8.5 Exercises

Reinforcement

R-8.1 The following questions refer to the tree of Figure 8.3.

a. Which node is the root?
b. What are the internal nodes?
c. How many descendants does node cs016/ have?
d. How many ancestors does node cs016/ have?
e. What are the siblings of node homeworks/?
f. Which nodes are in the subtree rooted at node projects/?
g. What is the depth of node papers/?
h. What is the height of the tree?

R-8.2 Show a tree achieving the worst-case running time for algorithm depth.

R-8.3 Give a justification of Proposition 8.3.

R-8.4 What is the running time of a call to T.height(p) when called on a position p distinct from the root of tree T? (See Code Fragment 8.5.)

R-8.5 Describe an algorithm, relying only on the BinaryTree operations, that counts the number of leaves in a binary tree that are the *left* child of their respective parent.

R-8.6 Let T be an n-node binary tree that may be improper. Describe how to represent T by means of a ***proper*** binary tree T' with $O(n)$ nodes.

R-8.7 What are the minimum and maximum number of internal and external nodes in an improper binary tree with n nodes?

R-8.8 Answer the following questions so as to justify Proposition 8.7.

a. What is the minimum number of external nodes for a proper binary tree with height h? Justify your answer.
b. What is the maximum number of external nodes for a proper binary tree with height h? Justify your answer.
c. Let T be a proper binary tree with height h and n nodes. Show that

$$\log(n+1)-1 \leq h \leq (n-1)/2.$$

d. For which values of n and h can the above lower and upper bounds on h be attained with equality?

R-8.9 Give a proof by induction of Proposition 8.8.

R-8.10 Find the value of the arithmetic expression associated with each subtree of the binary tree of Figure 8.6.

R-8.11 Draw an arithmetic expression tree that has four external nodes, storing the numbers 1, 5, 6, and 7 (with each number stored in a distinct external node, but not necessarily in this order), and has three internal nodes, each storing an operator from the set $\{+,-,*,/\}$, so that the value of the root is 21. The operators may return and act on fractions, and an operator may be used more than once.

R-8.12 Draw the binary tree representation of the following arithmetic expression: "$(((5+2)*(2-1))/((2+9)+((7-2)-1))*8)$".

R-8.13 Justify Table 8.2, summarizing the running time of the methods of a tree represented with a linked structure, by providing, for each method, a description of its implementation, and an analysis of its running time.

R-8.14 Let T be a binary tree with n nodes, and let $f()$ be the level numbering function of the positions of T, as given in Section 8.3.2.

a. Show that, for every position p of T, $f(p) \leq 2^n - 2$.
b. Show an example of a binary tree with seven nodes that attains the above upper bound on $f(p)$ for some position p.

R-8.15 Show how to use an Euler tour traversal to compute the level number $f(p)$, as defined in Section 8.3.2, of each position in a binary tree T.

R-8.16 Let T be a binary tree with n positions that is realized with an array representation A, and let $f()$ be the level numbering function of the positions of T, as given in Section 8.3.2. Give pseudocode descriptions of each of the methods root, parent, left, right, isExternal, and isRoot.

R-8.17 Our definition of the level numbering function $f(p)$, as given in Section 8.3.2, begins with the root having number 0. Some people prefer to use a level numbering $g(p)$ in which the root is assigned number 1, because it simplifies the arithmetic for finding neighboring positions. Redo Exercise R-8.16, but assuming that we use a level numbering $g(p)$ in which the root is assigned number 1.

R-8.18 In what order are positions visited during a preorder traversal of the tree of Figure 8.6?

R-8.19 In what order are positions visited during a postorder traversal of the tree of Figure 8.6?

R-8.20 Let T be an ordered tree with more than one node. Is it possible that the preorder traversal of T visits the nodes in the same order as the postorder traversal of T? If so, give an example; otherwise, explain why this cannot occur. Likewise, is it possible that the preorder traversal of T visits the nodes in the reverse order of the postorder traversal of T? If so, give an example; otherwise, explain why this cannot occur.

