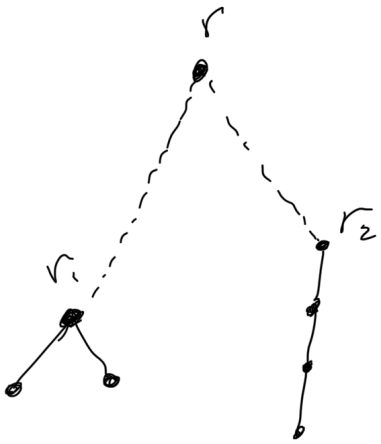
Section 6.2: Trees and their representations

March 8, 2021

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 node.



1 Tree Definition and Terminology

Definition: a **tree** is an acyclic, connected graph with one node designated as the **root** node. Note that trees are usually considered undirected, even though things tend to descend from the root....

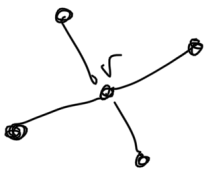
"Because a tree is a connected graph, there is a path from the root to any other node in the tree; because the tree is acyclic, that path is unique [provided that no arc is used twice]." p. 510, Gersting.

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 n trees $\{T_i\}_{i\in\{1,\ldots,n\}}$ is a tree.

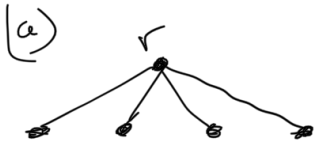
Example: create a tree, subject to the definition. Some of you should make rather ordered trees; others might think of very strange trees.

Use this tree terminology handout to classify your tree.

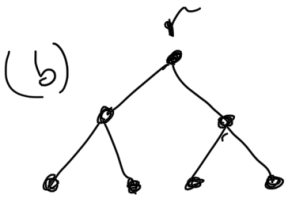


Consider Exercise #3, p. 521. Sketch a picture of each of the following trees:

(a) Tree with five nodes and depth 1.

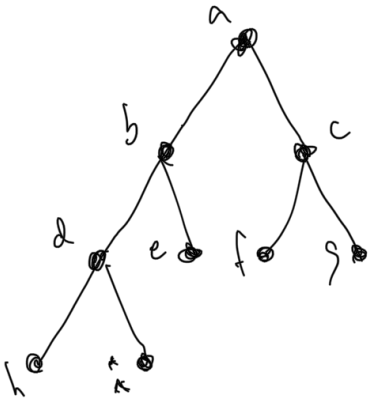


(b) Full binary tree of depth 2.



(c) Tree of depth 3 where each node at depth i has $i+1$ children.

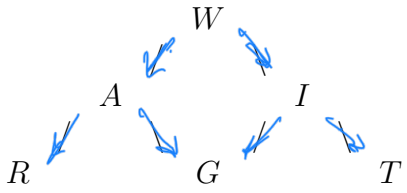
Exercise #4, p. 522.



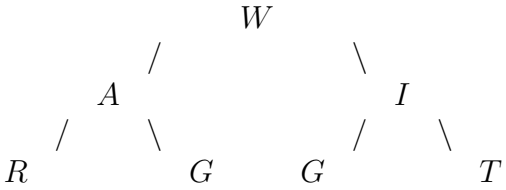
a. Is it binary? Y
b. Is it a full binary? N
c. Is it a complete binary? Y
d. What is the parent of e? b
e. What is the right child of e? NA
f. What is the depth of g? 2
g. What is the height of the tree? 3

2 Examples of trees in action (p. 511)

- My favorite: the UNIX operating system (with root node called root!)
- Trees and tree-like graphs appear in funny places: I was listening to NPR (3/13/05) and Will Short, the Puzzle Guy, gave this puzzle:



