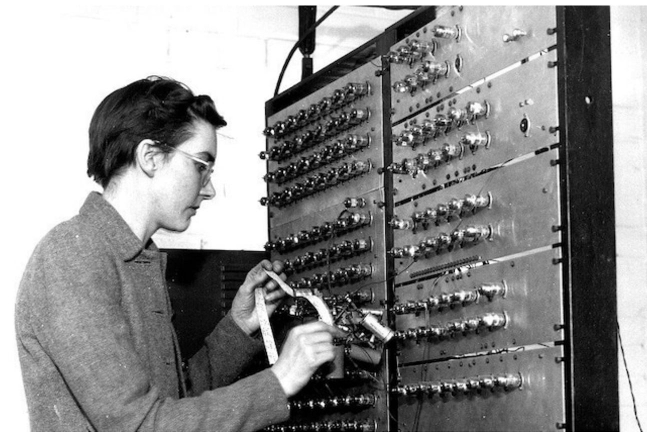
CompSci 201, L17: Tree Recursion

Person in CS: Kathleen Booth

- 1922 – 2022
- British Mathematician, PhD in 1950
- Worked to design the first *assembly language* for early computer designs in the 1950s
- May have been the first woman to write a book on programming
- Early interest in *neural networks*



Announcements, Coming up

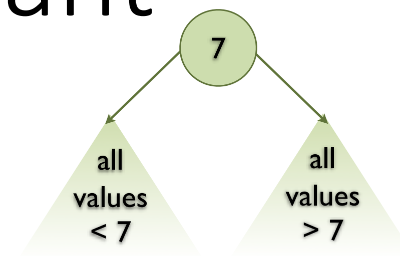
- Wednesday 3/22
  - Midterm 2, linked list through Monday's lecture
  - Practice exams available on Sakai resources
- Next Monday 3/27
  - Project P4: Autocomplete due
- Next Wednesday 3/29
  - APT 7 (tree recursion problems) due

Today's Agenda

1. Review/Wrap up binary search tree

2. Tree Recursion problems
   1. TreeCount
   2. HeightLabel
   3. Diameter

Binary Search Tree Invariant



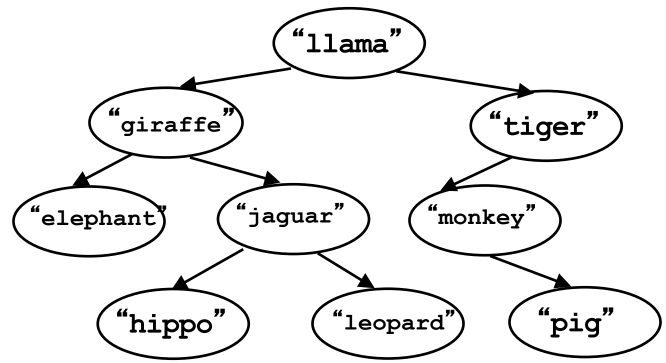
A binary tree is a binary **search** tree if *for every node*:

- Left subtree values are all less than the node's value

AND

- Right subtree values are all greater than the node's value

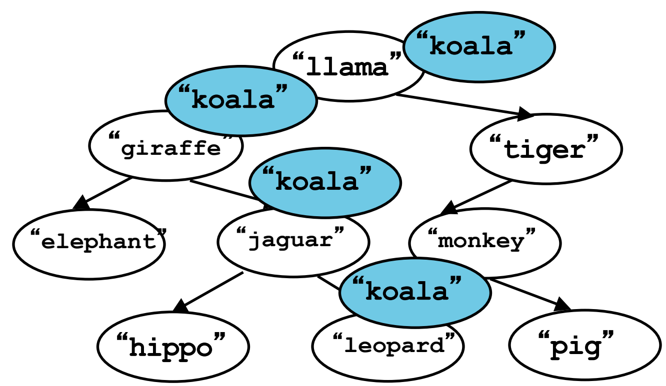
According to some ordering (comparable or comparator)



Enables efficient search, similar to binary search!

