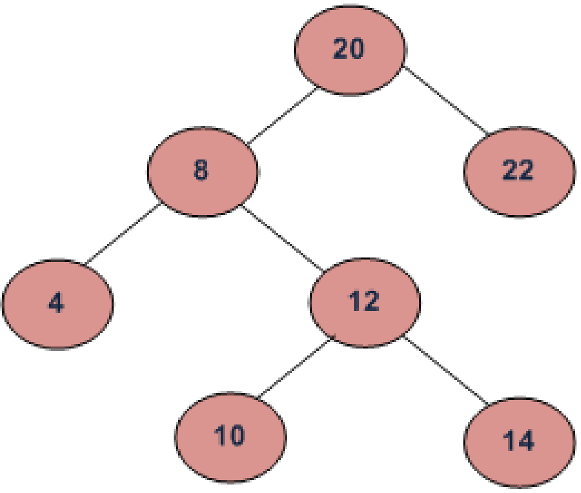


CSSE 230 Day 10

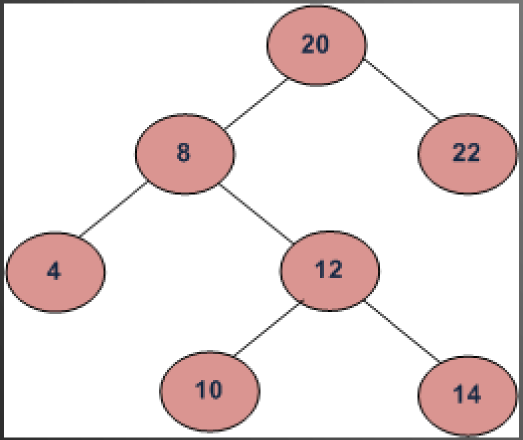
Binary Search Tree intro
BST with order properties

After today, you should be able to…
… implement insertion into a BST
… implement search (contains) in a BST
… implement deletion from a BST

Announcements

- Doublets
  - Due tonight
  - Team eval due Monday
  - Behavior of different ChainManagers?
- Quiz review problems

Binary Search Trees



Binary Trees that store elements so that an they appear in increasing order in an in-order traversal

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

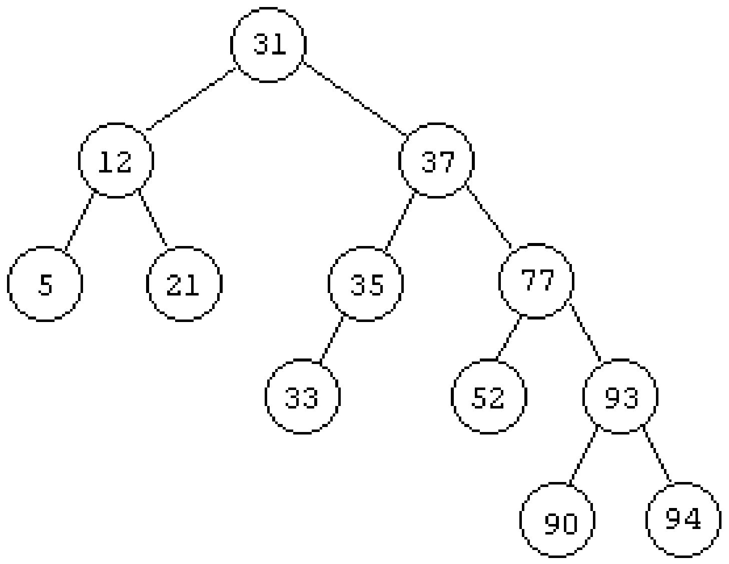
Draw a "birthday BST"

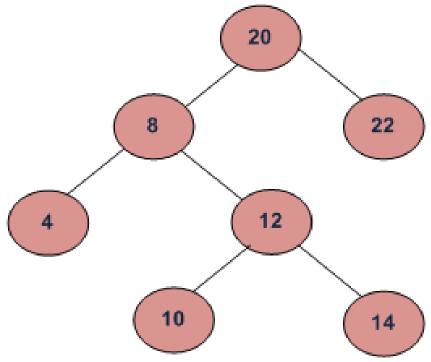
- A BST is a Binary Tree T with these properties:
  1. Elements are Comparable, and non-null
  2. No duplicate elements (we implement TreeSet)
  3. All elements in T's left subtree are less than the root element
  4. All elements in T's right subtree are greater than the root element
  5. Both subtrees are BSTs

