CS 367 - Introduction to Data Structures
Tuesday, March 29, 2016

Homework 7 due 10 pm Friday, April 1st

Program 4 assigned tomorrow

Last Time

General Trees
- determining tree height

Binary Trees
- implementing

Tree Traversals
Categorizing ADTs Part 2
Binary Search Tree (BST)

Today

Binary Search Tree (BST)
- BSTnodes
- BST class
- implementing print
- implementing lookup, insert, delete
- complexities of BST methods

CS Options/Courses

Next Time

Read: *Red-Black Trees*
Classifying Binary Trees
Balanced Search Trees
Red-Black Trees
- tree properties
- print, lookup
- insert

Recall BSTs and `BSTnodes`

BST – Binary Search Tree

- is a key-value oriented collection of items where duplicate keys are not allowed
- **goal:** combine speed of binary search for access in an array O( )
  with speed of linking/unlinking in a chain of nodes O( )
- **shape constraint:** binary tree structure
- **order constraint:**

- in lecture, we'll explore BSTs with items having only a key value
  see readings for items having a key value and an associated value

BSTnode Class

```
class BSTnode<K> {

   private K key;
   private BSTnode<K> left, right;

   public BSTnode(K key, BSTnode<K> left, BSTnode<K> right) {
      this.key   = key;
      this.left  = left;
      this.right = right;
   }

   public K getKey() { return key; }
   public BSTnode<K> getLeft()  { return left; }
   public BSTnode<K> getRight() { return right; }

   public void setKey(K newK)             { key = newK; }
   public void setLeft(BSTnode<K> newL)   { left = newL; }
   public void setRight(BSTnode<K> newR) { right = newR; }
}
```

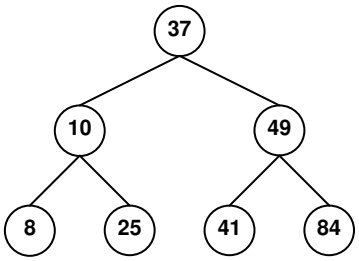
→ Draw a picture of the memory layout of a BSTnode.

BST Class

```
import java.io.*;  //for PrintStream

public class BST<K extends Comparable<K>> {

   private BSTnode<K> root;

   public BST() { root = null; }

   public void insert(K key)
                throws DuplicateException {


   }

   public void delete(K key) {


   }

   public boolean lookup(K key) {


   }

   public void print(PrintStream p) {


   }

   //add helpers ...




}
```

Implementing `print`

→ **Write a recursive definition** to print a binary tree given **`n`**, a reference to a BSTnode.



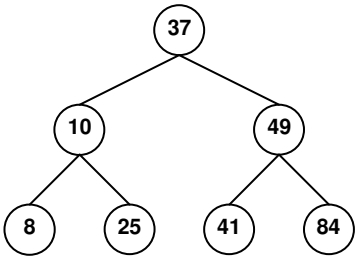
→ **Complete the recursive print method based** on the recursive definition.

```
public void print(PrintStream p) {
   print(root, p);
}

private void print(BSTnode<K> n, PrintStream p) {
```

Implementing `lookup`

Pseudo-Code Algorithm

```
private boolean lookup(BSTnode<K> n, K key) {
```

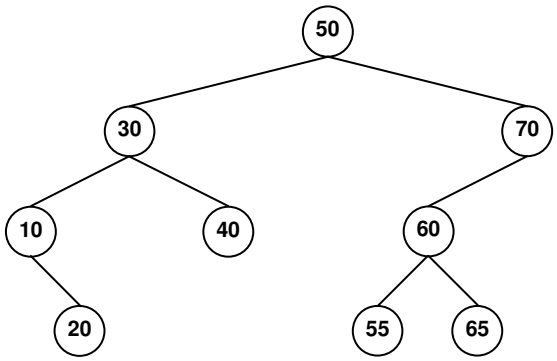


Implementing `insert`

High-Level Algorithm

```
private BSTnode<K> insert(BSTnode<K> n, K key)
                                  throws DuplicateException {
```

Practice - Inserting into a BST



→ **Insert 5, 27, 90, 73, 57 into the tree above.**

→ **What can you conclude about the shape of a BST when values are inserted in sorted order?**

→ **Will you get a bad shape *only* if values are inserted in sorted order?**

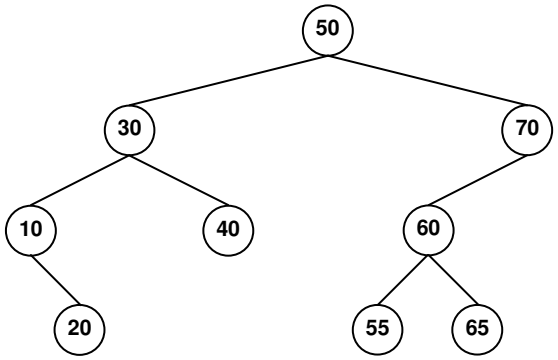
✵

Implementing `delete`

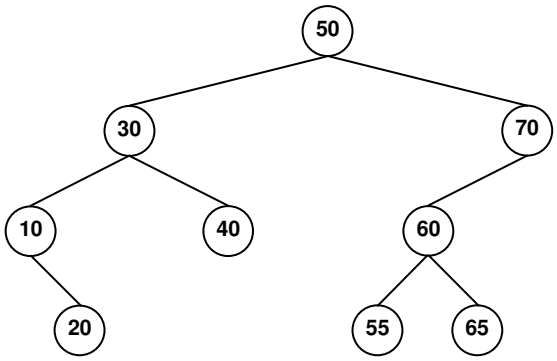
High-Level Algorithm

```
private BSTnode<K> delete(BSTnode<K> n, K key) {
```

