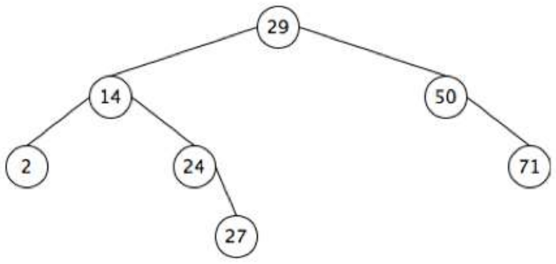
Practice Questions on Binary Trees (Week 7)

Tutorial Questions

1. This question concerns the following binary tree:



   a) Which is the root node?
   b) Which are the leaf nodes?
   c) Which nodes are the parents of 14?
   d) Which nodes are the children of 14?
   e) Which nodes are the descendants of 14?
   f) Which nodes are the ancestors of 14?
   g) Which nodes are the siblings of 14?

2. Build a binary search tree (BST) based on alphabetical ordering for the words **Le**xus, **Me**rcedes, **B**MW, **A**udi, **La**mborghini, **C**hevrolet, **P**orsche, **Ma**serati, **L**otus.

   a) Draw the BST obtained by inserting the words in the order they are listed above. How many comparisons are needed to locate the following words respectively:
      i. **A**udi,
      ii. **La**mborghini,
      iii. **Ma**serati,
      iv. **P**orsche?

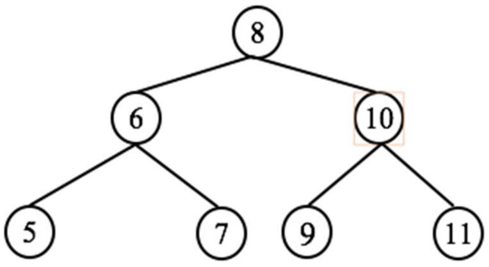
   b) Draw the BST obtained by inserting the words in alphabetical order (i.e. we will first insert **A**udi, followed by **B**MW, **C**hevrolet, and so on in alphabetical order). How many comparisons are needed to locate the following words respectively:
      i. **A**udi,
      ii. **La**mborghini,
      iii. **Ma**serati,
      iv. **P**orsche?

3. Andy is organizing a badminton tournament. Each game is played between two participants. The winner goes to the next round. The loser is eliminated. There are no ties. If there are an odd number of participants in a particular round, one unpaired participant will advance directly to the next round.

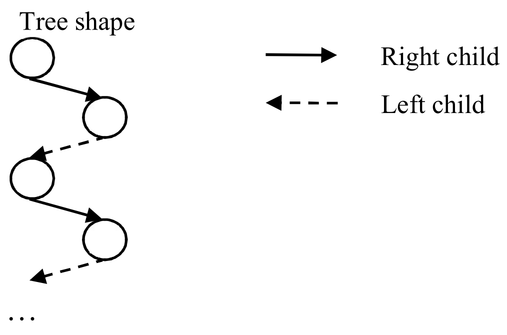
   a) Use a binary tree to determine how many games need to be played to determine the champion if there are 14 participants.
   b) We can view the tournament as an algorithm, which takes as input the number of participants **n**, and returns as output the champion. If each game is a step in this algorithm, what is the complexity of the algorithm?

4. We would like to design a sorting algorithm called TreeSort. It works by inserting each data element into a BST. To print the data elements in a sorted order, we traverse the tree in a certain order.
   a) Which traversal method do we use for this purpose?
   b) What is the complexity of the above algorithm?

5. Given an array of distinct numbers, we seek to determine whether the input array could conceivably be the postorder traversal of some BST containing elements in that array. For example, if the input is `a = [5, 7, 6, 9, 11, 10, 8]`, there exists a BST (shown below) for which `a` is the postorder traversal. For another example, if the input is `b = [3, 1, 2]`, there is no BST containing these integers could possibly have `b` as its postorder traversal.
   a) Design an algorithm (in pseudocode or Python code) that will return **True** if the input array could conceivably be the postorder traversal of some BST, and return **False** otherwise.
   b) What is the worst-case complexity of your algorithm?



6. Given an array **A** = [2, 10, 4, 7, 19, 5, 8, 11, 3, 15]
   a) The binary search tree (BST) of an array is constructed by inserting elements in the array according to their index position from left to right. Draw the BST of the given array **A** above.
   b) For the BST built above, produce the sequence of nodes visited by preorder and postorder traversals respectively.
   c) The sequence of numbers in an array affects the shape of the constructed BST. One of the shapes resulting in the highest number of comparisons in the worst case when searching is below:



Reorder the elements in the given array **A** to produce another array **B**, such that the BST of **B** would have the above shape.

