Binary trees

# A binary tree is a tree in which each node has at most two children. The first tree on the right is a binary tree. It has nodes with two children, one child, and 0 children. The second tree is not a binary tree because its root has three children.

# In a binary tree, the children are called the left child and the right child.

# Binary trees have lots of applications. Here’s an example of a binary tree that is on the internet. The website <https://genealogy.math.ndsu.nodak.edu> maintains the PhD genealogy of almost 237,500 PhDs in math and CS. On this site, a PhD can have up to two advisors, so the tree of advisors of a PhD is a binary tree.

# To the right, we show the first three levels of the advisor tree for George Forsythe, the first chair of CS at Stanford, beginning in 1965. George worked in the relatively new field of *numerical analysis*. At the time of his move from the Math Dept. to the new CS Dept., he quipped that, “Many numerical analysts have progressed from being queer people in math departments to queer people in CS departments." If you stay in CS or Math, you will quite likely see his name and the names of his intellectual grandparents, Markov and Courant, again.

# Here are some facts about binary trees.

1. **Minimum number of nodes in a binary tree of height h** is h+1. For example, the tree to the right has height 2 and 3 nodes. A tree with the minimum number of nodes will have one node on each level.
2. **The maximum number of nodes at depth d** is 2d. Check out the tree to the right. You can see that:

* The number of nodes on depth 0 is 20 = 1, the root.
* At each level, the number of nodes is twice that on the previous level because each node has two children.

1. **Maximum number of nodes in a binary tree of height h** is 20 + 21 + … + 2h = 2h+1 – 1.

The formula 20 + 21 + … + 2h for the maximum number of nodes is a direct result of the previous point 2. As an example, for the perfect binary tree above, the number of nodes is 20 + 21 + 22 = 7. For a proof of 20 + 21 + … + 2h = 2h+1 – 1, see JavaHyperText entry *binary tree* .

1. **Height of a balanced binary tree**. A binary tree is balanced if for each node, the heights of its left and right subtrees differ by at most 1. The height of a balanced binary tree with n nodes is O(log n). (For a proof see JavaHyperText entry *binary tree*.

**Java implementation of a binary tree**

**public class** TreeNode<T> {

**private** T datum;

**private** TreeNode<T> left; // left subtree (null if empty)

**private** TreeNode<T**>** right; // right subtree (null if empty)

/\*\* Constructor: one-node tree with datum d \*/

**public** TreeNode (T d) { datum= d; }

/\*\* Constr: Tree with root datum d, left tree l, right tree r \*/  
 **public** TreeNode (T d, TreeNode<T> le, TreeNode<T> r) {

datum= d; left= le; right= r;

}

}

# To the right is the start of class TreeNode, which implements a node of a binary tree and contains a value of generic type T. It needs only three fields: the field that contains a value, the left subtree, and the right subtree.

# Two constructors are provided for flexibility. The first creates a one-node binary tree; the second creates a root with two given subtrees.

# Naturally, this class has more methods —getters for the fields, a toString method, and so on. There is no need to describe them here.

**Recursion on binary trees**