Data Structures and Algorithms in Python

Michael T. Goodrich
Department of Computer Science
University of California, Irvine

Roberto Tamassia
Department of Computer Science
Brown University

Michael H. Goldwasser
Department of Mathematics and Computer Science
Saint Louis University

Study Guide: Hints to Exercises

WILEY

Chapter 8 Trees

Hints

Reinforcement

R-8.1) Be sure not to get confused between depth (which is for a single node) and height (which is for the entire tree).

R-8.2) Recall that the worst-case running time for the depth method is $O(n)$.

R-8.3) Use the definitions of height and depth, and note that the height of the tree has to be realized by some path from the root to an external node. In addition, using induction is the easiest justification technique to use with this proposition.

R-8.4) Use a new parameter, n_p, which refers to the number of positions in the subtree rooted at p.

R-8.5) Review the binary tree ADT.

R-8.6) Consider using dummy nodes.

R-8.7) You need to give four values—the minimum and maximum number of internal nodes and the same for external nodes. In addition, one of these four values is 1.

R-8.8) Use the formulas presented in the book that relate external nodes, internal nodes, and height.

R-8.9) Consider how any proper binary tree with n nodes can be related to a proper binary tree with $n-2$ nodes.

R-8.10) Although this is an abstract class, you may rely on its other (abstract) methods.

R-8.11) Each subtree is rooted at a node of the tree; give the value associated with each such node.

R-8.12) This one is a real puzzler, and it doesn't even use the operators $+$ and $\times$.

R-8.13) Review how arithmetic expressions can be represented with binary trees.

R-8.14) Assume that the container of children is implemented as a positional list.

R-8.15) We'll get you started...

```
class MutableLinkedBinaryTree(LinkedBinaryTree):
  def add_root(self, e):
    return self._add_root(e)
```

R-8.16) Think about a tree that is really a path.

R-8.17) Recall the definition of the level numbering and use this definition when passing information from a node to its children.

R-8.18) All of these methods can be easily performed using formulas involving level numbers.

R-8.19) Use a substitution of $g(p) = f(p) + 1$.

R-8.20) Draw the tree starting with the beginning and end characters of the two character strings.

R-8.21) Draw the tree on a separate sheet of paper and then mark nodes as they are visited, writing down the output produced for each visit.

R-8.22) Draw the tree on a separate sheet of paper and then mark nodes as they are visited, writing down the output produced for each visit.

R-8.23) Draw a tree with one node, one with two nodes, and then one with three nodes.

R-8.24) Draw a tree with one node and then one with three nodes (and note it is impossible to draw a proper binary tree with exactly two nodes).

R-8.25) We already got you started...

R-8.26) You can add all of c's children to the fringe at once.

R-8.27) Draw the tree again and use a pencil and paper to mark how the recursive calls move around the tree.

R-8.28) Argue that this method basically performs the same computations as a familiar tree traversal method.

R-8.29) Consider the information and computations that need to be made in each of the three visits during an Euler tour traversal.

R-8.30) Regular expressions can be helpful for parsing such strings.

Creativity

C-8.31) Use the fact that we can build T from a single root tree via a series of pairs of insertLeft and insertRight operations.

C-8.32) The minimum value will occur when T has one external node.

C-8.33) Consider how you could possibly combine a tree with maximum depth with a tree with minimum depth.

C-8.34) First show that there is a node with three external node children, and then consider what changes when the children of that node are all removed.

C-8.35) Use the definition to derive a recursive algorithm.

C-8.36) Explore the different ways you can have empty nodes on the bottom level.

C-8.37) Explore with pen and paper.

C-8.38) Can you tell how many elements are removed?

C-8.39) There are many special cases to consider when p and q neighbor each other.

C-8.40) Try to avoid conditionals when possible.

C-8.41) Recursion can be very helpful.

C-8.42) Build the new tree in preorder fashion.

C-8.43) The answer to the first part of (c) is ”yes”.

C-8.44) Use a tree traversal.

C-8.45) Derive a formula that relates the depth of a position p to the depths of positions adjacent to p.

C-8.46) Modify an algorithm for computing the depth of each position so that it computes path lengths at the same time.

C-8.47) Try to compute node heights and balance factors at the same time.

C-8.48) Draw a picture of a proper binary tree and its preorder traversal and then hold this picture up to mirror.

C-8.49) First derive a formula for $post(p) - pre(p)$.

C-8.50) Think about what could be the worst-case number of nodes that would have to be traversed to answer each of these queries.

C-8.51) See Section 2.3.4 for discussion of iterators.

C-8.52) Use induction to show that no crossing edges are produced.

C-8.53) Use induction to show that no crossing edges are produced.

C-8.54) The y coordinates can be the same as in the binary tree case. The trick, then, is to find a good substitute for the inorder number used in the binary tree drawing algorithm.

C-8.55) `import os; help(os.walk)`.

C-8.56) Consider a recursive algorithm.

C-8.57) Consider a recursive algorithm.

C-8.58) It helps to know the relative depths of p and q.

C-8.59) Be careful—the path establishing the diameter might not include the root of the tree.

C-8.60) Consider what numbers $f(p)+1$, $f(q)+1$, and $f(a)+1$ look like in binary.

C-8.61) Parameterize your recursion by an index into the list of tokens which begins a subtree.

C-8.62) For the base case, you must determine whether the token is a numeric literal or a string variable.

C-8.63) Use the inherited postorder method.

Projects

P-8.64) What are natural update methods for this representation?

P-8.65) Use a Python list for the children.

P-8.66) You will need to reimplement the _make_position utility.

P-8.67) The essential part of this structure is just a binary tree, so take each part one step at a time.

P-8.68) This is a huge project, so plan accordingly.

P-8.69) Review the definition of this representation to get the details right.

P-8.70) Use recursion where appropriate.