CSCI 136:  
Data Structures  
and  
Advanced Programming  
Lecture 24  
Trees, part 2  
Instructor: Dan Barowy  
Williams

Topics

Binary tree implementation  
Tree height

Your to-dos

1. Read **before Fri**: reading on balanced trees.  
2. Lab 8 (solo lab), **due Tuesday 4/26 by 10pm**.

Tree ADT

A **tree** is a recursive data structure that stores information hierarchically. A tree is either:

- **empty** (i.e., Ø), or  
- a **node** containing a **value** and references to one or more **trees**.

The empty tree:  

A non-empty **binary** tree:

\[
\begin{array}{c}
\text{b} \\
\text{Ø} \\
\text{a} \quad \text{Ø} \quad \text{Ø} \\
\text{Ø} \\
\text{c} \quad \text{Ø} \quad \text{Ø}
\end{array}
\]
Let's implement this together.

Encode this binary tree using `BinaryTree<T>`

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

Let’s think about some corner cases.

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()?
Let’s implement this together.

**Recap & Next Class**

**Today:**
- Binary tree implementation
- Tree height

**Next class:**
- Binary tree traversals
- Binary search trees
- Tree balance
- Asymptotic analysis