R-8.21 Answer the previous question for the case when T is a proper binary tree with more than one node.

R-8.22 Draw a binary tree T that simultaneously satisfies the following:

- Each internal node of T stores a single character.
- A *preorder* traversal of T yields EXAMFUN.
- An *inorder* traversal of T yields MAFXUEN.

R-8.23 Consider the example of a breadth-first traversal given in Figure 8.15. Using the annotated numbers from that figure, describe the contents of the queue before each pass of the while loop in Code Fragment 8.14. To get started, the queue has contents $\{1\}$ before the first pass, and contents $\{2,3,4\}$ before the second pass.

R-8.24 Give the output of the method parenthesize(T, T.root()), as described in Code Fragment 8.26, when T is the tree of Figure 8.6.

R-8.25 Describe a modification to parenthesize, from Code Fragment 8.26, that relies on the length() method for the String class to output the parenthetic representation of a tree with line breaks added to display the tree in a text window that is 80 characters wide.

R-8.26 What is the running time of parenthesize(T, T.root()), as given in Code Fragment 8.26, for a tree T with n nodes?

Creativity

C-8.27 Describe an efficient algorithm for converting a fully balanced string of parentheses into an equivalent tree. The tree associated with such a string is defined recursively. The outermost pair of balanced parentheses is associated with the root and each substring inside this pair, defined by the substring between two balanced parentheses, is associated with a subtree of this root.

C-8.28 The ***path length*** of a tree T is the sum of the depths of all positions in T. Describe a linear-time method for computing the path length of a tree T.

C-8.29 Define the ***internal path length***, $I(T)$, of a tree T to be the sum of the depths of all the internal positions in T. Likewise, define the ***external path length***, $E(T)$, of a tree T to be the sum of the depths of all the external positions in T. Show that if T is a proper binary tree with n positions, then $E(T) = I(T) + n - 1$.

C-8.30 Let T be a (not necessarily proper) binary tree with n nodes, and let D be the sum of the depths of all the external nodes of T. Show that if T has the minimum number of external nodes possible, then D is $O(n)$ and if T has the maximum number of external nodes possible, then D is $O(n \log n)$.

C-8.31 Let T be a (possibly improper) binary tree with n nodes, and let D be the sum of the depths of all the external nodes of T. Describe a configuration for T such that D is $\Omega(n^2)$. Such a tree would be the worst case for the asymptotic running time of method heightBad (Code Fragment 8.4).

C-8.32 For a tree T, let n_I denote the number of its internal nodes, and let n_E denote the number of its external nodes. Show that if every internal node in T has exactly 3 children, then $n_E = 2n_I + 1$.

C-8.33 Two ordered trees T' and T'' are said to be ***isomorphic*** if one of the following holds:

- Both T' and T'' are empty.
- Both T' and T'' consist of a single node
- The roots of T' and T'' have the same number $k \geq 1$ of subtrees, and the i^{th} such subtree of T' is isomorphic to the i^{th} such subtree of T'' for $i = 1, \ldots, k$.

Design an algorithm that tests whether two given ordered trees are isomorphic. What is the running time of your algorithm?

C-8.34 Show that there are more than 2^n improper binary trees with n internal nodes such that no pair are isomorphic (see Exercise C-8.33).

C-8.35 If we exclude isomorphic trees (see Exercise C-8.33), exactly how many proper binary trees exist with exactly 4 leaves?

C-8.36 Add support in LinkedBinaryTree for a method, pruneSubtree(p), that removes the entire subtree rooted at position p, making sure to maintain an accurate count of the size of the tree. What is the running time of your implementation?

C-8.37 Add support in LinkedBinaryTree for a method, swap(p, q), that has the effect of restructuring the tree so that the node referenced by p takes the place of the node referenced by q, and vice versa. Make sure to properly handle the case when the nodes are adjacent.

