CSSE 230 Day 11

Binary Search Tree intro
BST with order properties

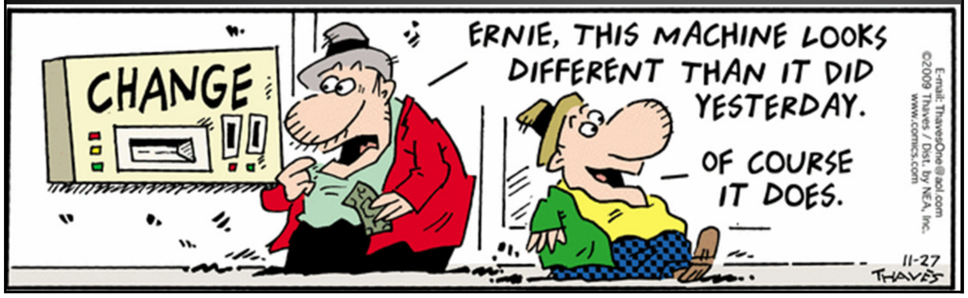
Check out BST project from SVN

Announcements

- Exam 1 Wednesday 7 PM (Sec 1: O267 Sec 2: O269)
  - Optional Q&A session today 10th hour O269
  - I plan to be available almost all day Wednesday
- Hardy/Colorize Partner Evaluation due Wednesday at noon
- Doublets Partner Preference survey due Thursday at noon. Teams of 2 again.
- WA4 due Friday at 5 PM
  - I will be out of town and will have limited internet access Friday and Saturday (available most of Thursday).
- Displayable due Monday, April 9 at 8 AM
  - With a "grace day" until Tuesday, April 10, at 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.
- WA5 assignment document has been updated for this term. Due Thursday, April 12 at 8 AM

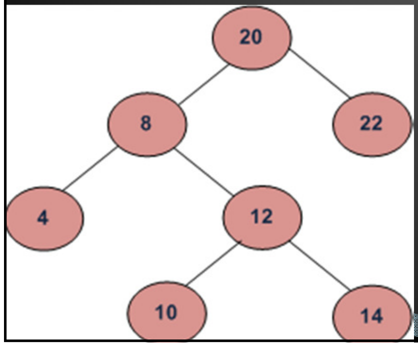
Questions?

- Exam, Displayable, WA4, …


CHANGE
ERNIE, THIS MACHINE LOOKS DIFFERENT THAN IT DID YESTERDAY.
OF COURSE IT DOES.
11-27
THAVES
E-mail: ThavesOne@aol.com
©2009 Thaves / Dist. by NEA, Inc.
www.comics.com

Binary Search Trees

- Binary Trees that store elements in increasing order



A Binary Search Tree (BST) allows easy and fast lookup of its items because it keeps them ordered

Q1

Draw a "birthday BST"

- A BST is a Binary Tree T with these properties:
  1. Elements are Comparable, and non-null
  2. No duplicate elements
  3. All elements in T's left subtree are less than the root element
  4. All elements in T's right subtree are greater than the root element
  5. Both subtrees are BSTs
- Advantage: Lookup of items is O(height(T))
- What does the inorder traversal of a BST yield?

BST insert, contains, and delete are different than in a regular binary tree

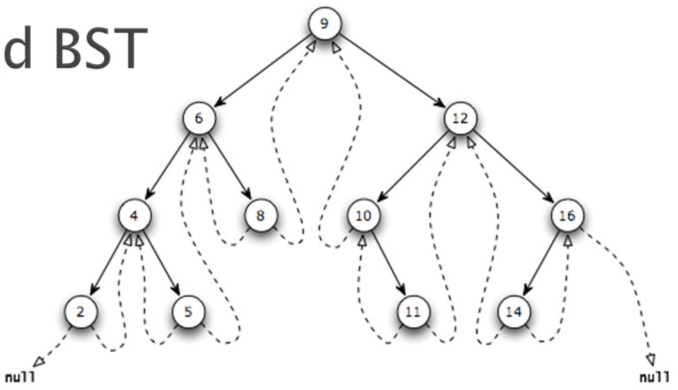
Q2-5

```
public class BinarySearchTree<T extends Comparable<T>> {

  private BinaryNode<T> root;

  public BinarySearchTree() {
    this.root = null;
  }

  // insert obj, if not already there
  public void insert(T obj)

  // Does this tree contain obj?
  public boolean contains(T obj)

  // delete obj, if it's there
  public void delete(T obj)
```

Threaded BST

- Quick preview of a WA5 problem

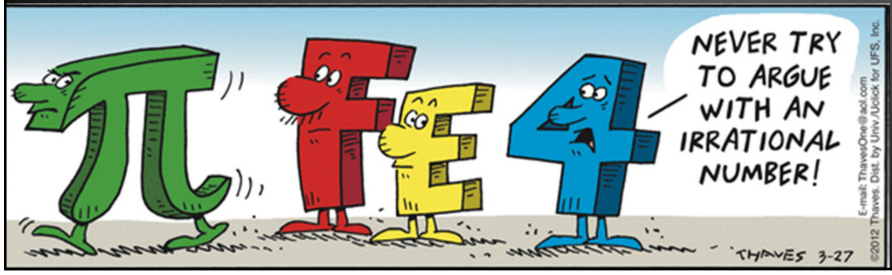
Threaded BST



We use the "unused" null pointers to point to a node's inorder successor (right thread) and inorder predecessor (left thread)