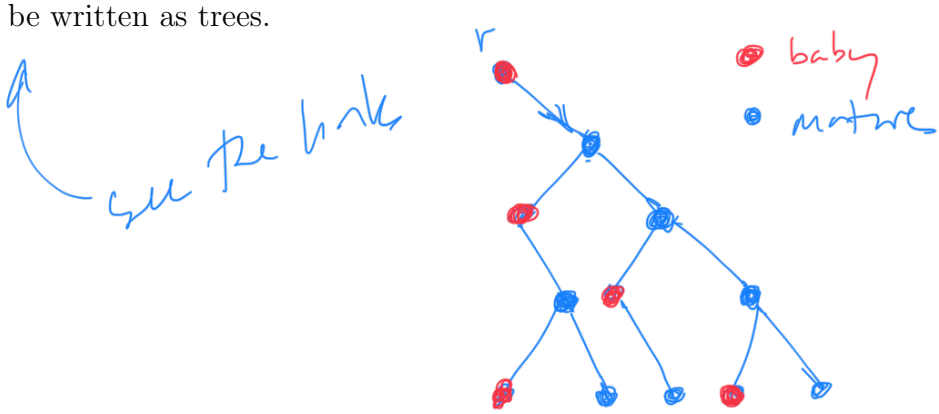
This is a graph, but not a tree: it is connected, directed, and acyclic. Let's turn it into a tree, by duplicating some nodes:



In traversing this graph from root to leaf (more about this in a moment), in four possible ways, every way spells out a word: WAR, WAG, WIG, WIT; he asked us to create a full binary tree of depth 3 that uses the ten letters in ANALOGIES+K and does the same thing.... (as a hint, he said that an A is the left-most leaf, only not in those words – he used a lot more!:).

One of my students took the challenge of solving this (by brute force – oh well!;): Josh Werrmann

- Family "trees" (if intermarriages are forbidden!)
- Fibonacci numbers can be written as trees.



- Org charts (most people report to only one person above them; have only one "parent", as all good trees should)

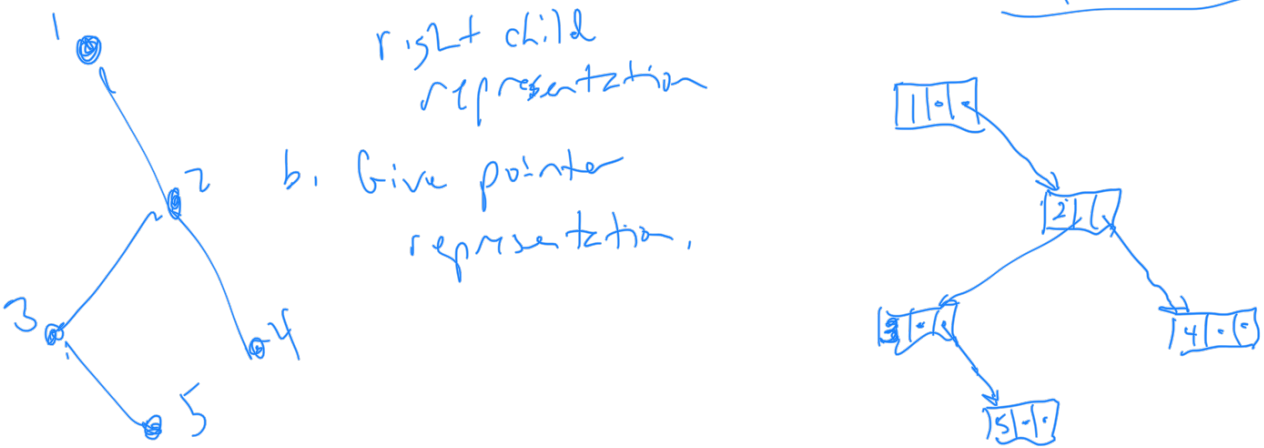
3 Tree Representations

Since trees are graphs, we can use the usual graph strategies for representation (e.g. matrices, adjacency lists).

Binary trees are special in that their children are classified as left and right. So in order to represent them, we need to specify left to right children for each node. Two representations are shown in Example 24, p. 513: a two-column array, and an adjacency list with three-field records. Notice how the adjacency list (pointer representation) is arranged to reproduce the geometry of the tree.

	l	r
1	0	2
2	3	4
3	0	5
4	0	0
5	0	0

Practice 20, p. 513.