Iterative search in binary search tree

```java
// assumes node is a search tree, else may return false negatives
public static boolean contains(TreeNode<String> node, String target) {
    while (node != null) {
        int comp = node.info.compareTo(target);
        if (comp == 0) {
            return true;
        }
        else if(comp > 0) {
            node = node.left;
        }
        else {
            node = node.right;
        }
    }
    return false;
}
```



Again, insertion is very similar

DIY TreeSet

- See videos of live coding a DIYTreeSet as a binary search tree:
  - Part 1: Getting started, traversal, iterator
  - Part 2: add and contains

- And here is the code: coursework.cs.duke.edu/cs-201-spring-23/diytreeset

Tree Recursion and Problem-Solving

Tree Recursion tips / common mistakes

1. Draw it out! Trace your code on small examples.
2. Return type of the method. Do you need a helper method?
3. Base case first, otherwise infinite recursion / null pointer exception.
4. If you make a recursive call, make sure to use what it returns.

FAQ: Can I make a tree?

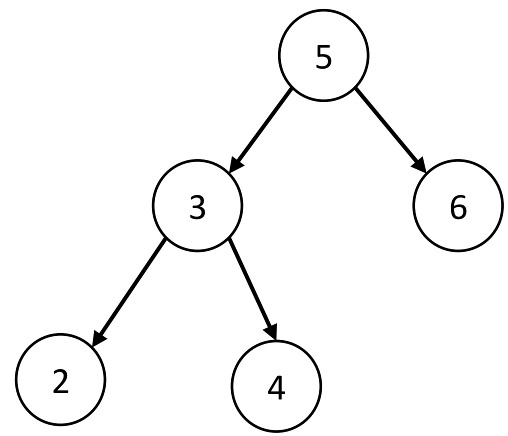
```
public class TreeNode {
    int info;
    TreeNode left;
    TreeNode right;
    TreeNode(int x){
        info = x;
    }
    TreeNode(int x, TreeNode lNode, TreeNode rNode){
        info = x;
        left = lNode;
        right = rNode;
    }
```

Just call the TreeNode constructor for each new node and connect them.

```
TreeNode root = new TreeNode( x: 5);
root.left = new TreeNode( x: 3);
root.right = new TreeNode( x: 6);
root.left.left = new TreeNode( x: 2);
root.left.right = new TreeNode( x: 4);
```

More terse version

```
TreeNode myTree = new TreeNode( x: 5,
        new TreeNode( x: 3,
                new TreeNode( x: 2),
                new TreeNode( x: 4)),
        new TreeNode( x: 6));
```