C-8.38 We can simplify parts of our LinkedBinaryTree implementation if we make use of of a single sentinel node, such that the sentinel is the parent of the real root of the tree, and the root is referenced as the left child of the sentinel. Furthermore, the sentinel will take the place of null as the value of the left or right member for a node without such a child. Give a new implementation of the update methods remove and attach, assuming such a representation.

C-8.39 Describe how to clone a LinkedBinaryTree instance representing a proper binary tree, with use of the attach method.

C-8.40 Describe how to clone a LinkedBinaryTree instance representing a (not necessarily proper) binary tree, with use of the addLeft and addRight methods.

C-8.41 Modify the LinkedBinaryTree class to formally support the Cloneable interface, as described in Section 3.6.

C-8.42 Give an efficient algorithm that computes and prints, for every position p of a tree T, the element of p followed by the height of p's subtree.

C-8.43 Give an $O(n)$-time algorithm for computing the depths of all positions of a tree T, where n is the number of nodes of T.

C-8.44 The ***balance factor*** of an internal position p of a proper binary tree is the difference between the heights of the right and left subtrees of p. Show how to specialize the Euler tour traversal of Section 8.4.6 to print the balance factors of all the internal nodes of a proper binary tree.

C-8.45 Design algorithms for the following operations for a binary tree T:

- preorderNext(p): Return the position visited after p in a preorder traversal of T (or null if p is the last node visited).
- inorderNext(p): Return the position visited after p in an inorder traversal of T (or null if p is the last node visited).
- postorderNext(p): Return the position visited after p in a postorder traversal of T (or null if p is the last node visited).

What are the worst-case running times of your algorithms?

C-8.46 Describe, in pseudocode, a nonrecursive method for performing an inorder traversal of a binary tree in linear time.

C-8.47 To implement the preorder method of the AbstractTree class, we relied on the convenience of creating a snapshot. Reimplement a preorder method that creates a ***lazy iterator***. (See Section 7.4.2 for discussion of iterators.)

C-8.48 Repeat Exercise C-8.47, implementing the postorder method of the AbstractTree class.

C-8.49 Repeat Exercise C-8.47, implementing the AbstractBinaryTree's inorder method.

C-8.50 Algorithm preorderDraw draws a binary tree T by assigning x- and y-coordinates to each position p such that $x(p)$ is the number of nodes preceding p in the preorder traversal of T and $y(p)$ is the depth of p in T.

a. Show that the drawing of T produced by preorderDraw has no pairs of crossing edges.
b. Redraw the binary tree of Figure 8.19 using preorderDraw.

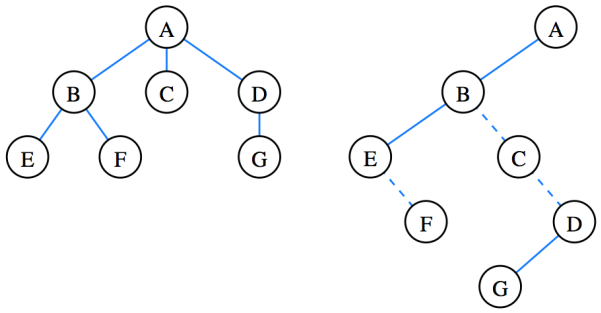
C-8.51 Redo the previous problem for the algorithm postorderDraw that is similar to preorderDraw except that it assigns $x(p)$ to be the number of nodes preceding position p in the postorder traversal.

C-8.52 We can define a ***binary tree representation*** T' for an ordered general tree T as follows (see Figure 8.21):

- For each position p of T, there is an associated position p' of T'.
- If p is a leaf of T, then p' in T' does not have a left child; otherwise the left child of p' is q', where q is the first child of p in T.
- If p has a sibling q ordered immediately after it in T, then q' is the right child of p' in T; otherwise p' does not have a right child.