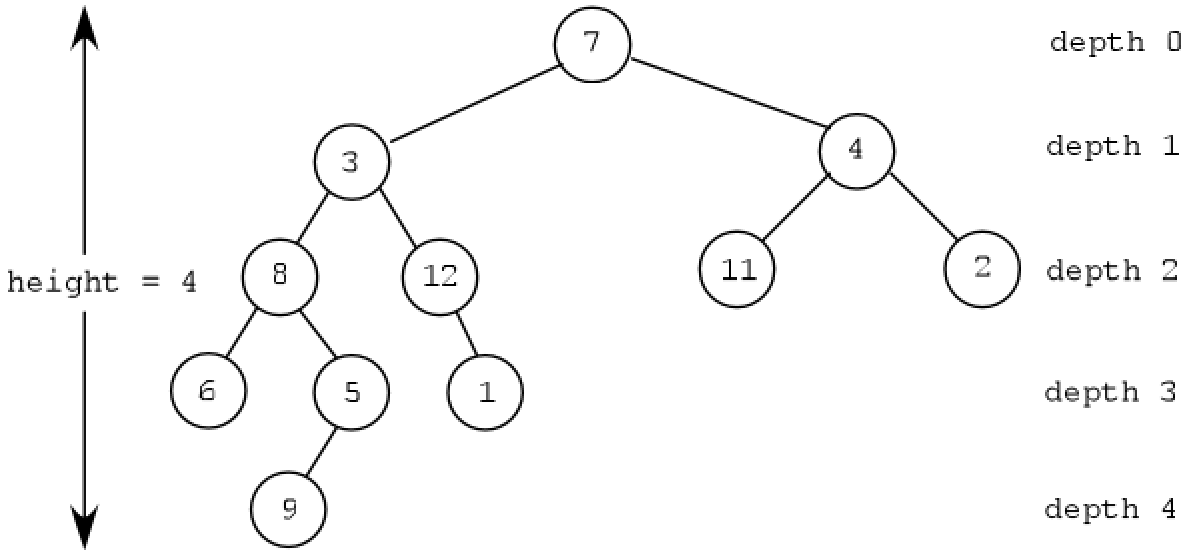
BST insert, contains, and delete need to maintain BST properties

- Search (contains) is now easier, and *possibly* more efficient
  - Why?
  - What can we say about running time of contains()?
- How to insert a new item?
- How to delete an item?
- Running times?

Example Trees







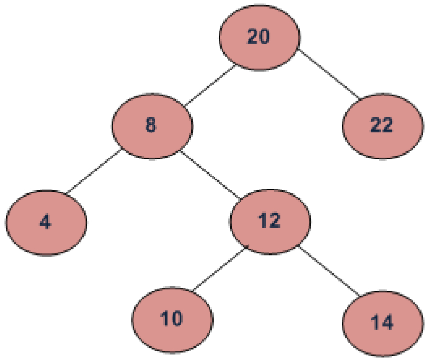
Implementation

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

 private BinaryNode root;
 public BinarySearchTree() {
   this.root = NULL_NODE;
 }
 // Does this tree contain x?
 public boolean contains(T x)

 // insert x. If already there, return false
 public boolean insert(T x)

 // delete x. If not there, return false
 public boolean delete(T x)
           // 3 cases (see text)
```



https://en.wikipedia.org/wiki/Binary_search_tree#Deletion

http://stackoverflow.com/questions/21800298/remove-a-node-in-binary-search-tree

Hibbard deletion: http://dl.acm.org/citation.cfm?id=321108

Implementation issues, part 1 (notes from spec)

- The **recursive BinaryNode** insert() and delete() in the text return BinaryNodes. We want our BinaryNode.insert operation to return a reference to a BinaryNode and also a Boolean.
  So how do the BinaryNode methods return Booleans?

- Could let the Boolean be a BinarySearchTree field. But this field acts like a global variable to the recurseive BinaryNode insert() delete() metods. But could encapsulate better.

- Can the helper method return 2 things?
  - Create a simple composite class to hold both a boolean and a BinaryNode.

- Can you pass a parameter to the helper method and mutate it?
  - Parameters are call-by-value, so primitives can't be mutated.
  - Pass a simple BooleanContainer object so you can mutate the boolean inside.

Implementation issues, part 2

- Modifying (inserting/deleting) from a tree should cause any active iterators to fail the next time the active iterator is accessed (i.e., throw a `ConcurrentModificationException`).
  - How do you detect this?

- How do you implement an iterator's remove()?
  - Just call BST remove().
  - But throw exceptions if next() hasn't been called, or if remove() is called twice in a row. (Javadoc for TreeSet iterator has details.)