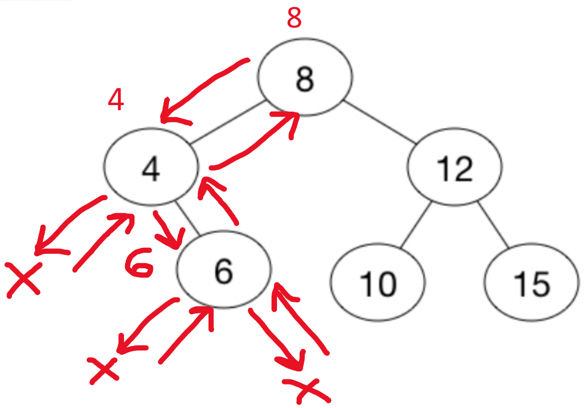


TreeCount APT and pre-order string representation

Problem Statement

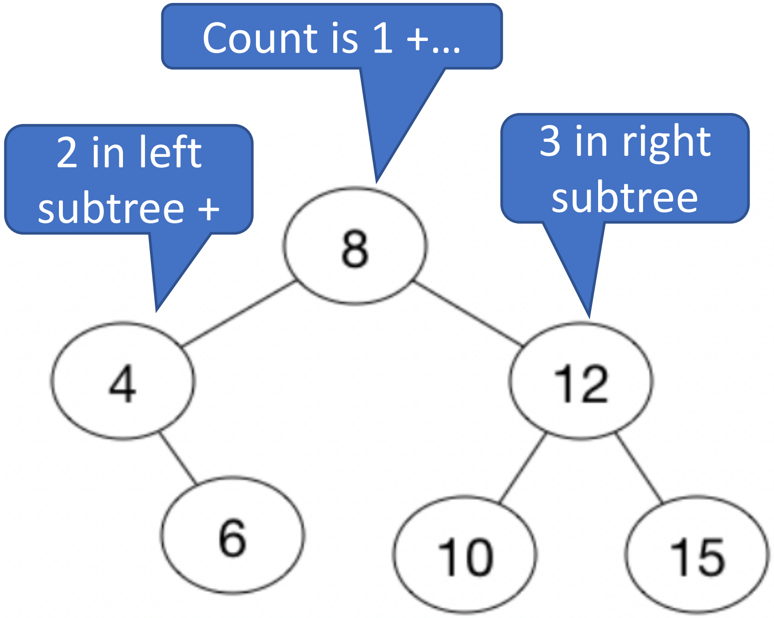
Write a method that returns the number of nodes of a binary tree. The TreeNode class will be accessible when your method is tested.

```
public class TreeCount {
    public int count(TreeNode tree) {
        // replace with working code
        return 0;
    }
}
```



is characterized by the pre-order string **8, 4, x, 6, x, x, 12, 10, x, x, 15, x, x**

Solving TreeCount in Picture & Code



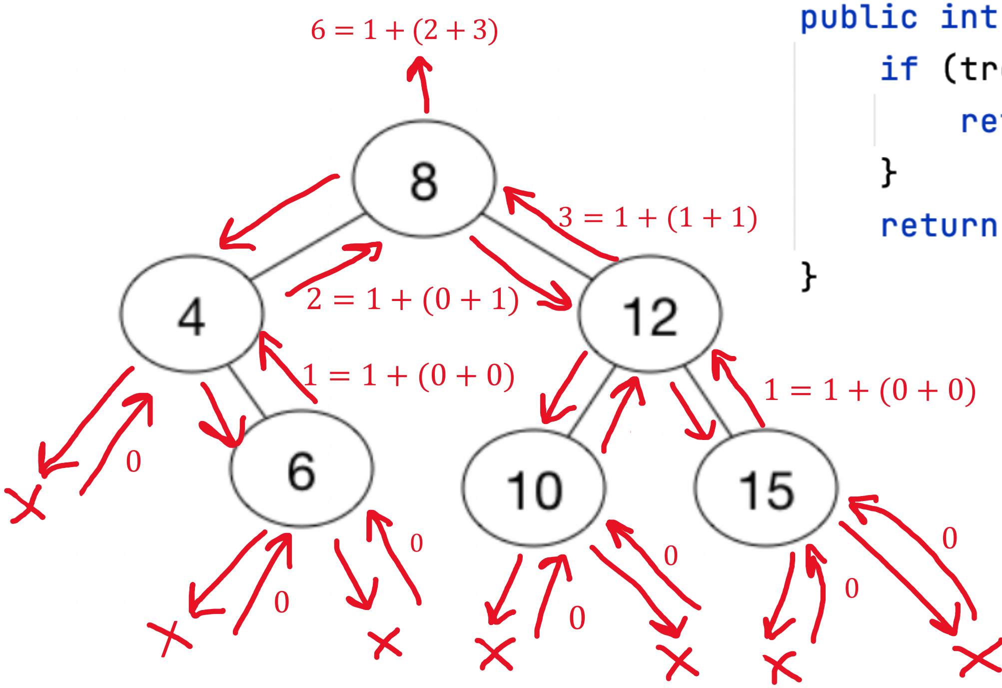
Base case: 0 nodes in an empty tree / null

Recursive case:

