Week 2

ASSIGNMENTS
h0 available and due before 10pm on Monday 1/28
h1 available and due before 10pm on Monday 2/4
p1 available and due before 10pm on Thursday 2/7

TA Lab Consulting - See schedule (cs400 home pages)
Peer Mentoring available - Friday 8am-12pm, 12:15-1:30pm in 1289CS

TAs and Peer Mentors will focus on helping students get p1 JUnit tests running in Lab. We can not guarantee that we can get your personal computers configured.

Module: Week 2 (start on week 3 before next week)

THIS WEEK

- Ready Set Program 1!
  - Read Assignment - there are getting started instructions there
  - Create and configure project for JUnit5, compile and run **TestDS_My**
  -
- Testing: JUnit5
- Java: inner classes
- Determining Height of a Tree (Recursion Review)
- Binary Search Trees (BST) (Review?)
  - operations
  - implementing
  - complexities
- Classifying Binary Trees
- Balanced Search Trees
- George Adelson-Velsky and Evgenii Landis

NEXT WEEK

- X-team Exercise x1 (in-class exercise with your assigned teams)
- Watch for instructions to find your team number and how to meet

Writing JUnit5 Tests

What is JUnit?

How do I use it?

What does a @test method look like?

Details:

1. **Import JUnit classes**

```
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

public class TestClass {

   @Test
   void test123_try_someting_check_results() {
      // try something
      // if (cond_expression) fail("descriptive failure message")
      // assertEquals(  expr1, expr2 )
      // fail if something unexpected occurs or if something expected does not occur.
   }

}
```

Java's Inner Classes

https://docs.oracle.com/javase/tutorial/java/javaOO/innerclasses.html

What is an inner class?

Define a generic Tree cass using an inner class for the individual node type of the tree.

```
public class Tree<K extends Comparable<K>> {

}
```

Determining Height of a Tree (or sub-tree)

height of a tree

We define height of a tree as the # nodes on path from the root to the deepest leaf.

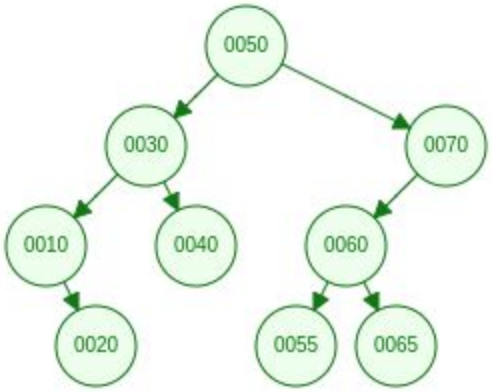
Write a recursive definition for the height of a general tree.

Complete the recursive height method based on the recursive definition.
Assume the method is added to a Tree class having a root instance variable.

```
public int height() {                                    }
```

Binary Search Tree (BST) Review

Image created at: https://www.cs.usfca.edu/~galles/visualization/BST.html



Insert 5, 27, 90, 73, 57 into the above BST tree (recall binary search algorithm)

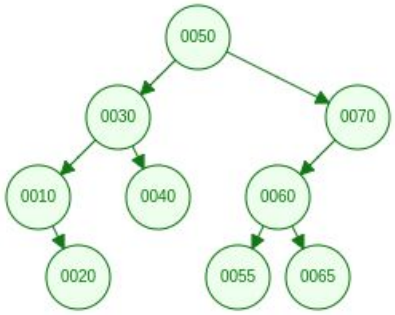
Insert the values 1,2,3,4,5 into an empty BST

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

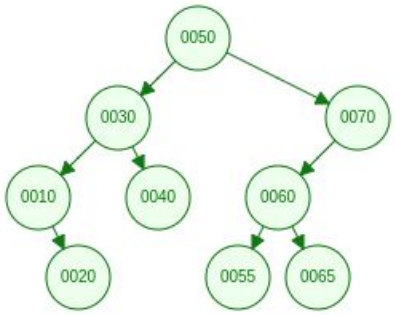
Will you only get this shape if inserted in sorted order?