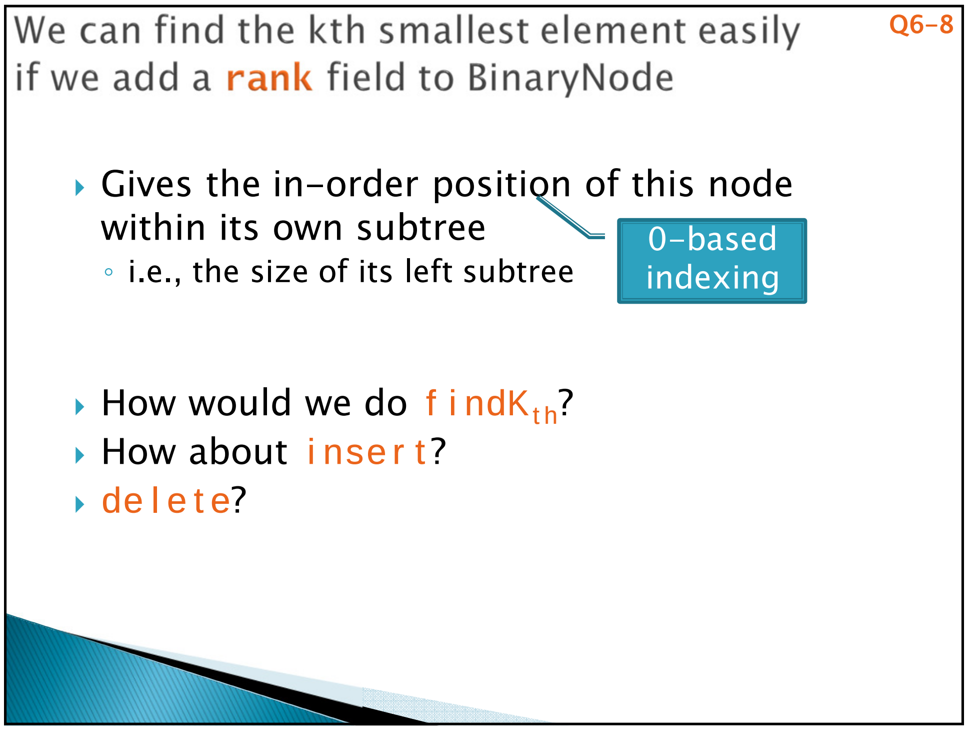
BST with Rank

- Explore the concept
  How do Find and Insert work?


NEVER TRY TO ARGUE WITH AN IRRATIONAL NUMBER!
E-mail: ThavesOne@aol.com
©2012 Thaves. Dist. by Univ./Uclick for UFS, Inc.
THAVES 3-27

BSTs are an efficient way to represent ordered lists

- What's the performance of
  - insertion?
  - deletion?
  - find?
  - iteration?
- What about finding the kth smallest element?



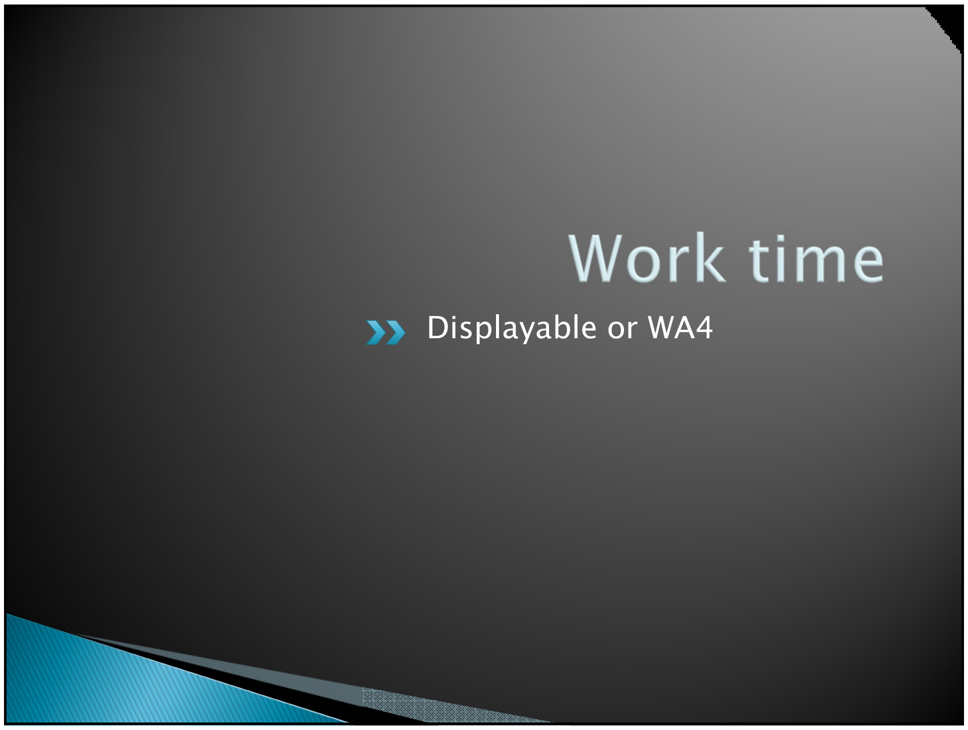
We can find the kth smallest element easily if we add a **rank** field to BinaryNode

Q6-8

- Gives the in-order position of this node within its own subtree
  - i.e., the size of its left subtree

0-based indexing

- How would we do findK$_{th}$?
- How about insert?
- delete?



Work time

- Displayable or WA4