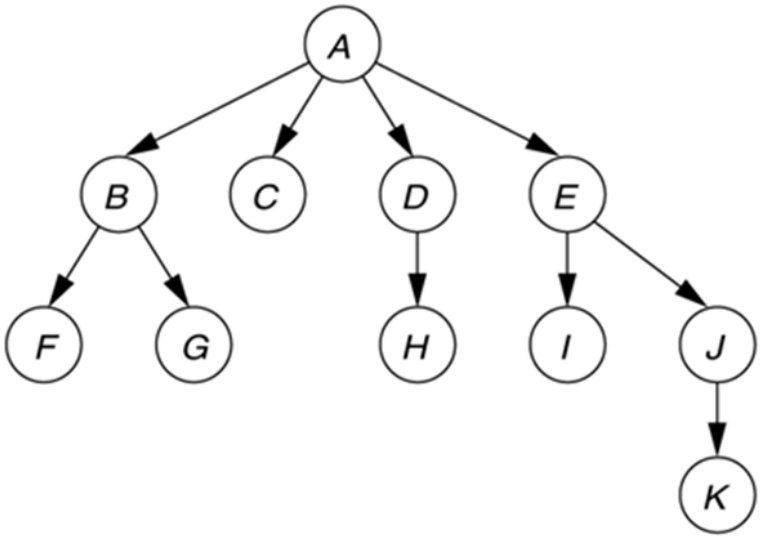
CSSE 230 Day 9

More simple BinaryTree methods
Tree Traversals



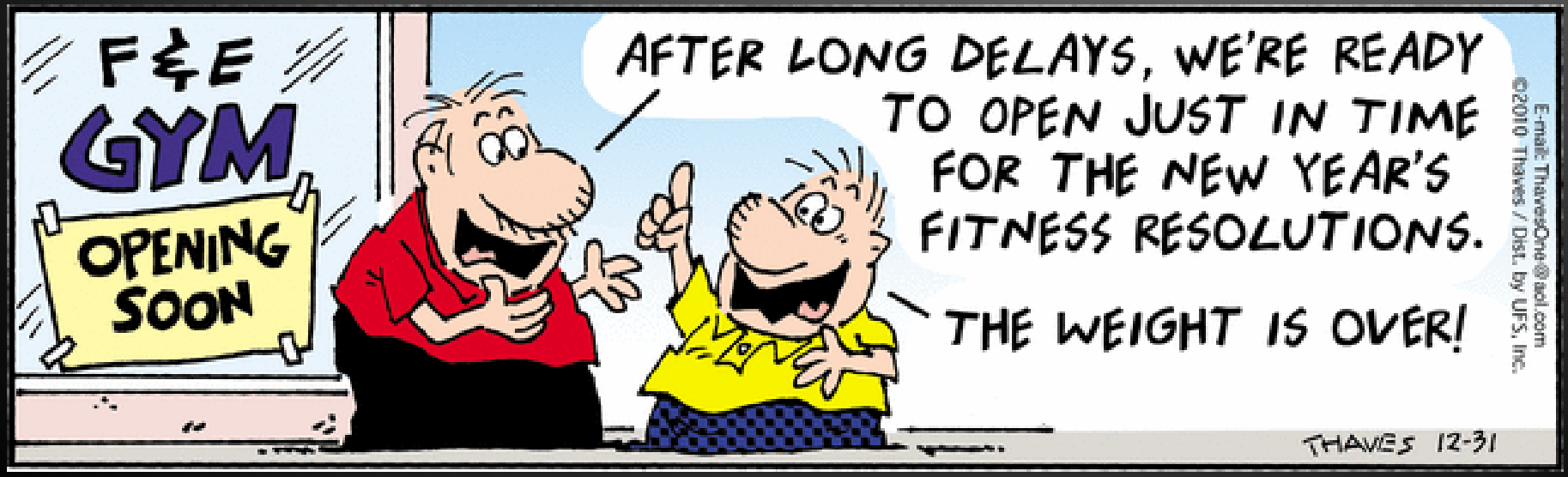
Reminders

- Exam 1 – Day 11 in class
  - Coverage:
    - Everything from reading and lectures, Sessions 1-9
    - Programs through Hardy
    - Written assignments 1-3
  - Allowed resources:
    - Written part: One side of one 8.5 x 11 sheet of paper
    - Programming part:
      - Textbook
      - Eclipse (including programs you wrote in your repos)
      - Course web pages and materials on ANGEL
      - Java API documentation
    - A previous 230 Exam 1 is available in Moodle

Exam 1 Possible Topics

- Sessions 1-11
  - Terminology
  - OOP and inheritance
  - Growable Arrays
  - Homework and Programs
  - Big-oh, Big-Omega, and Big-Theta
  - Limits and asymptotic behavior
  - Basic data structures
  - Comparable and Comparator
  - MCSS
  - Recursion, stack frames
  - Recursive binary search
  - Binary trees
  - Binary tree traversals
  - Size vs. height for binary trees
  - Binary Search Tree basics
  - No induction problems yet.