Practice Deleting from a BST

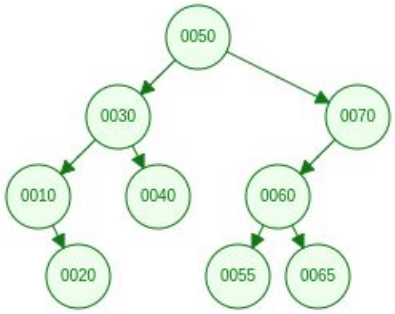
Delete 90 from this tree.
Delete 40 and then 65 from this tree.



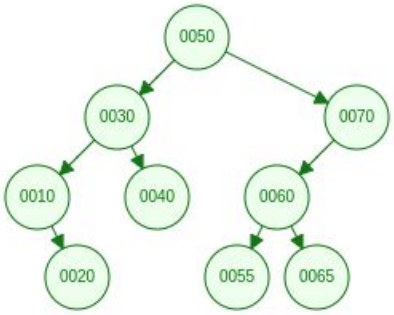
Delete 10 and then 70 from this tree.



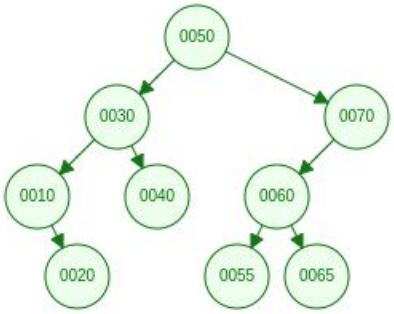
How do you delete 30 or 50 from this tree?



Delete 30 from this tree using the in-order predecessor



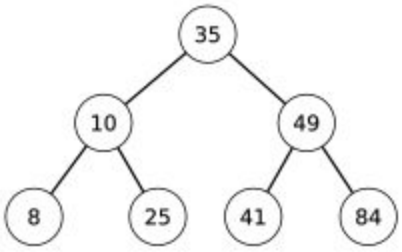
Delete 50 from this tree using the in-order successor



How do we find in-order predecessor or in-order successor?

Implementing BST Operations

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




```
  public insert(K key) { insert(root,key); }

  /** pre-cond: key is NOT null */
  private BSTnode<K> insert(BSTnode<K> n, K key)
          throws DuplicateKeyException {
```

Implementing BST Operations

delete

```
/** pre-cond: key is not null */
private BSTnode<K> delete(BSTnode<K> n, K key) {
```

CASE #1: n has no children

CASE #2: n has one child
// return n's other child to parent

CASE #3: n has two children

Complexities of BST Operations

Problem size: N =

print:

lookup:

insert:

delete:

k-ary Tree Classification Terms

Perfect: (referred to as full in some readings)

Complete:

Height-Balanced:

Balanced

Balanced Search Trees

Goal:

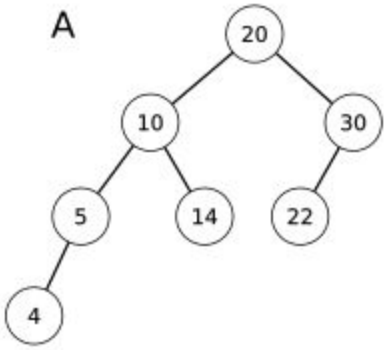
Idea:

How do we detect and fix?

