CSCI 136: Data Structures and Advanced Programming
Lecture 23
Trees, part 3
Instructor: Kelly Shaw
Williams

Your to-dos

1. Read before Fri: review Bailey, Ch 14.
2. Lab 8 (solo lab), due Tuesday 11/15 by 10pm.
   Note: you will implement a tree data structure called a trie for lab 8; the structure is described in the lab handout. Please bring a short design document to your lab meeting.

Topics

Tree terminology

Activity: Binary Tree Height

The height of a tree is the length of the longest path between the root and any leaf.

Height of tree = 2
Let’s think about some corner cases.

What is the height of a tree with just one node?

The height of a tree is the length of the longest path between the root and any leaf.

Height of tree = 0

What about the empty tree?

The height of a tree is the length of the longest path between the root and any leaf.

Height of tree = -1

Here’s a more formal definition.

The height of a tree is defined as:

-1 if the tree is empty, or
height(left) or height(right), whichever is bigger, +1

empty tree: -1
just a root: 0
any other tree: longest path

How might we implement getHeight()?
Suppose you are asked to write an `Iterator<T>` for a binary tree. What order do you choose?

Remember that tree nodes store data (T). A **traversal** corresponds with the order that data is returned.

**Pre-order traversal:** Return data from each node **before its children**, and then return child data from **left to right**.

Returns the sequence: `a, b, d, e, c, f, g`
**Binary tree traversals**

**In-order traversal**: Return data from each node after its left child and before its right child.

Returns the sequence: \text{d, b, e, a, f, c, g}

**Post-order traversal**: Return data from each node after its children; return child data from left to right.

Returns the sequence: \text{d, e, b, f, g, c, a}

**Level-order traversal** (aka breadth-first order): Return data from each node in level \text{i} before data in level \text{i+1}.

Returns the sequence: \text{a, b, c, d, e, f, g}

---

**Activity: What traversal should I use?**

Suppose I encode the arithmetic expression $1 - 2^4 \times 2$ using the following tree.
Ordered Trees

Binary search tree

A binary search tree is a binary tree that maintains the binary search property as elements are added or removed. In other words, the key in each node:

• must be > any key stored in the left subtree, and
• must be ≤ any key stored in the right subtree.

As with other ordered structures, order is maintained on insertion.

Binary search tree (alternative)

A binary search tree is a binary tree that maintains the binary search property as elements are added or removed. In other words, the key in each node:

• must be ≥ any key stored in the left subtree, and
• must be < any key stored in the right subtree.

As with other ordered structures, order is maintained on insertion.

Key, Value nodes

Note that I said key instead of element.

Storing a key and a value in each node allows the greatest flexibility when arranging a tree. I.e., the key type K need not be the value type V.

Restriction: keys must be comparable in some way (e.g., Comparable<K> or Comparator<K>).
Example

Insert the following elements: 71, 20, 27, 17, 91, 14, 87
Assume K and V are the same.

Example

Insert the following elements: 71, 20, 27, 17, 91, 14, 87
Assume K and V are the same.
Example
Insert the following elements: 71, 20, 27, 17, 91, 14, 87
Assume K and V are the same.

Example
Insert the following elements: 71, 20, 27, 17, 91, 14, 87
Assume K and V are the same.

Example
Insert the following elements: 71, 20, 27, 17, 91, 14, 87
Assume K and V are the same.

Activity
Insert the following elements:
Assume K and V are the same.
Let's start implementing this together.

**Today:**
- Tree terminology
- Tree traversals

**Next class:**
Binary search trees