Notice that these representations (and the notion of left and right children) are **enhancements** of an ordinary graph: if we change the shape of the graph (without cutting arcs, adding nodes, etc.), then we should be looking at the same graph. But for a binary tree with right and left children, reflecting the graph would change the meaning (and the representation).

Trees are also different from graphs in that there is a special node – the root node.

4 Tree Traversal Algorithms

We'll look at three tree traversal algorithms, which are defined recursively – ah, recursion! – as follows (see p. 514):

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

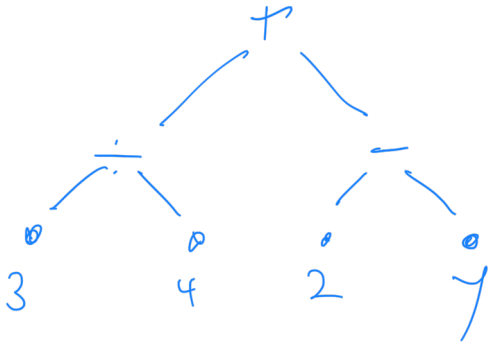
Check out this animation showing how a tree is traversed by the three algorithms.

It is clear from this animation how one handles non-binary tree traversal for pre- and postorder; 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. For example, our author's definition of algorithm preorder:

```
Preorder(tree T)
// Writes the nodes of a tree with root r in preorder
  write(r)
  for i=1 to t do
    Preorder(T_i)
  end for
end Preorder
```

Exercise #21, p. 525 – traverse using each procedure

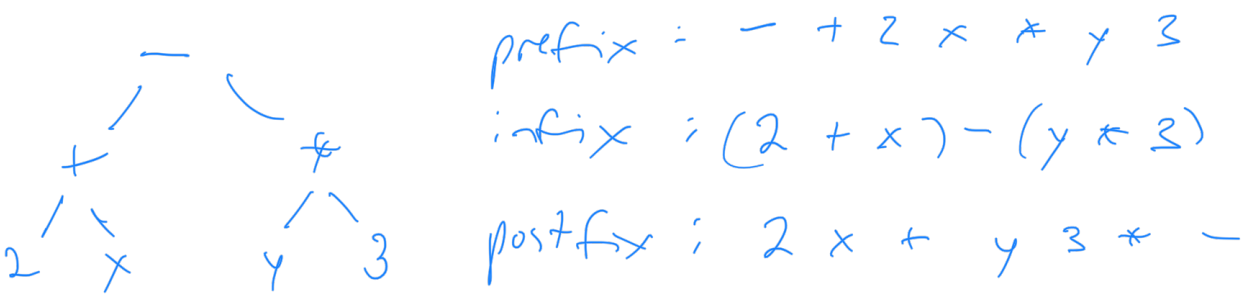


pre-order : + ÷ 3 4 − 2 y

in-order : (3 ÷ 4) + (2 − y)

post-order : 3 4 ÷ 2 y − +

Each traversal method has advantages in different situations. The example we consider is that of binary trees as representations of arithmetic expressions: Example 23 and Figure 6.40, p. 512.



- Inorder traversal writes the expressions in the most familiar form from our early schooldays (**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, in honor of 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 almost all modern computers converted statements to RPN for execution. (In fact, some computer manufacturers designed their computers around postfix notation.)"[1]

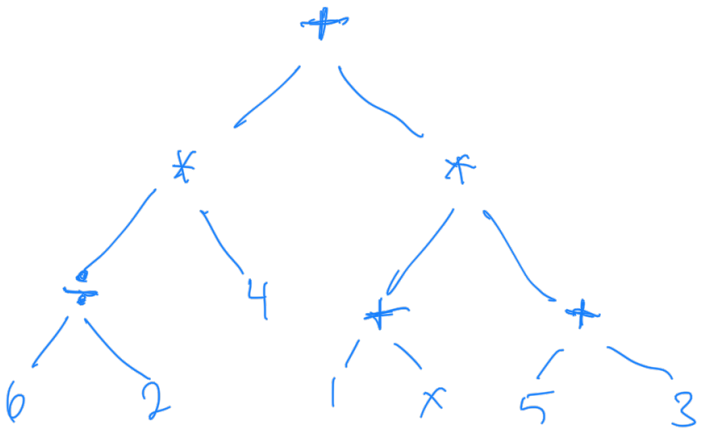
- Preorder traversal represents the expressions in Polish notation (which is what LISP uses (my favorite language)). Like postorder, preorder doesn't require parentheses (provided the operators are binary). If not all operations are binary, then parentheses creep in, as LISP lovers (and haters!) know only too well....

