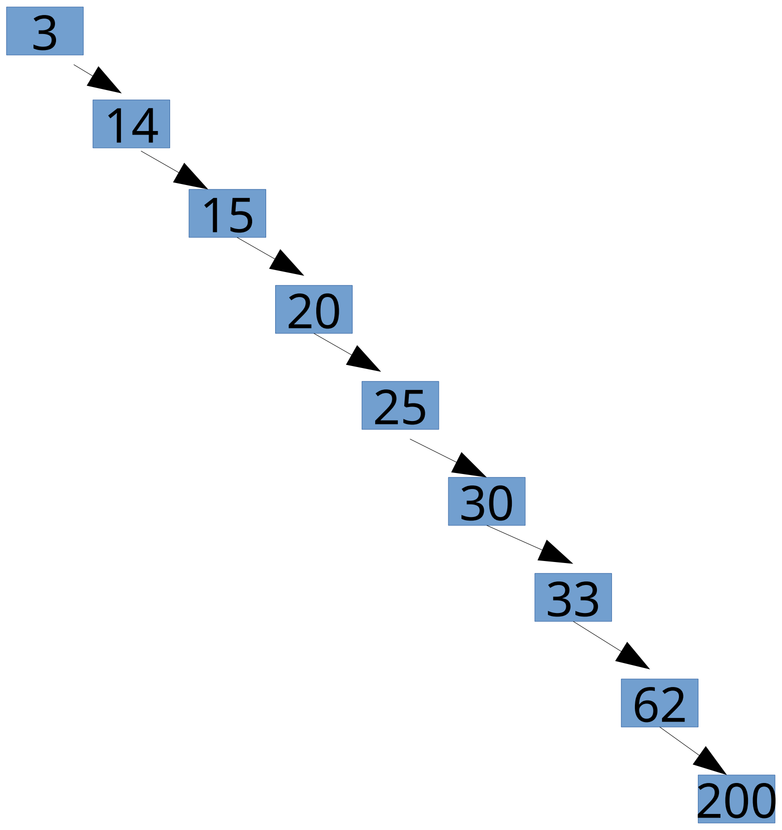
In this worksheet we will practice the concepts of using a Binary Search Tree for the Bag interface. For each of the following problems, draw the resulting Binary Search Tree.

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



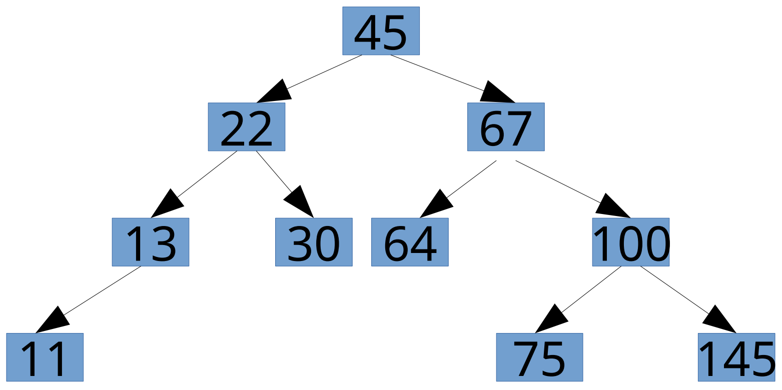
2. What is the height of the tree from #1? What is the height of the subtree rooted at the node holding the value 22? What is the depth of the node holding the value 22?
Height of subtree rooted at 22 is 2. Depth is 1.

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

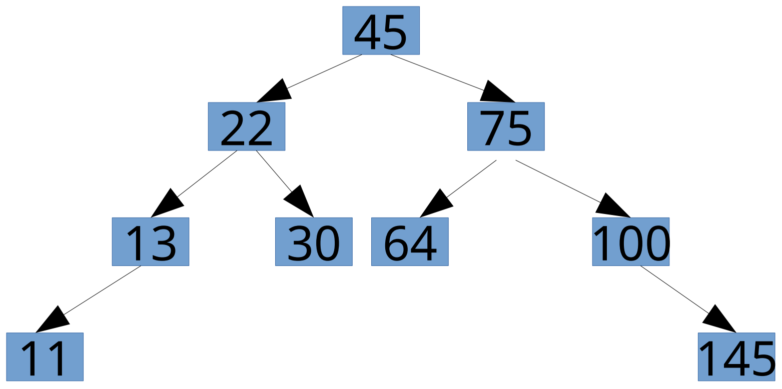


4. Is the tree from #3 balanced? Why not? What is the execution time required for searching for a value in this tree?
Not at all balanced. There is no left subtree of the root node, or any node. The execution time for searching this tree is O(n). Makes sense, this tree looks like a linked list.

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?



I replaced 67 with its left most child node in the right subtree, 75. This keeps the order of the tree node values.