Extra Practice Questions

7. Suppose we run the following operations (do not actually run it by code, instead do them by hand).

```
>> t = BinaryTree()
>> t.setRoot(Node("A"))
>> t.root.setLeft(Node("B"))
>> t.root.setRight(Node("C"))
>> t.root.left.setLeft(Node("D"))
>> t.root.right.setRight(Node("E"))
>> t.root.right.right.setLeft(Node("F"))
>> t.root.left.setRight(Node("G"))
```

   a) What is the output of calling `t.root.right`?
   b) What is the output of calling `t.root.left.right`?
   c) What is the height of the tree?
   d) What is the parent of "E"?
   e) What is the order of nodes visited for preorder traversal?
   f) What is the order of nodes visited for inorder traversal?
   g) What is the order of nodes visited for postorder traversal?

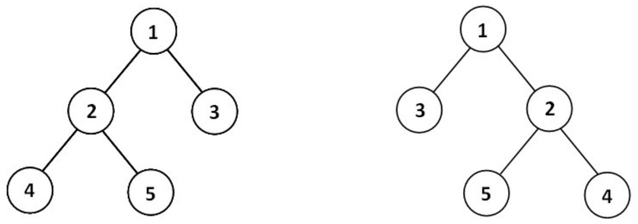
8. If a BST with **n** nodes is not full,
   a) What is the worst case complexity of **searchbst** in terms of **n**?
   b) What is the shape of the tree for this worst case?

9. Bob has a contact list of his colleagues. He finds it tiresome to have to flip through his notebook to find the names. Bob has not taken the Computational Thinking course yet, and is seeking your advice on how to organize his contact list into a BST (based on alphabetical ordering).

   a) What is the best way, i.e., in what sequence, should Bob insert the names of his colleagues into a BST? Explain why.
   b) What is the worst way? Explain why.

10. Andy starts a chain email. He sends an email to two of his friends, with an instruction to either forward it to two other friends (who have never received it), or not to forward it at all. The chain of email forwarding can be represented by a binary tree. If there are 100 people who end up sending an email:
    a) What is the maximum possible height of this binary tree?
    b) What is the minimum possible height of this binary tree?

11. Consider the problem of getting the mirror of a given binary tree. The mirror of a binary tree is another binary tree with the left and right children of all non-leaf nodes interchanged. For example, the following figure shows two binary trees that are mirrors of each other.
    a) Design an algorithm to derive the mirror of a binary tree using recursion.
    b) What is the Big O complexity of this recursive function?



Mirror Binary Trees

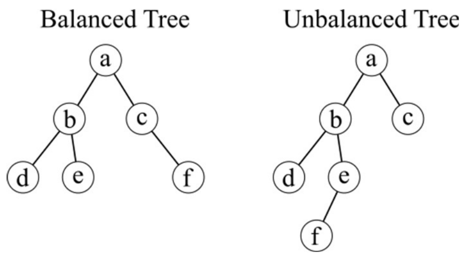
12. The following algorithm depth determines the height of a tree rooted at the input root.

```
def depth(root):
  if root == None:
    return 0
  else:
    return 1 + max(depth(root.left), depth(root.right))
```

a) What is the worst-case complexity of the **depth** algorithm for a binary tree of **N** nodes?

b) Write a recursive algorithm **is_balanced**, which takes in as input the root of a binary tree, and returns **True** if that binary tree is balanced and **False** otherwise. You may assume that the algorithm **depth** above is given, and you are allowed to use it within your proposed recursive algorithm **is_balanced**. State the worst-case complexity of the algorithm **is_balanced** for a binary tree of **N** nodes and clearly show your workings.

A binary tree is balanced if for every node, the depth of the node's left subtree is different from the depth of the node's right subtree by at most one. Refer to the trees at the right: the tree on the left is balanced. The tree on the right is not balanced because for node a, its left subtree has a height of depth(b) = 3 and its right subtree has a height of depth(c) = 1.



Here is another example to explain how to determine if a tree is balanced or not:

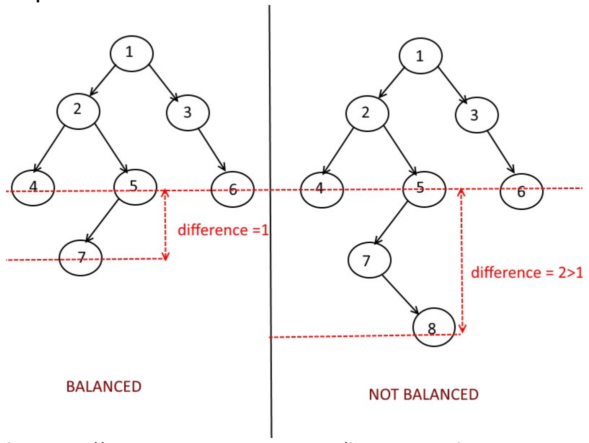


Diagram taken from https://algorithms.tutorialhorizon.com/find-whether-if-a-given-binary-tree-is-balanced

~End