Lecture 17: Binary Trees

CS 62

Fall 2017

Kim Bruce & Alexandra Papoutsaki
A tree is either:
- Empty or
- consists of a node, called the root node, together with a collection of trees, called its subtrees. These trees are disjoint from each other and the root.
Definitions

- An **edge** connects a node to its subtrees.
- The roots of the subtrees of a node are said to be the **successors** or **descendants** of the node.
- Nodes without successors are called **leaves**. The others are called **interior nodes**.
- All nodes except root have unique predecessor.
- A collection of trees is called a **forest**.
Example: Binary Search Tree

K, C, A, N, B, V, F, U, D, H, M
Expression Tree

\[(A \cdot (B - C)) + (D / \sim E)\]
Family Tree Terminology

- **Parent** node is directly above **child** node:
  - K is parent to C, N.
- **Sibling** node has same parent:
  - A, F
- K is **ancestor** of B
- B is **descendant** of K
- Node plus all descendants gives **subtree**
More Terminology

- **Simple path** is series of distinct nodes s.t. there is edge between successive nodes.
- **Path length** = # edges in path
- **Height of node** = length of longest path to a leaf
- **Height of tree** = height of root
- **Degree of node** is # of children
- **Degree of tree (arity)** = max degree of any its nodes
- Binary tree has arity ≤ 2.
Even More Terminology

- **Level/depth** of node defined recursively:
  - Root is at level 0
  - Level of any other node is one greater than level of parent
- Level of node is also length of path from root to the node.
Counting

- Lemma: if $T$ is a binary tree, then at level $k$, $T$ has $\leq 2^k$ nodes.
- Theorem: If $T$ has height $h$, then # nodes in $T \leq 2^{h+1} - 1$.
- Equivalently, if $T$ has $n$ nodes then $n - 1 \geq h \geq \log(n + 1) - 1$
Binary Trees in Java

• No implementation in standard Java libraries
• Structure5 has BinaryTree<E> class, but no interface (though we provide one!).
• Like doubly-linked list:
  • value: E
  • parent, left, right: BinaryTree<E>
Linked Representation
Tree Traversals

- Traversals:
  - Pre-Order: root, left subtree, right subtree
  - In-Order: left subtree, root, right subtree
  - Post-Order: left subtree, right subtree, root

- Most algorithms have two parts:
  - Build tree
  - Traverse tree, performing operations on nodes
Evaluate Expression Tree

• Evaluate left subtree, right subtree, perform operation at root.
• Generate stack-based code to evaluate: post-order