Given such a representation T' of a general ordered tree T, answer each of the following questions:

a. Is a preorder traversal of T' equivalent to a preorder traversal of T?
b. Is a postorder traversal of T' equivalent to a postorder traversal of T?
c. Is an inorder traversal of T' equivalent to one of the standard traversals of T? If so, which one?



(a) (b)

Figure 8.21: Representation of a tree with a binary tree: (a) tree T; (b) binary tree T' for T. The dashed edges connect nodes of T' that are siblings in T.

C-8.53 Design an algorithm for drawing *general* trees, using a style similar to the inorder traversal approach for drawing binary trees.

C-8.54 Let the ***rank*** of a position p during a traversal be defined such that the first element visited has rank 1, the second element visited has rank 2, and so on. For each position p in a tree T, let $\mathsf{pre}(p)$ be the rank of p in a preorder traversal of T, let $\mathsf{post}(p)$ be the rank of p in a postorder traversal of T, let $\mathsf{depth}(p)$ be the depth of p, and let $\mathsf{desc}(p)$ be the number of descendants of p, including p itself. Derive a formula defining $\mathsf{post}(p)$ in terms of $\mathsf{desc}(p)$, $\mathsf{depth}(p)$, and $\mathsf{pre}(p)$, for each node p in T.

C-8.55 Let T be a tree with n positions. Define the ***lowest common ancestor*** (LCA) between two positions p and q as the lowest position in T that has both p and q as descendants (where we allow a position to be a descendant of itself). Given two positions p and q, describe an efficient algorithm for finding the LCA of p and q. What is the running time of your algorithm?

C-8.56 Suppose each position p of a binary tree T is labeled with its value $f(p)$ in a level numbering of T. Design a fast method for determining $f(a)$ for the lowest common ancestor (LCA), a, of two positions p and q in T, given $f(p)$ and $f(q)$. You do not need to find position a, just value $f(a)$.

C-8.57 Let T be a binary tree with n positions, and, for any position p in T, let d_p denote the depth of p in T. The ***distance*** between two positions p and q in T is $d_p + d_q - 2d_a$, where a is the lowest common ancestor (LCA) of p and q. The ***diameter*** of T is the maximum distance between two positions in T. Describe an efficient algorithm for finding the diameter of T. What is the running time of your algorithm?

C-8.58 The ***indented parenthetic representation*** of a tree T is a variation of the parenthetic representation of T (see Code Fragment 8.26) that uses indentation and line breaks as illustrated in Figure 8.22. Give an algorithm that prints this representation of a tree.

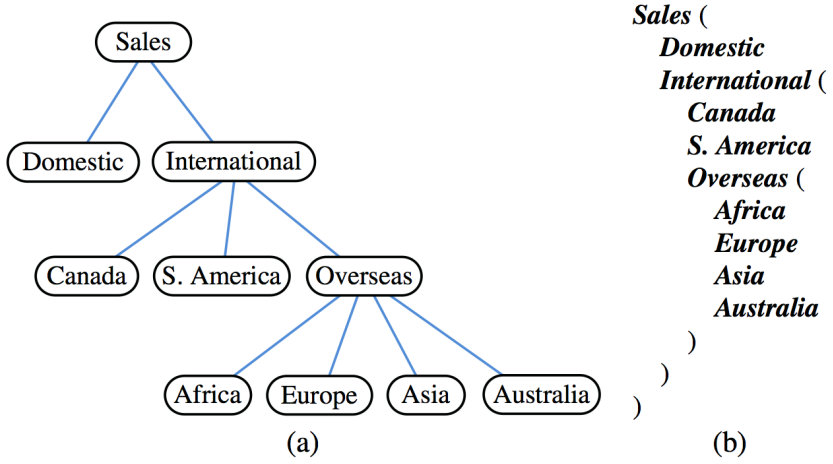


Figure 8.22: (a) Tree T; (b) indented parenthetic representation of T.

