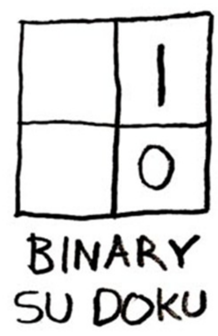


CSSE 230 Day 10

Binary Tree Properties
Displayable Binary Trees

Announcements

- Exam 1 Wednesday 7 PM
  - Optional Q&A session Tuesday 10th hour
    - O269
  - I plan to be available almost all day Wednesday
- Hardy/Colorize Partner Evaluation due Wednesday at noon
- WA4 due Friday at 5:00 PM
  - I will be out of town and have limited internet access Friday and Saturday
- Displayable due Monday, April 9 at 8 AM
  - With a "grace day" until Tuesday 8 AM.
  - A grace day is a "free late day".
  - You may not use additional late days.
  - Ideally you should finish it before Monday, so you have sufficient time for the next assignment.

Recap: Exam Announcements

- Exam 1 Wednesday 7 PM (O267-269)
  - Coverage:
    - Everything from reading and lectures, Sessions 1-10
    - Programs through Hardy/Colorize
    - 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 in your workspace repositories)
      - Course web pages and materials on ANGEL
      - Java API documentation
  - A previous 230 Exam 1 is available on ANGEL

No devices with headphones or earbuds are allowed

Exam 1 Topics

- Sessions 1-10
  - Terminology
  - Growable Arrays
  - Homework and Programs
  - Big-oh, Big-Omega, and Big-Theta
  - Limits and asymptotic behavior
  - Basic data structures
  - Comparable and Comparator
  - MCSS
  - Finite State Machines
  - Recursion, stack frames
  - Recursive binary search
  - Binary trees
  - Binary tree traversals
  - Binary tree iterators
  - Size vs. height for binary trees
  - No induction problems yet.

Agenda

- Another induction example
- Finish Tree iterators
- WA4 hints, questions
- More binary trees
- # of nodes in Binary tree with height h
- Displayable Binary Trees

Recap: 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?

Implementing Binary Tree Iterators

- What methods does an iterator typically provide?
  - Weiss uses: **first, isValid, advance, retrieve**

TreeIterator abstract class

```
// TreeIterator class; maintains "current position"
//
// CONSTRUCTION: with tree to which iterator is bound
//
// ******************PUBLIC OPERATIONS**********************
//     first and advance are abstract; others are final
// boolean isValid( )   --> True if at valid position in tree
// Object retrieve( )   --> Return item in current position
// void first( )        --> Set current position to first
// void advance( )      --> Advance (prefix)
// ******************ERRORS*********************************
// Exceptions thrown for illegal access or advance
```

TreeIterator class's fields and methods

```java
protected BinaryTree t;     // Tree
protected BinaryNode current;     // Current position

public TreeIterator( BinaryTree theTree )  {
    t = theTree;
    current = null;
}

abstract public void first( );

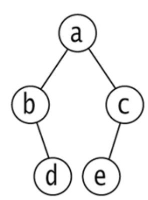
final public boolean isValid( )  {
    return current != null;
}

final public Object retrieve( ) {
    if( current == null )
        throw new NoSuchElementException( );
    return current.getElement( );
}

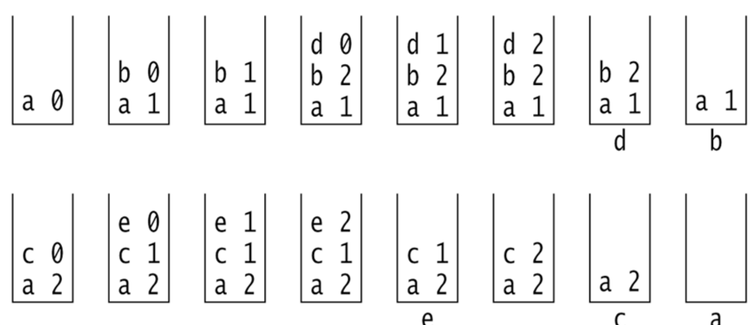
abstract public void advance( );
```

Recap The Stack in a PostOrder iterator

Q1



For InOrder each element is only pushed twice. Show how it works.



Alternative:

- Each node can store pointer to the next node of a particular traversal
- Must update this extra info in constant time as tree changes

An upcoming written assignment will include these "threaded binary trees"

Wouldn't it be nice?