[1] From the link above, on `http://www.hpmuseum.org/rpn.htm`

Exercise #8, p. 522: draw the expression tree of

$$[(6 \div 2) * 4] + [(1 + x) * (5 + 3)]$$

(and write the expression in prefix and postfix notation).

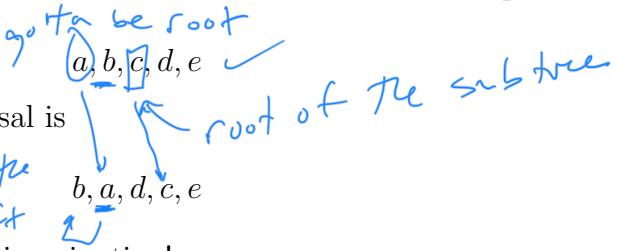


prefix: + * ÷ 6 2 4 * + 1 x + 5 3

postfix:

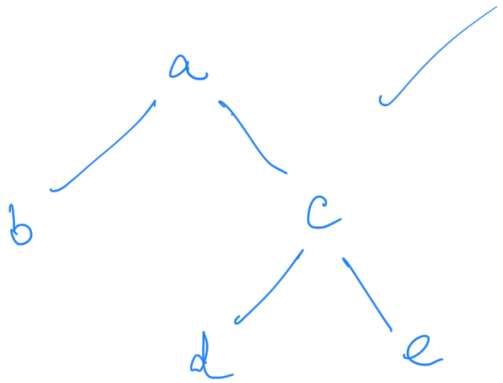
6 2 ÷ 4 * 1 x + 5 3 + * +

Exercise #33, p. 525, is a little harder: Draw a tree whose preorder traversal is



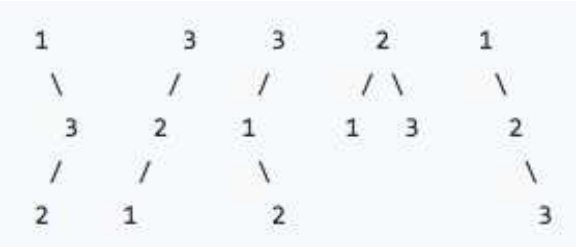
and whose inorder traversal is

This one requires a little imagination!



5 Climbing some trees...

Problem 1: Here's a really interesting problem, that connects trees to recursion, and the Catalan numbers in particular: Given $n \in \mathbb{N}$, find $s(n)$ – the number of structurally unique binary search trees that store values 1 through n. Find the recursion relation for $s(n)$, and its first 8 values. (For convenience let's define $s(0) = 1$.) This figure shows that $s(3) = 5$:



Problem 2: To test blood from blood donors for coronavirus, small samples of blood from $n = 2^m$ ($m \in \mathbb{N}$) donors are pooled, and then **the pooled sample is tested**. If negative, great – all clear, after just one test; if positive, however, apply the strategy recursively on pooled samples of two halves, one after the other.

(a) **Best case scenario**: Suppose we know that **only one person** is infected (**and the lab knows it, too**): how many Covid-19 tests will the lab do in order to identify the individual? (You might consider some simple cases.) How does that compare to just testing everyone?

(b) **Worst case scenario**: Suppose we know that **all** are infected (**but the lab doesn't know it**). How many tests will the lab do in order to determine that? How does that compare to just testing everyone?

(c) **Reality**: is generally somewhere between best and worst cases. At what prevalence (infection rate) will sequential testing of all donors require about the same number of tests as this divide-and-conquer strategy?