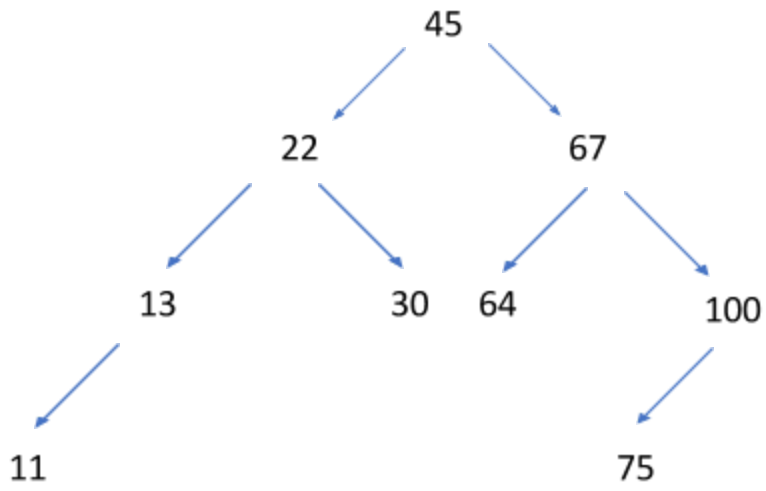
Worksheet 28: Binary Search Trees

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



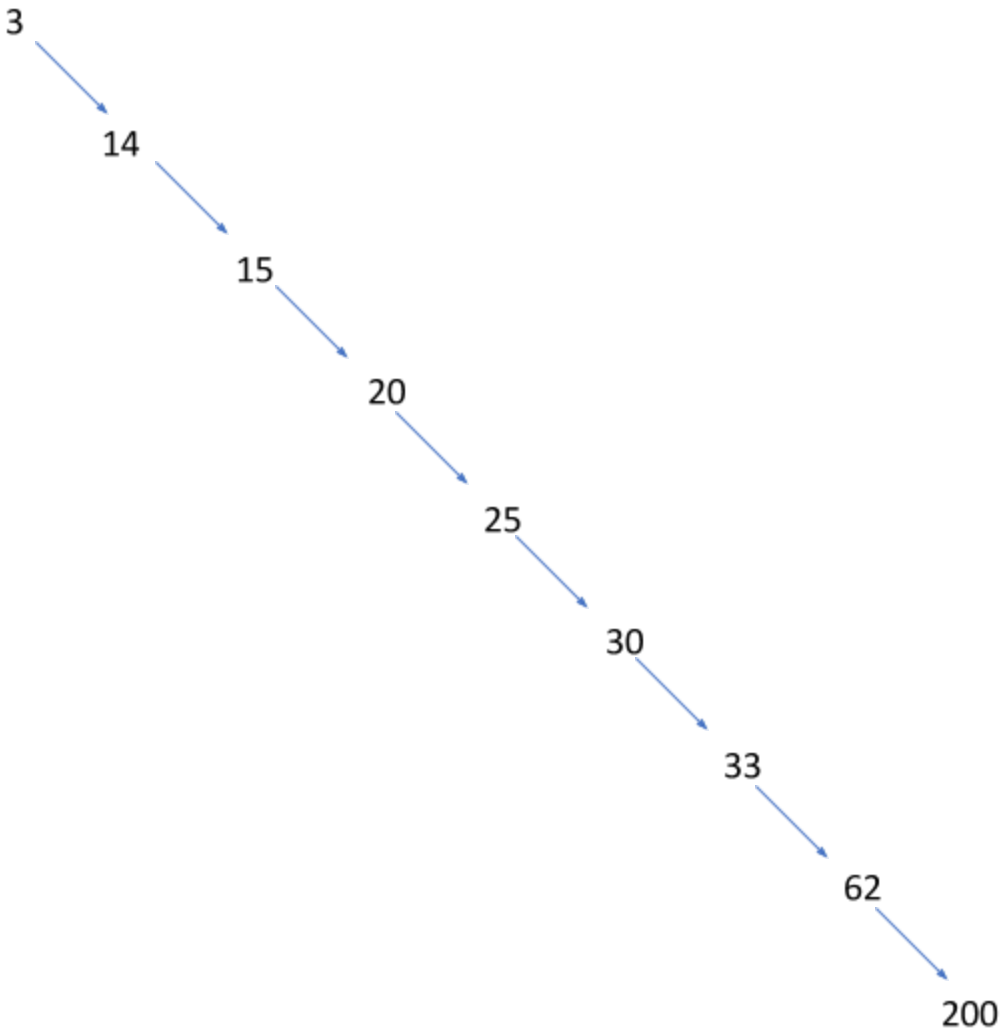
2. A. What is the height of the tree from #1? B.What is the height of the subtree rooted at the node holding the value 22? C.What is the depth of the node holding the value 22?

   A.The height of the tree is 3 because there are 3 arcs to traverse between the root node to the farthest leaf node.

   B.The height of the subtree rooted at the node holding the value 22 is 2.

   C.The depth of the node holding the value 22 is 1 (length from the root to the node).

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

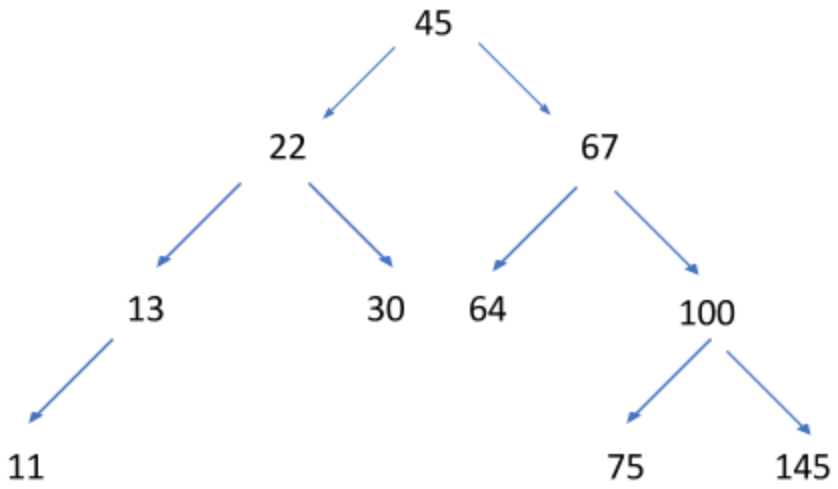


4. Is the tree from #3 balanced? Why not? What is the execution time required for searching for a value in this tree?

The tree is not balanced because there are no nodes to the left of any nodes. There is no left subtree.

In this tree, the elements are inserted in order, which is equivalent to a simple linked list (p. 14, Chapter 10). The execution time for searching a value in this tree is O(n).

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?

The value 67 is replaced with 75 because it is the leftmost child of the right subtree.