C-8.59 As mentioned in Exercise C-6.19, ***postfix notation*** is an unambiguous way of writing an arithmetic expression without parentheses. It is defined so that if "(exp_1) **op** (exp_2)" is a normal (infix) fully parenthesized expression with operation **op**, then its postfix equivalent is "$pexp_1$ $pexp_2$ **op**", where $pexp_1$ is the postfix version of exp_1 and $pexp_2$ is the postfix version of exp_2. The postfix version of a single number or variable is just that number or variable. So, for example, the postfix version of the infix expression "$((5+2)*(8-3))/4$" is "5 2 + 8 3 − ∗ 4 /". Give an efficient algorithm for converting an infix arithmetic expression to its equivalent postfix notation. (Hint: First convert the infix expression into its equivalent binary tree representation.)

C-8.60 Let T be a binary tree with n positions. Define a ***Roman position*** to be a position p in T, such that the number of descendants in p's left subtree differ from the number of descendants in p's right subtree by at most 5. Describe a linear-time method for finding each position p of T, such that p is not a Roman position, but all of p's descendants are Roman.

Projects

P-8.61 Implement the binary tree ADT using the array-based representation described in Section 8.3.2.

P-8.62 Implement the tree ADT using a linked structure as described in Section 8.3.3. Provide a reasonable set of update methods for your tree.

P-8.63 Implement the tree ADT using the binary tree representation described in Exercise C-8.52. You may adapt the LinkedBinaryTree implementation.

P-8.64 The memory usage for the LinkedBinaryTree class can be streamlined by removing the parent reference from each node, and instead implementing a Position as an object that keeps a list of nodes representing the entire path from the root to that position. Reimplement the LinkedBinaryTree class using this strategy.

P-8.65 Write a program that takes as input a fully parenthesized, arithmetic expression and converts it to a binary expression tree. Your program should display the tree in some way and also print the value associated with the root. For an additional challenge, allow the leaves to store variables of the form x_1, x_2, x_3, and so on, which are initially 0 and which can be updated interactively by your program, with the corresponding update in the printed value of the root of the expression tree.

P-8.66 A ***slicing floor plan*** divides a rectangle with horizontal and vertical sides using horizontal and vertical ***cuts***. (See Figure 8.23a.) A slicing floor plan can be represented by a proper binary tree, called a ***slicing tree***, whose internal nodes represent the cuts, and whose external nodes represent the ***basic rectangles*** into which the floor plan is decomposed by the cuts. (See Figure 8.23b.) The ***compaction problem*** for a slicing floor plan is defined as follows. Assume that each basic rectangle of a slicing floor plan is assigned a minimum width w and a minimum height h. The compaction problem is to find the smallest possible height

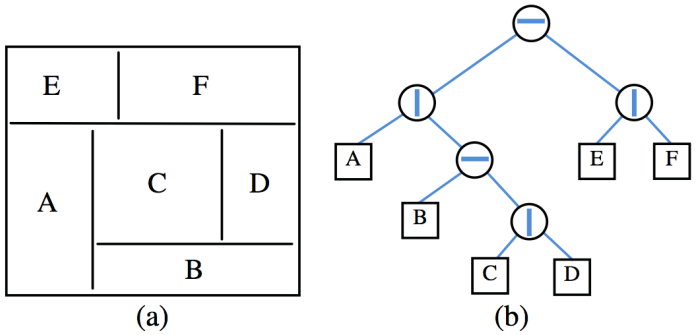


Figure 8.23: (a) Slicing floor plan; (b) slicing tree associated with the floor plan.