Questions?


F&E
GYM
OPENING SOON
AFTER LONG DELAYS, WE'RE READY TO OPEN JUST IN TIME FOR THE NEW YEAR'S FITNESS RESOLUTIONS.
THE WEIGHT IS OVER!
THAVES 12-31
E-mail: ThavesOne@aol.com
©2010 Thaves / Dist. by UFS, Inc.

Agenda

- Another induction example
- Implementing Binary Trees (continued)
- Binary Tree Traversals
- Hardy/Colorize work time

Another induction proof example

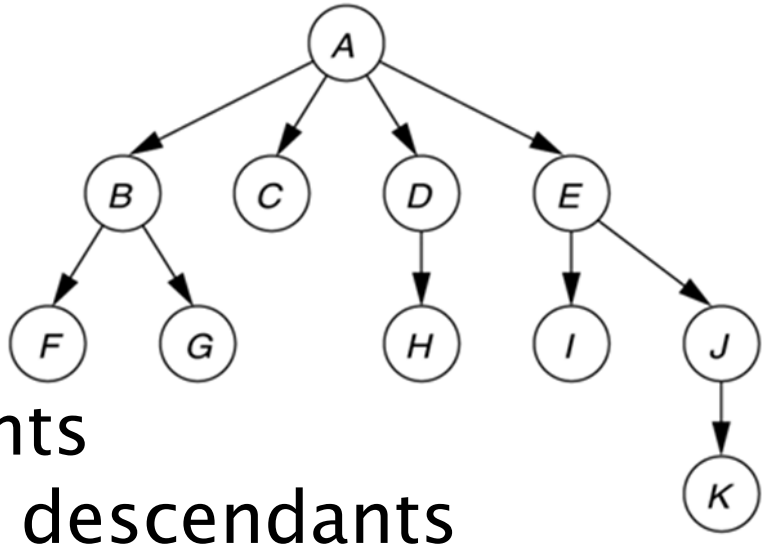
Show by induction that $2n + 1 < n^2$ for all integers $n \geq 3$

There are other ways that we could show this (using calculus, for example)

But for now the goal is to have another example that can illustrate how to do proofs by induction

Recap: Tree Terminology

- Parent
- Child
- Grandparent
- Sibling
- Ancestors and descendants
- Proper ancestors, proper descendants
- Subtree
- Leaf, interior node
- Depth and height of a node
- Height of a tree



Growing Trees

Let's continue implementing a *BinaryTree<T>* class including methods *size()*, *height()*, *duplicate()*, and *contains(T)*.

Binary tree traversals

- PreOrder (top-down, depth-first)
  - root, left, right
- PostOrder (bottom-up)
  - left, right, root
- InOrder (left-to-right, if tree is spread out)
  - Left, root, right
- LevelOrder (breadth-first)
  - Level-by-level, left-to-right within each level

If the tree has N nodes, what's the (worst-case) big-Oh run-time of each traversal? big-Oh space used?

```
// Print tree rooted at current node using preorder
public void printPreOrder( )  {
    System.out.println( element );        // Node
    if( left != null )
        left.printPreOrder( );            // Left
    if( right != null )
        right.printPreOrder( );           // Right
}


// Print tree rooted at current node using postorder
public void printPostOrder( ) {
    if( left != null )
        left.printPostOrder( );           // Left
    if( right != null )
        right.printPostOrder( );          // Right
    System.out.println( element );        // Node
}

// Print tree rooted at current node using inorder t
public void printInOrder( ) {
    if( left != null )
        left.printInOrder( );             // Left
    System.out.println( element );        // Node
    if( right != null )
        right.printInOrder( );            // Right
}
```

Binary Tree Iterators

What if we want to iterate over the elements in the nodes of the tree one-at-a-time instead of just printing all of them?

Hardy Work time

The assistants and I will be available for help