- 1 (count current node)
- + count of left subtree
- + count of right subtree

```java
public int count(TreeNode tree) {
    if (tree == null) {
        return 0;
    }
    return 1 + count(tree.left) + count(tree.right);
}
```

Messy Details of TreeCount Solution




```java
public int count(TreeNode tree) {
    if (tree == null) {
        return 0;
    }
    return 1 + count(tree.left) + count(tree.right);
}
```

Analyzing Recursive Runtime

Develop a recurrence relation of the form

$$T(N) = a \cdot T\big(g(N)\big) + f(N)$$

Total runtime

Recursive call(s)

Non-recursive runtime

Where:

- $T(N)$ - runtime of method with input size *N*
- a is the number of recursive calls
- $g(N)$ - how much input size decreases on each recursive call
- $f(N)$ - runtime of non-recursive code on input size N

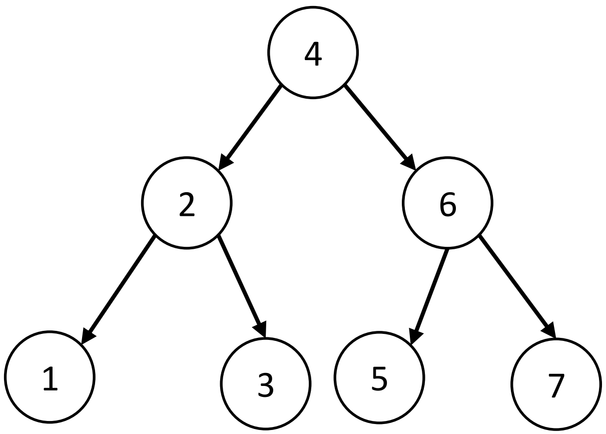
Table of Recurrences

Recurrence	Algorithm	Solution
`T(n) = T(n/2) + O(1)`	binary search	`O(log n)`
`T(n) = T(n-1) + O(1)`	sequential search	`O(n)`
`T(n) = 2T(n/2) + O(1)`	tree traversal	`O(n)`
`T(n) = T(n/2) + O(n)`	qsort partition ,find k^{th}	`O(n)`
`T(n) = 2T(n/2) + O(n)`	mergesort, quicksort	`O(n log n)`
`T(n) = T(n-1) + O(n)`	selection or bubble sort	$O(n^2)$

We expect you to be able to derive a recurrence relation from an algorithm, but not necessarily to solve. We will provide a table of solutions like this for exams.

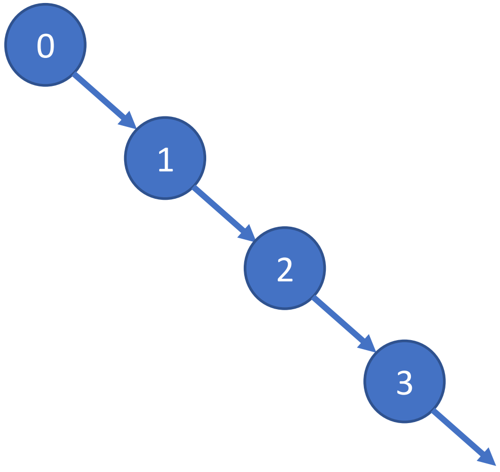
Balance and Trees

Balanced



Left and right subtrees have roughly equal number of nodes.

Unbalanced



One subtree has many more nodes than the other.

Recurrence relation and runtime for traversing a **balanced** tree

- ***T(n)*** time **`count(tree)`** with ***n*** nodes (balanced)

```java
public int count(TreeNode tree) {
    if (tree == null) {
        return 0;
    }
    return 1 + count(tree.left) + count(tree.right);
}
```

n/2 nodes in this subtree

n/2 nodes in this subtree

- $T(n) = 2T(n/2) + O(1)$
- $= O(n)$

Recurrence relation and runtime for traversing **unbalanced** tree

- ***T(n)*** time **`count(tree)`** with ***n*** nodes (unbalanced)

```java
public int count(TreeNode tree) {
    if (tree == null) {
        return 0;
    }
    return 1 + count(tree.left) + count(tree.right);
}
```