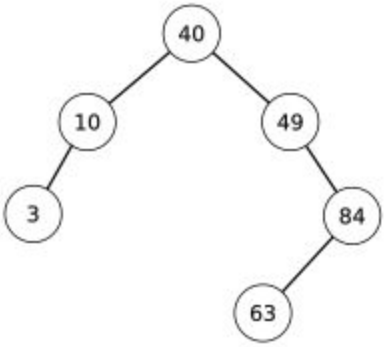
Balance Factor:

Height-Balanced Practice

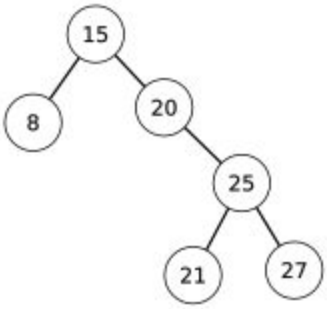
A



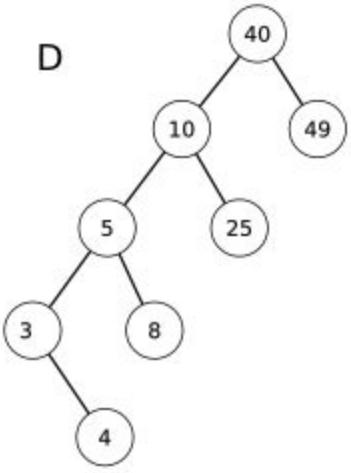
B



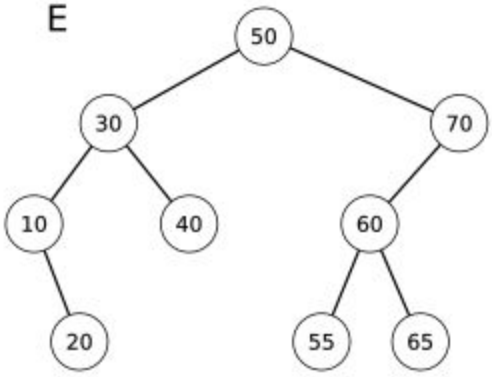
C



D



E



Balancing I - Practice

Draw BST for: 10,20,30

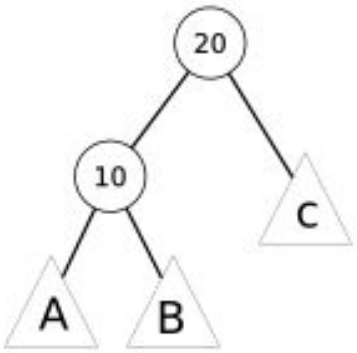
Draw BST for: 30,20,10

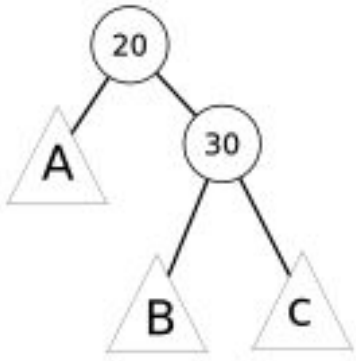
Draw BST for 10,30,20

Draw BST for 30,10,20

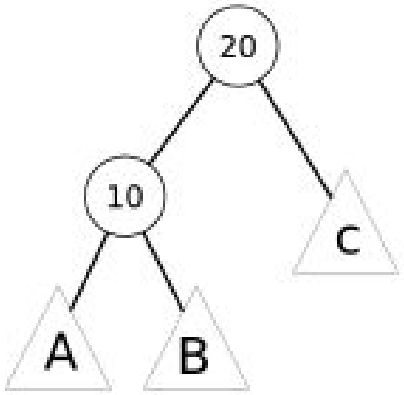
Balancing II - General Balancing

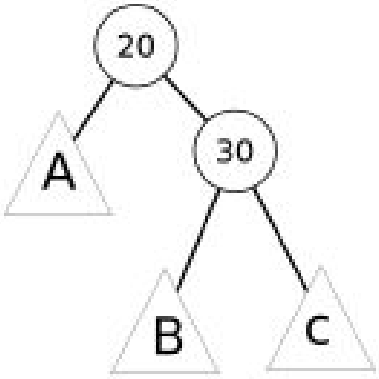
Assume: a triangle represents sub-tree of unknown height, but that trees shown are *height-balanced*, and show the relative difference (balance factor of -1, 0, 1)





Balancing III - More General Balancing





Georgy Adelson-Velsky and Evgenii Landis

Complexity of operations

Implementation

How can we know height of each sub-tree?