- If we did not have to maintain the stack for these iterators?
- If we could somehow "tap into" the stack used in the recursive traversal?
  - I.e. Take a "snapshot of that call stack, and restore it later when we need it.
  - This is called a continuation.
    - A big subject in the PLC course, CSSE 304

Q2

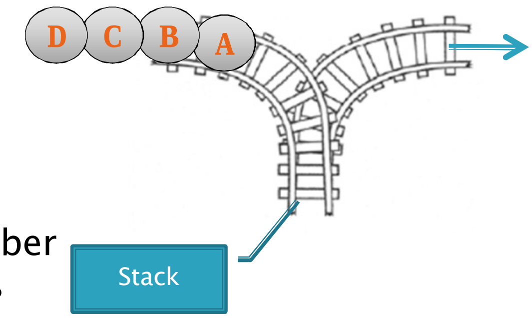
Another induction proof

- Show by induction that n(n+1)(n+2)is divisible by 6 for all non-negative integers n.

Tips on WA4

WA4, Problem 2 Application

- Railroad switching



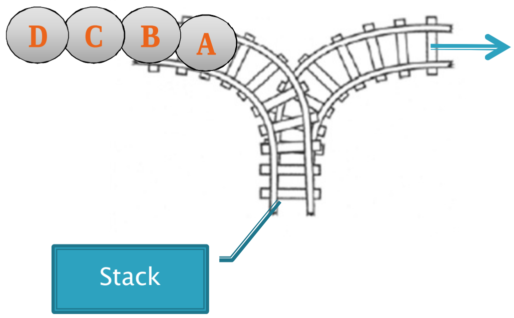
- Problem is equivalent to counting the number of possible orders the cars can leave the station

General Approach to Puzzle Problems

- Make up tiny examples like the given problem
  - No really tiny, I'm serious
- Solve the tiny problem
- Solve a slightly larger problem
- Solve a slightly larger problem than that
- Once you see the pattern, then try to solve the given problem

What's the smallest problem like this?

- In how many possible orders can the cars leave the station?



More Binary Trees

- If a tree falls in the forest and there are two people around to hear it…

Merge Method (from Weiss chapter 18)

```
/** Replaces the root element of this
 *  tree with the given item and the
 *  subtrees with the given ones.
 * … */
public void merge(T rootItem,
                  BinaryTree<T> left,
                  BinaryTree<T> right)
```

- Simple approach:
  - ```
    this.root = new BinaryTreeNode<T>(rootItem,
                                      left.root,
                                      right.root);
    ```

What could go wrong?

Problems With Naïve Merge

- A node should be part of one and only one tree.

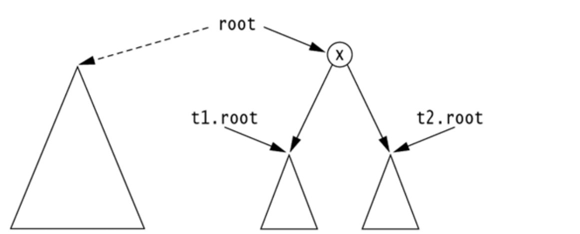
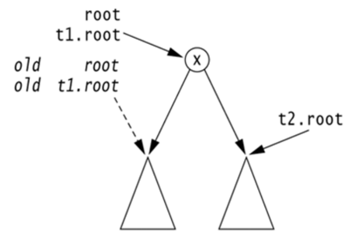


figure 18.14

Result of a naive `merge` operation: Subtrees are shared.

figure 18.15

Aliasing problems in the `merge` operation; `t1` is also the current object.


    ```

Correct Merge Method

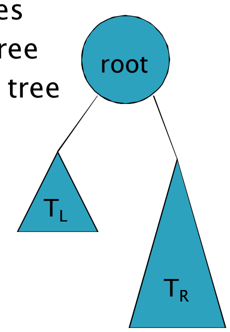
```java
/**
 * Merge routine for BinaryTree class.
 * Forms a new tree from rootItem, t1 and t2.
 * Does not allow t1 and t2 to be the same.
 * Correctly handles other aliasing conditions.
 */
public void merge( AnyType rootItem,
                   BinaryTree<AnyType> t1, BinaryTree<AnyType> t2 )
{
    if( t1.root == t2.root && t1.root != null )
        throw new IllegalArgumentException( );

        // Allocate new node
    root = new BinaryNode<AnyType>( rootItem, t1.root, t2.root );