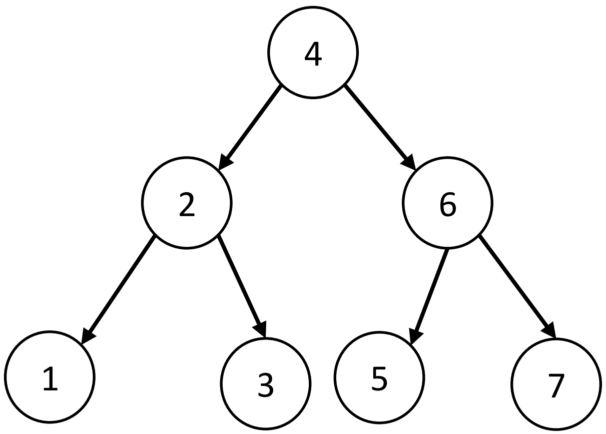
1 node in this subtree

n-1 nodes in this subtree

- T(n) = T(1) + T(n-1) + O(1)
- = O(n)

Balance Binary Search Tree Runtime (add, contains)

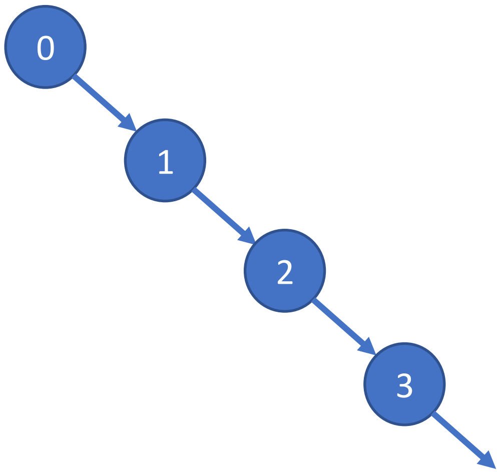
Balanced



$T(n) = T(n/2) + O(1)$
$= O(\log(n))$

We will return to this problem later!

