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

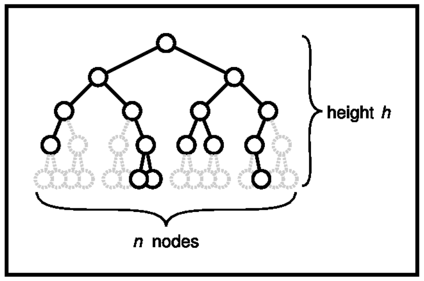
- Assignment: Calculator
  - Postfix calculator
  - Start with simplified version that requires "enter" before each operation
- Lab: Eclipse Debugger
  - Learn how to inspect your program's state
- No quiz on Friday

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)



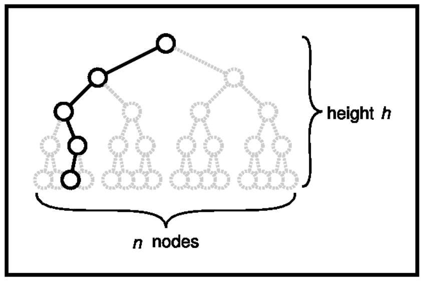
$$n \le 2^{h+1} - 1$$

Nodes in a Tree

- Tree T with n nodes of height h:

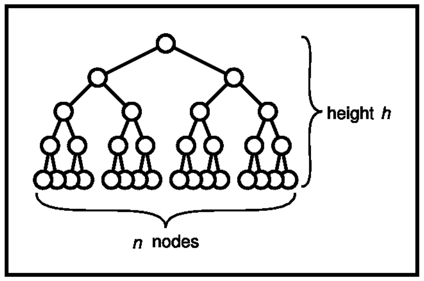
$$n \le 2^{h+1} - 1$$

$$\log(n+1) - 1 \le h \le n - 1$$



$$\log(n+1) - 1 \le h \le n - 1$$

$$h = n - 1$$



$$\log(n+1)-1 \le h \le n-1$$

$$h = \log(n+1)-1$$

The height of a tree is a log …unless the tree is a stick.

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

```
int simple(int m, int n) {
  return (m + n - 1);
}
```

translates to

```
method int simple(int, int)
0 iload_1
1 iload_2
2 iadd
3 iconst_1
4 isub
5 ireturn
```

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

Keeping Track with a Stack

- See e.g., `BinaryTree.java` and `BTPreorderIterator.java`

Lambda Expressions

```java
public void doPostorder(Consumer<? super E> action) {
  if (!isEmpty()) {
    left.doPostorder(action);
    right.doPostorder(action);
    action.accept(val);
  }
}

tree.doPostorder(s -> System.out.println(s + " "));
```

`Consumer` *objects have an* `accept` *method*

Anonymous functions can be `Consumer`*s*

Java figures out the type of "`s`*"*

Anonymous Function Limits

- Can't modify outside variables:

```
int sum = 0;
tree.doInorder(s -> sum = sum + s);
```

…this is illegal!

Calculating using Lambdas

```
public E calcPreorder(TrinaryFunction<E> operation, E base) {
  if (isEmpty()) {
    return base;
  } else {
    return operation.apply(
      val,
      left.calcPreorder(operation, base),
      right.calcPreorder(operation, base)
    );
  }
}

System.out.println(
  "The sum is "
+ tree.calcPostorder((s, t, u) -> s + t + u, 0)
);
```