Data Structures and Algorithms in Java™

Sixth Edition

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

R-8.11) This one is a real puzzler, and it doesn't even use the operators $+$ and $*$.

R-8.12) Review how arithmetic expressions can be represented with binary trees.

R-8.13) Assume that the container of children is implemented as a positional list.

R-8.14) Think about a tree that is really a path.

R-8.15) Recall the definition of the level numbering and use this definition when passing information from a node to its children.

R-8.16) All of these methods can be easily performed using formulas involving level numbers.

R-8.17) Use a substitution of $g(p) = f(p) + 1$.

R-8.18) 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-8.19) 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-8.20) Draw a tree with one node, one with two nodes, and then one with three nodes.

R-8.21) 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-8.22) Draw the tree starting with the beginning and end characters of the two character strings.

R-8.23) We already got you started...

R-8.24) Draw the tree again and use a pencil and paper to mark how the recursive calls move around the tree.

R-8.25) The trick is to keep track of how much of the current line you have already used up and whether the next piece of output will exceed the 80 character limit.

R-8.26) Argue that this method basically performs the same computations as a familiar tree traversal method.

Creativity

C-8.27) Define the method recursively.

C-8.28) Modify an algorithm for computing the depth of each position so that it computes path lengths at the same time.

C-8.29) Use the fact that we can build T from a single root tree via a series of pairs of insertLeft and insertRight operations.

C-8.30) The minimum value will occur when T has one external node.

C-8.31) Consider how you could possibly combine a tree with maximum depth with a tree with minimum depth.

C-8.32) 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-8.33) Use the definition to derive a recursive algorithm.

C-8.34) Explore the different ways you might attach external nodes at the bottom level of a tree.

C-8.35) Explore with pen and paper.

C-8.36) Can you tell how many elements are removed?

C-8.37) There are many special cases to consider when p and q neighbor each other.

C-8.38) Try to avoid conditionals when possible.

C-8.39) Recursion can be very helpful.

C-8.40) Build the new tree in preorder fashion.

C-8.41) You may use the technique from either of the previous two problems.

C-8.42) Use a tree traversal.

C-8.43) Derive a formula that relates the depth of a position p to the depths of positions adjacent to p.

C-8.44) Try to compute node heights and balance factors at the same time.

C-8.45) Think about what could be the worst-case number of nodes that would have to be traversed to answer each of these queries.

C-8.46) Use a stack.

C-8.47) The algorithm from Exercise C-8.45 will be useful.

C-8.48) The algorithm from Exercise C-8.45 will be useful.

C-8.49) The algorithm from Exercise C-8.45 will be useful.

C-8.50) Use induction to show that no crossing edges are produced.

C-8.51) Use induction to show that no crossing edges are produced.

C-8.52) The answer to the first part of (c) is ”yes”.

C-8.53) 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-8.54) First derive a formula for $post(p) - pre(p)$.

C-8.55) It helps to know the relative depths of p and q.

C-8.56) Consider what numbers $f(p)+1$, $f(q)+1$, and $f(a)+1$ look like in binary.

C-8.57) Be careful—the path establishing the diameter might not include the root of the tree.

C-8.58) Consider a recursive algorithm.

C-8.59) First convert the infix expression into its equivalent binary tree representation.

C-8.60) Consider a recursive algorithm.

Projects

P-8.61) What are natural update methods for this representation?

P-8.62) Use a Java list for the children.

P-8.63) Review the definition of this representation to get the details right.

P-8.64) You can use an ArrayList of positions to represent the tree path.

P-8.65) Consider building the expression tree using a recursive algorithm, as described in the book.

P-8.66) The essential part of this structure is just a binary tree, so take each part one step at a time.

P-8.67) This is a huge project, so plan accordingly.

P-8.68) Use recursion where appropriate.

P-8.69) You don't have to use the drawing algorithm presented in the book.

P-8.70) You can use a generalization of the binary-tree drawing algorithm presented in the book.

P-8.71) We are considering the tree of all descendants here.

P-8.72) Use the drawing style presented in the book.