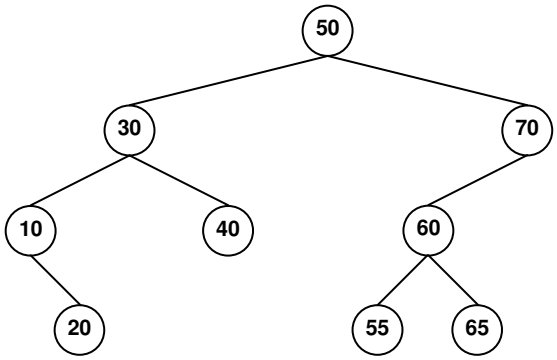
Practice - Deleting from a BST



→ Delete 90 from the tree above.
→ Delete 40 and 65 from the tree above.



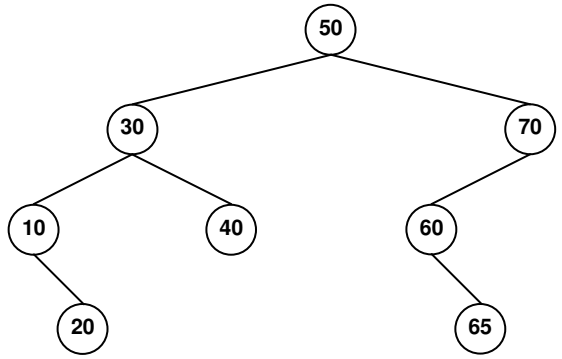
→ Delete 10 and 70 from the tree above and redraw the tree.



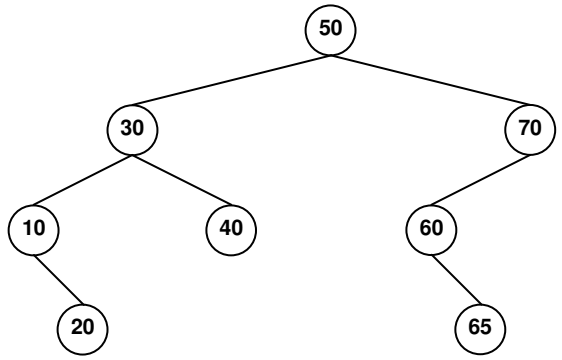
→ How do you delete 50 or 30 from the tree above?

Implementing `delete` (cont.)

Practice - Deleting from a BST



→ Delete 30 from the tree above using the _____.



→ Delete 50 from the tree above using the _____.

Complexities of BST Methods

Problem size: N =

print:

lookup:

insert:

delete:

CS Options

CS Certificate

5 CS Courses (12 credits minimum)
- Data Structures – CS 367 (& possibly prereq 302)
- 2 CS Courses >=400 level
- 2 Other CS Courses

CS Major

Basic CS
- Discrete Math – CS 240
- Programming + Data Structures – CS 302, CS 367
- Basic Systems – CS 252, CS 354

Math
- Calculus – MA 221, MA 222
- 2 Beyond Calc – STATS 324 (intro applied stats), MA 340 (linear algebra)

Group A Theory
- Algorithms – CS 577

Group B Hardware/Software
- OS – CS 537

Group C Applications
- AI – CS 540

Group D Electives
- 2 CS Courses >=400 level

CS Double Major

- Must complete major requirements
- Easy for Computer Engineering Majors

CS Courses

Take Next

- CS 240 Introduction to Discrete Mathematics
- CS 252 Introduction to Computer Engineering (prereq for CS 354)
- CS 354 Machine Organization and Basic Systems (prereq for many group B)
- (CS 368) Learning a New Programming Language (C++ for CS 537)

- NOTE: CS 352 Digital Systems Fundamentals no longer required

>= 400 can take after CS 367

- CS 407 Foundations of Mobile Systems (spring, popular)
- CS 540 Introduction to Artificial Intelligence
- CS 570 Human Computer Interaction (spring)

>= 400 can take after CS 367 + Math

- CS 412 Introduction to Numerical Methods – MA 222 + MA 234 or CS 240
- CS 435 Introduction to Cryptography – MA 320 or MA 340
- CS 525 Linear Programming Methods – MA 320 or MA 340 or MA 443
- CS 533 Image Processing – MA 320 or MA 340 (fall)
- CS 559 Computer Graphics – MA 320 or MA 340
- CS 576 Introduction to Bioinformatics – MA 222 (fall)
- CS 577 Introduction to Algorithms – CS 240