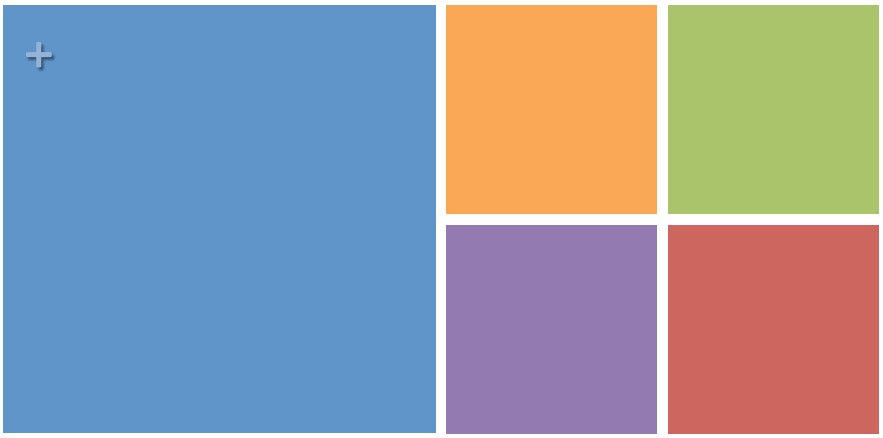


Lecture 16: Binary Trees
Implementation and Traversal

Today

- Reading
  - JS Ch. 12.6-12.10 (Trees)
- Objectives
  - Weekly assignment
  - Implementing a binary tree
  - Tree traversals

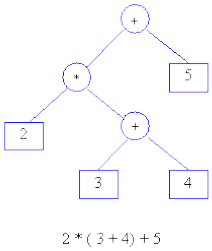
Announcements

- Just a reminder
  - Start assignments earlier (Monday)!
  - Thursday mentor hours are much emptier (hint, hint)
  - Come to office hours (or make an appointment)
  - Ask questions in class
  - Ask questions on Piazza
  - Read lab writeups before coming to lab
  - Don't spend valuable time finishing lab – get to assignment
- Everyone should have gotten Assignment 2 grades back

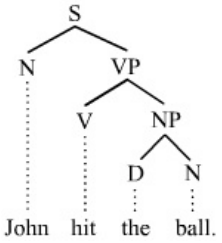
More on the Weekly Assignment

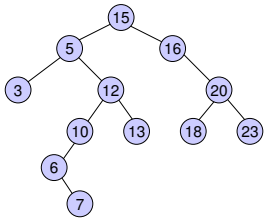
- Calculator
  - Updated the programming assignment writeup
  - Calculator is only responsible for integer precision, e.g. "3 2 /" should result in "1" being displayed

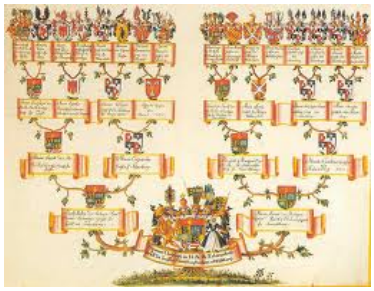
Recap: Trees abound!