        // Ensure that every node is in one tree
    if( this != t1 )
        t1.root = null;
    if( this != t2 )
        t2.root = null;
}
```

Weiss, figure 18.16

Size vs. Height in Binary Trees

Binary Tree: Recursive definition

- A Binary Tree is either
  - empty, or
  - consists of:
    - a distinguished node called the root, which contains an element, and two disjoint subtrees
    - A left subtree T_L, which is a binary tree
    - A right subtree T_R, which is a binary tree



Time out for math!

- Want to prove some properties about trees
- Weak induction isn't enough
- Need strong induction instead:

The former governor of California

Strong Induction

- To prove that p(n) is true for all n >= n_0:
  - Prove that p(n_0) is true, and
  - For all k > n_0, prove that if we assume p(j) is true for $n_0 \leq j < k$, then p(k) is also true

- Weak induction uses the previous domino to knock down the next
- Strong induction uses a whole box of dominoes!

Q3-5

Size and Height of Binary Trees

- Notation:
  - Let T be a tree
  - Write h(T) for the height of the tree, and
  - N(T) for the size (i.e., number of nodes) of the tree

- Given h(T), what are the bounds on N(T)?

- Given N(T), what are the bounds on h(T)?

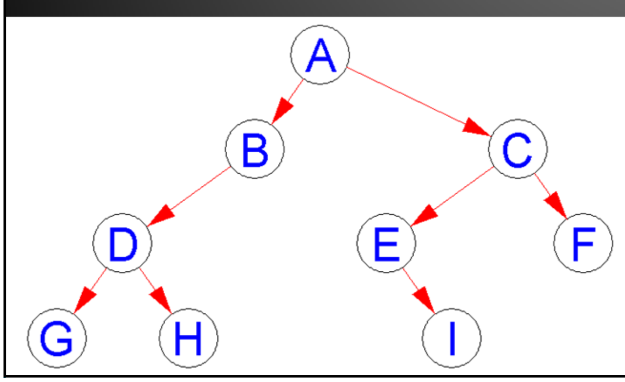
Q6-7

Extreme Trees

- A tree with the maximum number of nodes for its height is a **full** tree.
  - Its height is O(log N)
- A tree with the minimum number of nodes for its height is essentially a _____ _____
  - Its height is O(N)
- Height matters!
  - We will see that the algorithms for search, insertion, and deletion in a Binary search tree are O(h(T))

Displayable Binary Trees

» Some suggestions



Displayable Binary Tree is an individual assignment

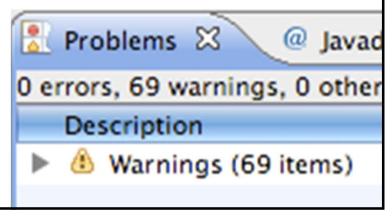
- Check out **Displayable** from your individual repo.
- You will get errors on the Weiss imports like

```
import weiss.nonstandard.Stack;
```

- So install the Weiss packages now. See:

http://www.rose-hulman.edu/class/csse/csse230/201230/Slides/WeissPackage.html

- Should be no errors.
- If errors, see next slide.



Troubleshooting the Weiss install

- Close all Eclipse projects except **Displayable**
- Did you put jars in the right folder?
- Are the jars and not zips?
- Is Eclipse using that JRE?
  - See **Windows → Preferences**, then **Java → Installed JREs → Edit.**
  - They should be in that list.

Get help now if you're stuck.
Help others if you aren't.

Work time

- WA 4 or Displayable

  If you wish, read the hints on remaining slides after you read the Displayable spec.

Displayable Binary Trees Steps

- Solve the sub-problems in this order:
  - `BuildTree.preOrderBuild()`
  - `BinaryTree.inOrder()`
  - Graphics
- Run `CheckDisplaybleBinaryTree` to test
  - Doesn't use JUnit
  - Tests `preOrderBuild` and `inOrder` first
  - Prompts for test case for which to display graphics
  - Each tree should be displayed in a separate window.

Better Exception Reporting in CheckDisplayableBinaryTrees

- Add a stack trace in **main()**

```java
                    tp.dbTree.display();
                } catch (InternalError e) {
                    System.out
                            .println("You must 
                }

            }
        } catch (Exception e) {
            System.out.println(e.toString());
            e.printStackTrace();
        }

    }

    private static String inOrder(int index) {
        switch (index) {
```

preOrderBuild Hints

- Like WA4, problem 3
- Consider:
  - **chars = 'ROSEHULMAN'**
  - **children = '22002R0RL0'**

inOrder Hints

- The iterators in TestTreeIterators.java are there for a reason!

- Recall how we can use Weiss iterators in a for loop:
  - ```
    for(iter.first();iter.isValid();iter.advance()) {
        Object elem = iter.retrieve();
        // ... do something with elem ...
    }
    ```

Graphics Hints

- Suggested order for your graphics work:
  - Figure out how to calculate node locations
  - Get code to display correctly sized windows
  - Add code to draw nodes
  - Add code to draw lines
  - Only work on arrow heads if all the rest works!

- Remember the **TreesSolution** project
  - Shows all the basic graphics except **drawString()**
    ```