Unbalanced

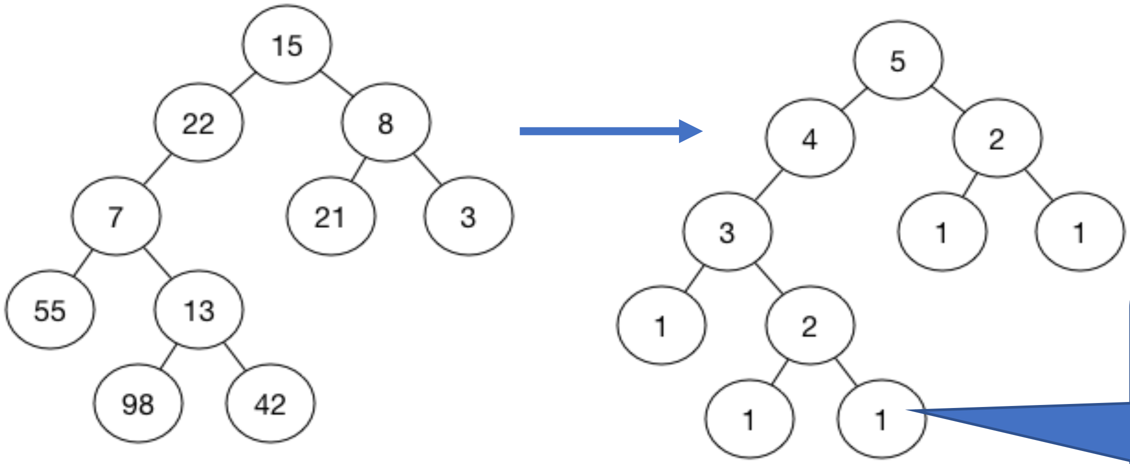


$T(n) = T(n-1) + O(1)$
$= O(n)$

HeightLabel APT

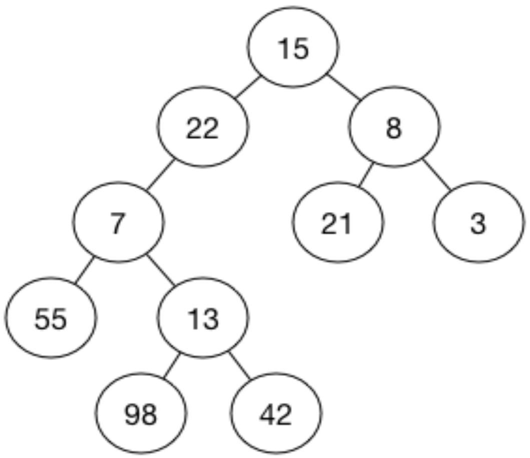
https://www2.cs.duke.edu/csed/newapt/heightlabel.html

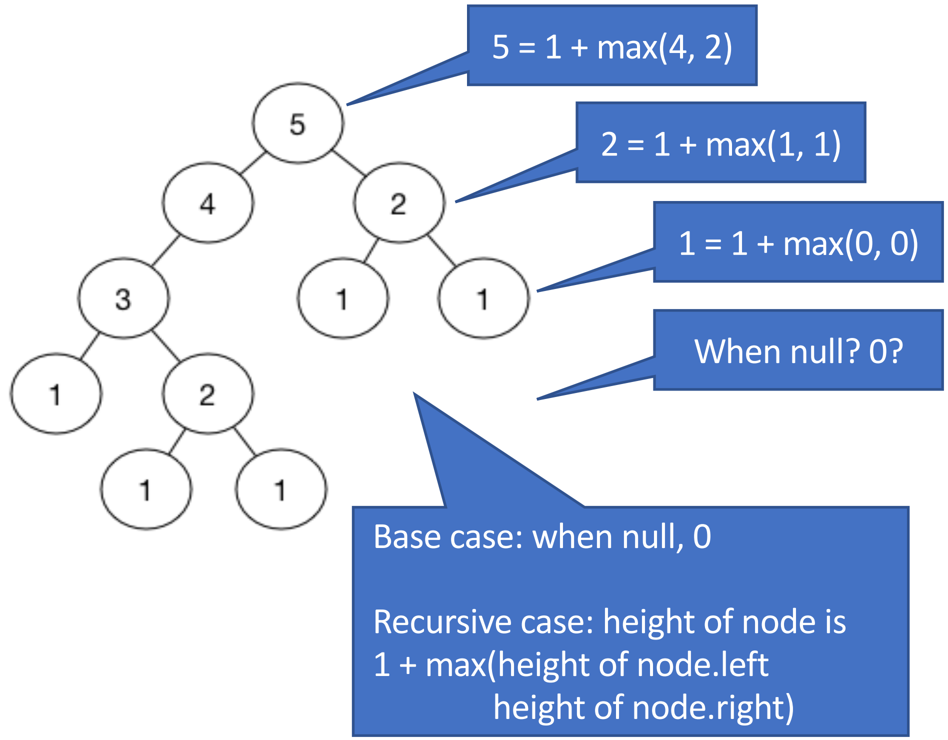
- Create a new tree from a tree parameter
  - Same shape, nodes labeled with height
  - Use **`new TreeNode`**. With what values …



Note that this APT 1-indexes height/depth. We introduced it 0-indexed.

Solving HeightLabel in Pictures





Solving HeightLabel in Code

```
private int height(TreeNode t) {
     if  (t == null) return 0;
     return 1 + Math.max(height(t.left),
                         height(t.right));
}
```

Base case: when null, 0

Recursive case: height of node is 1 + height of node.left + heigh of node.right

```
public class HeightLabel {
    public TreeNode rewire(TreeNode t) {
        // replace with working code
        return null;
    }
}
```

Method doesn’t just calculate height, is supposed to create and return new tree with new nodes…

```
public TreeNode rewire(TreeNode t)  {
     if (t == null) return null;
      return new TreeNode(height(t),
                          rewire(t.left),
                          rewire(t.right));
}
```

Using height helper method, get height, create new node, return.