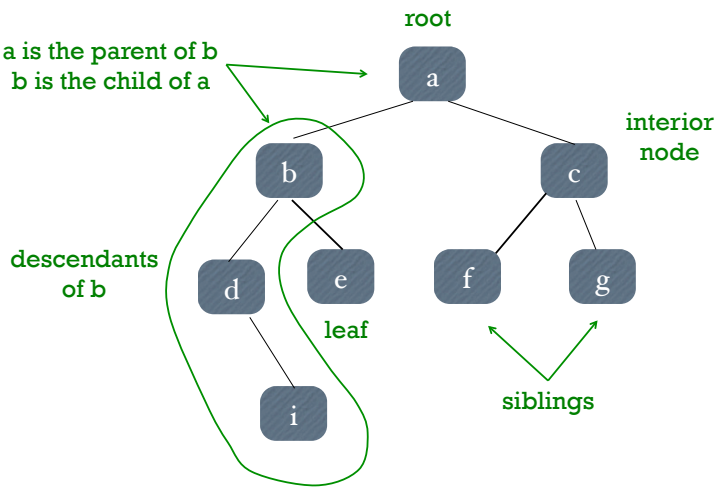
2 * (3 + 4) + 5







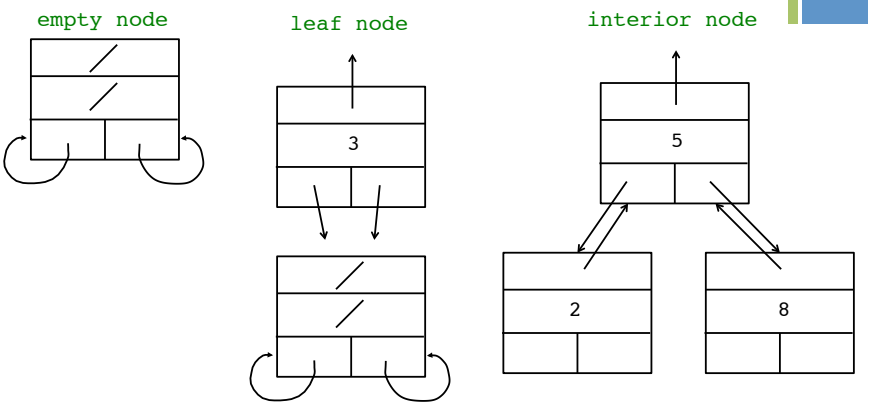
Recap: Tree Terminology



Recap: Binary Tree Theorems

- (Theorem 1) Let T be a binary tree. For every $k \geq 0$, there are at most 2^k nodes at level (depth) k
- (Theorem 2) Let T be a binary tree with height h. Then T has at most $2^{h+1}-1$ nodes.
- (Theorem 3) Let T be a binary tree with N nodes. Then the height h of T is at least $\lceil \log(N+1) \rceil - 1$
- A binary tree T is *balanced* if it has the minimum possible height for its number of nodes.

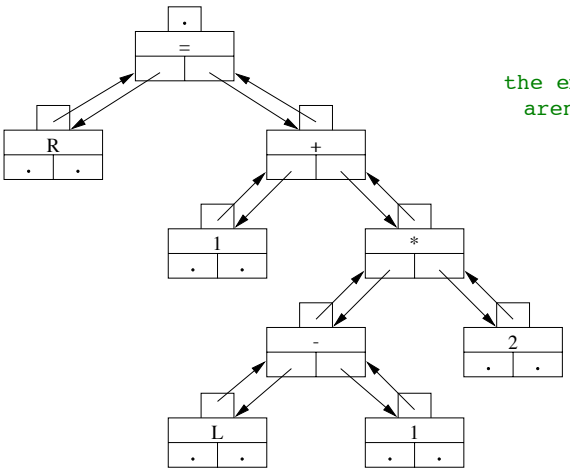
Bailey BinaryTree Implementation



Bailey BinaryTree Implementation

- `public BinaryTree<E> left()`
- `public BinaryTree<E> right()`
- `public void setRight(BinaryTree<E> new Right)`
- `public E value()`
- `public boolean isEmpty() // tests if value is null`
- `depth(), height(), size(),...`

Bailey BinaryTree Implementation



the empty nodes aren't shown

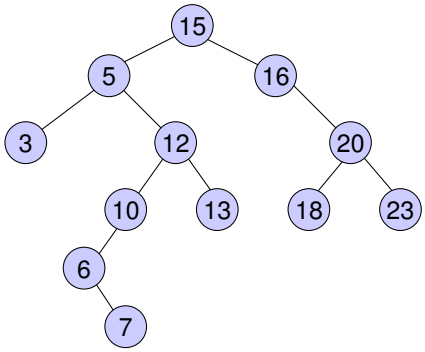
Tree Traversals

- Unlike lists, there is no standard order of traversal for trees
- A *tree traversal* is a specific order in which to traverse a tree structure
- Pre-order Traversal
  - root, left subtree, right subtree
- In-order Traversal
  - left subtree, node, right subtree
- Post-order Traversal
  - left subtree, right subtree, root

recursive or stack-based implementation

Tree Traversals

- List the order in which the nodes are visited



- Pre-order Traversal
  - root, left subtree, right subtree
- In-order Traversal
  - left subtree, node, right subtree
- Post-order Traversal
  - left subtree, right subtree, root

Tree Traversals

- Write a recursive method to print out the values in a `BinaryTree` using a pre-order traversal

```
public void preOrderTraversal(BinaryTree<E> node)
{









}
```

Evaluate Expression Tree

- Use post-order traversal to evaluate expression