Section 5.2: Trees and their representations

March 2, 2005

Abstract

Trees are defined, some applications are presented, some computer representation strategies are considered, and tree traversal algorithms are discussed. Trees are file cabinets: good ways to store stuff. Once it's stored, however, we'll need to retrieve it; hence we must be able to efficiently traverse the tree checking what's in each file!

1 Tree Definition and Terminology

Definition: a **tree** is an acyclic, connected graph with one node designated as the **root** node. (Note: NIST (the National Institute of Standards and Technology) defines a tree as **undirected**!)

"Because a tree is a connected graph, there is a path from the root to any other node in the tree; because the tree is acyclic, that path is unique [provided that no arc is used twice]." p. 371, Gersting.

The set of trees can also be defined recursively, as follows:

- **Base case:** A single node is a tree, with that node as root.
- **Inductive step:** The graph formed by attaching a new node r by a single arc to each root of trees $\{T_i\}$ is a tree.

Example: Create a tree on a 4x6 card. Some of you should make rather ordered trees; others might think of very strange trees.

Use this tree terminology handout to classify your tree.

Exercise #1, p. 381.

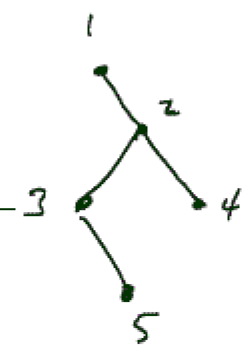
Exercise #2, p. 381.

2 Examples of trees in action (p. 372)

- My favorite: the UNIX operating system (with root node called root!)
- The Hexstat probability demonstrator is based on what looks like a loose extension of a full binary tree, what we might call a **directed acyclic graph**; and on the binomial theorem.
- Family "trees" (if intermarriages are forbidden!)
- Org charts (most people report to only one person above them - what happens if not? Remember Prolog?)

3 Tree Representations

Binary trees are special in that their children are classified as left to right. So in order to represent them, we need to specify left to right children for each node. Two representations are shown in Example 23, p. 375: a two-column array, and an adjacency list with three-field records.



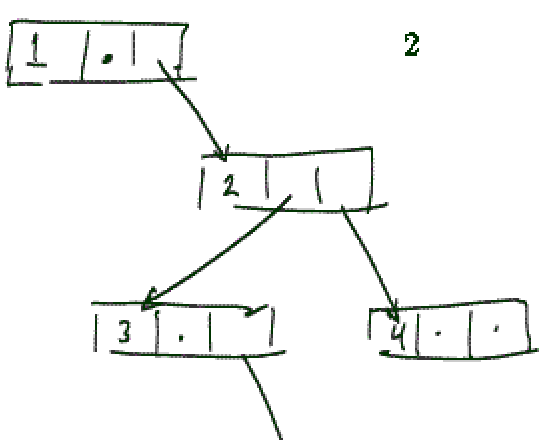
Practice 20, p. 375.

4 Tree Traversal Algorithms

We'll look at three tree traversal algorithms, which are represented recursively as follows:

preorder	*root*	*left*	*right*
inorder	*left*	*root*	*right*
postorder	*left*	*right*	*root*

	left	right
1	0	2
2	3	4
3	0	5
4	0	0
5	0	0



5.1.

Check out this animation showing how a tree is traversed by the three algorithms. It is clear from this animation how one handles non-binary tree traversal for pre- and post-order; for inorder, our book's algorithm would do "left, root, right, righter, ..., rightest"!

Note the nice definitions of these traversal algorithms based on the recursive definition of a tree.

Exercise #18, p. 384.

Each traversal method has advantages in different situations. The example we consider is that of binary trees as representations of arithmetic expressions (e.g. Figure 5.40, p. 374).

- Inorder traversal writes the expressions in the most familiar form from school (Infix notation). Inorder traversal requires the introduction of parentheses to make the meaning of the expression unique.

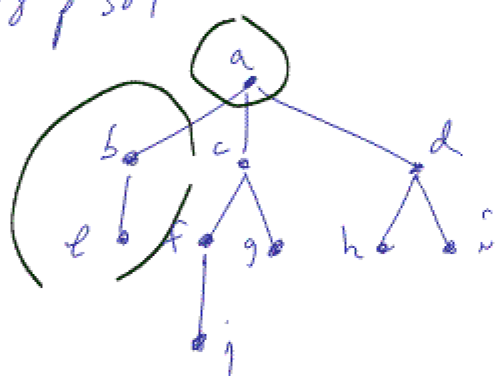
- Postorder traversal allows us to eliminate nodes once traversed (if we want to deallocate storage, for example - check, in the animation above, that you can deallocate each node as soon as it's been visited!). This is called Reverse Polish Notation, named for the Pole Jan Lukasiewicz.

  "Why Did/Does HP Use RPN?

  In the years that followed, computer scientists realized that RPN or postfix notation was very efficient for computer math. As a postfix expression is scanned from left to right, operands are simply placed into a last-in, first-out (LIFO) stack and operators may be immediately applied to the operands at the bottom of the stack. By contrast, expressions with parentheses and precedence (infix notation) require that operators be delayed until some later point. Thus, the compilers on on almost all modern computers converted statements to RPN for execution. (In fact, some computer manufacturers designed their computers around postfix notation.)" (from the link above, on http://www.hpmuseum.org/rpn.htm)

- Preorder traversal represents the expressions in Polish notation (which is what LISP uses (remember, LISP stands for "God's Computer Language")). Like postorder, preorder doesn't require parentheses (pro-

#18 p 384



preorder
a
b
e
c
f
j
g

inorder
e
b
a
j
f
c
a
g

postorder
e
b
j
f
g
c

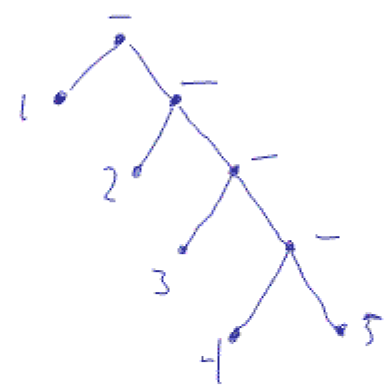
d h i   h d i   h i d a

vided the operators are binary). If not all operations are binary, then parantheses creep in, as LISP lovers (and haters!) know only too well....

Exercise #5, p. 381 (and write the expression in prefix and postfix notation).

or #28, p. 385

#5



$1-(2-[3-(4-5)])$

prefix: (- 1 (- 2 (- 3 (- 4 5)))).

postfix: 1 2 3 4 5 - - - -

#28 p385

preorder: a, b, c, d, e

inorder: b, a, d, c, e