Lecture 15: Binary Trees 2
Fall 2016
Kim Bruce & Peter Mawhorter

Midterm

- Midterm on Wednesday!
  - Prof Mawhorter: office hours Tuesday until 5.
  - Watch Piazza for posts about extra mentoring sessions (Your TAs are busy but trying to arrange something).

This Week

- Assignment: Calculator
  - Postfix calculator
  - Start with simplified version that requires “enter” before each operation
- Lab: Eclipse Debugger
  - Learn how to inspect your program’s state
- No quiz on Friday

Terminology

- What do the following mean?
  - Edge
  - Child/successor
  - Descendant
  - Leaf
  - Interior node
  - Parent/predecessor
  - Forest
  - Height/depth
Terminology

- What do the following mean?
  - Edge – connects two nodes
  - Child/successor – nodes immediately beneath
  - Descendant – child or child of child or ...
  - Leaf – node with no children
  - Interior node – node with children
  - Parent/predecessor – node above (unique)
  - Forest – collection of disconnected trees
  - Height/depth – length of path to leaf (longest)

Nodes in a Tree

- Tree T with n nodes of height h:
  
  \[ n \leq 2^{h+1} - 1 \]

  \[ \log(n + 1) - 1 \leq h \leq n - 1 \]
The height of a tree is a log …unless the tree is a stick.

\[ \log(n + 1) - 1 \leq h \leq n - 1 \]

\[ h = \log(n + 1) - 1 \]

Balanced Trees

(Un)Balanced Trees
Traversals

- Pre-, in-, and post-order.
  - Where does the root node go?
- Build tree, then traverse it.
  - Ideal: root → single leaf

Java Virtual Machine

Twenty Questions

- Guess an animal using only true/false questions.
- Demo program.

$$2^{20} \approx 1,000,000$$

BinaryTree.java

- Uses null where nodes are missing.
Iterators

- Pre-order: root, left subtree, right subtree
- Post-order: left subtree, right subtree, root
- In-order: left subtree, root, right subtree

In-order Traversal

```java
String inorder() { 
    String result = "";
    if (left != null) { 
        result = left.inorder() + ",";
    }
    result += this.value.toString();
    if (right != null) { 
        result += "," + right.inorder();
    }
    return result;
}
```

Lambda Expressions

```java
public void inorder(Consumer<? super E> action) { 
    if (left != null) { 
        action.accept(left);
        action.accept(this); 
        action.accept(right);
    }
}
```

```
Consumer objects have an accept method
Anonymous functions can be consumers
Java figures out the type of "s"
```

Keeping Track with a Stack

- See e.g., BinaryTree.java and BTPreorderIterator.java
Anonymous Function Limits

- Can’t modify outside variables:

```java
int sum = 0;
treevalInorder(left, right, base) = sum = sum + 1;
```

...this is illegal!