Administrative Details

- Pizza info session at 9pm in common room
- Darwin lab
  - Part 1 due tonight; Part 2 due next Monday
- Midterm 2
  - Wednesday during lab in Wege (1pm-3pm)
  - Covers Ch 7, 8, 10-13, Closed book
  - Review: Tuesday 9:30pm-10:30pm, TCL 202
  - No class on Wednesday (but I'll be in my office)
- Office hours today: 2-3:30

Last Time

- Started talking about BSTs
- Learned how to locate elements to a BST

Today’s Outline

- Wrap up binary search trees
- Maybe start talking about Graphs (Ch 16)
  - Learn a bit more about graphs during next lab

Implementing BSTs

- Important BST methods (from last time):
  - Constructor(s)
  - protected BT locate(BT root, Object value)
- Today we’ll cover:
  - public boolean contains(Object value)
  - public Object get(Object value)
  - public void add(Object value)
  - protected BT predecessor(BT root)

Recap: locate

protected BT locate(BT top, Object value) {
  // pre: top and value are non-null
  // post: returns "highest" node with the desired value,
  //       or node to which value should be added
  Object topValue = top.value();
  BT child;
  // found at top: done
  if (topValue.equals(value)) return top;
  // look left if less-than, right if greater-than
  if (ordering.compare(topValue, value) < 0) {
    child = top.right();
  } else {
    child = top.left();
  }
  // no child there: not in tree, return this node,
  // else keep searching
  if (child.isEmpty()) { return top; }
  else { return locate(child, value); }
}
Adding to a BST

• How do we add elements to a BST?

```java
add

public void add(Object value) {
    BT newNode = new BT(value);
    BT node = locate(root, value);
    if (root.isEmpty) { root = newNode; }
    else {
        Object nodeValue = node.value;
        // node is successor or predecessor of newNode
        if (ordering.compare(nodeValue, value) < 0) {
            node.setRight(newNode);
        } else {
            if (!node.left.isEmpty()) {
                // if value is in tree, we insert before it predecessor(node).setRight(newNode);
            } else {
                node.setLeft(newNode);
            }
        }
        count++;
    }
}
```

Removal

• Removing the root is the hardest
• Let's figure that out first
  • If we figure out how to remove the root, we can remove any element in BST in same way (why?)

• We need to implement:
  • public Object remove(Object item)
  • protected BT removeTop(BT top)