and width for each rectangle of the slicing floor plan that is compatible with the minimum dimensions of the basic rectangles. Namely, this problem requires the assignment of values $\mathsf{h}(p)$ and $\mathsf{w}(p)$ to each position p of the slicing tree such that:

$$\mathsf{w}(p) = \begin{cases} w & \text{if } p \text{ is a leaf whose basic rectangle has minimum width } w \\ \max(\mathsf{w}(\ell), \mathsf{w}(r)) & \text{if } p \text{ is an internal position, associated with a horizontal cut, with left child } \ell \text{ and right child } r \\ \mathsf{w}(\ell) + \mathsf{w}(r) & \text{if } p \text{ is an internal position, associated with a vertical cut, with left child } \ell \text{ and right child } r \end{cases}$$

$$\mathsf{h}(p) = \begin{cases} h & \text{if } p \text{ is a leaf node whose basic rectangle has minimum height } h \\ \mathsf{h}(\ell) + \mathsf{h}(r) & \text{if } p \text{ is an internal position, associated with a horizontal cut, with left child } \ell \text{ and right child } r \\ \max(\mathsf{h}(\ell), \mathsf{h}(r)) & \text{if } p \text{ is an internal position, associated with a vertical cut, with left child } \ell \text{ and right child } r \end{cases}$$

Design a data structure for slicing floor plans that supports the operations:

- Create a floor plan consisting of a single basic rectangle.
- Decompose a basic rectangle by means of a horizontal cut.
- Decompose a basic rectangle by means of a vertical cut.
- Assign minimum height and width to a basic rectangle.
- Draw the slicing tree associated with the floor plan.
- Compact and draw the floor plan.

P-8.67 Write a program that can play Tic-Tac-Toe effectively. (See Section 3.1.5.) To do this, you will need to create a ***game tree*** T, which is a tree where each position corresponds to a ***game configuration***, which, in this case, is a representation of the Tic-Tac-Toe board. (See Section 8.4.2.) The root corresponds to the initial configuration. For each internal position p in T, the children of p correspond to the game states we can reach from p's game state in a single legal move for the appropriate player, A (the first player) or B (the second player). Positions at even depths correspond to moves for A and positions at odd depths correspond to moves for B. Leaves are either final game states or are at a depth beyond which we do not want to explore. We score each leaf with a value that indicates how good this state is for player A. In large games, like chess, we have to use a heuristic scoring function, but for small games, like Tic-Tac-Toe, we can construct the entire game tree and score leaves as $+1$, 0, -1, indicating whether player A has a win, draw, or lose in that configuration. A good algorithm for choosing moves is ***minimax***. In this algorithm, we assign a score to each internal position p in T, such that if p represents A's turn, we compute p's score as the maximum of the scores of p's children (which corresponds to A's optimal play from p). If an internal node p represents B's turn, then we compute p's score as the minimum of the scores of p's children (which corresponds to B's optimal play from p).

P-8.68 Write a program that takes as input a general tree T and a position p of T and converts T to another tree with the same set of position adjacencies, but now with p as its root.

P-8.69 Write a program that draws a binary tree.

P-8.70 Write a program that draws a general tree.

P-8.71 Write a program that can input and display a person's family tree.

P-8.72 Write a program that visualizes an Euler tour traversal of a proper binary tree, including the movements from node to node and the actions associated with visits on the left, from below, and on the right. Illustrate your program by having it compute and display preorder labels, inorder labels, postorder labels, ancestor counts, and descendant counts for each node in the tree (not necessarily all at the same time).

Chapter Notes

Discussions of the classic preorder, inorder, and postorder tree traversal methods can be found in Knuth's *Fundamental Algorithms* book [60]. The Euler tour traversal technique comes from the parallel algorithms community; it is introduced by Tarjan and Vishkin [86] and is discussed by JáJá [50] and by Karp and Ramachandran [55]. The algorithm for drawing a tree is generally considered to be a part of the "folklore" of graph-drawing algorithms. The reader interested in graph drawing is referred to the book by Di Battista, Eades, Tamassia, and Tollis [29] and the survey by Tamassia and Liotta [85]. The puzzle in Exercise R-8.11 was communicated by Micha Sharir.