Rewire runtime?

- recurrence of this all-green code? T(n) =
- **`2T(n/2)+O(n)`**
  - Balanced tree
- **`T(n-1)+O(n)`**
  - Unbalanced

T(n)

```
public TreeNode rewire(TreeNode t) {
     if (t == null) return null;
      return new TreeNode(height(t),
                    rewire(t.left),
                    rewire(t.right));
}
```

O(n)

T(n/2) if balanced

T(n/2) if balanced

```
private int height(TreeNode t) {
     if  (t == null) return 0;
     return 1 + Math.max(height(t.left),
                    height(t.right));
}
```

HeightLabel Complexity

- Balanced? `O(N log N)`,
  - `2T(n/2) + O(n)`
- Unbalanced, $O(N^2)$,
  - `T(N) = T(N-1) + O(N)`
- Do in `O(N)` time? Yes, if we don't call height
  - Balanced: `T(N) = 2T(N/2) + O(1)`
  - Unbalanced: `T(N) = T(N-1) + O(1)`

HeightLabel in O(N) time

- If recursion works, subtrees store heights!
- Balanced? **O(N)** ,
  - **2T(n/2)+O(1)**
- Unbalanced, **O(N)**,
  - **T(N-1)+O(1)**

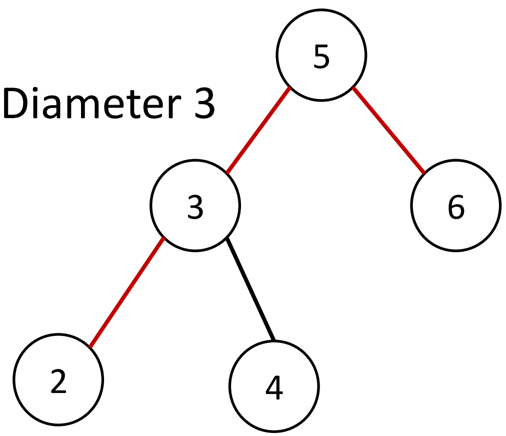
```java
public TreeNode rewire(TreeNode t) {
    if (t == null) { return null; }
    TreeNode leftOfMe = rewire(t.left);
    TreeNode rightOfMe = rewire(t.right);
    int lHeight = 0;
    int rHeight = 0;
    if (leftOfMe != null) { lHeight = leftOfMe.info; }
    if (rightOfMe != null) { rHeight = rightOfMe.info; }
    return new TreeNode(
            x: 1+Math.max(lHeight, rHeight),
            leftOfMe,
            rightOfMe);
}
```

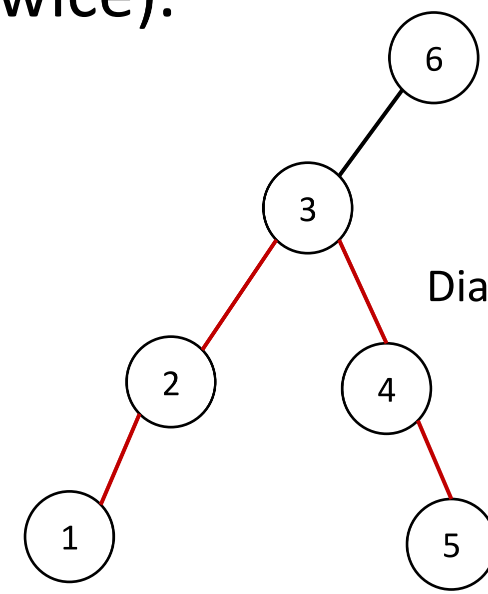
Diameter Problem

leetcode.com/problems/diameter-of-binary-tree

Calculate the *diameter* of a binary tree, the length of the longest path (maybe through root, maybe not, can't visit any node twice).

Diameter 3





Diameter 4

Live Coding