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, who was 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. We leave to you the proof that this formula equals 2h+1 – 1.

1. **Height of a balanced binary tree**. A binary tree is balanced if for each node, the heights of the left and right subtrees of that node 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 *balanced tree* .

**Java implementation of a binary tree**

**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> l, TreeNode<T> r) {

datum= d; left= l; right= r;

}

}

**Tree terminology**

You now know that a tree consists of a bunch of *nodes*, some of which are connected by *edges*. In the tree shown below, for example, nodes A and D are connected by an edge, while nodes A and F are not connected by an edge. The diagram below also describes the notion of a *parent* of a node, a *child* of a node, and a *sibling* of a node. Based on that, the definitions of *ancestor* and *descendent* are obvious from their English meaning.

**ancestor**: parent, parent’s parent, parent’s parent’s parent, etc.

**descendent** or **descendant**: child, child’s child, child’s child’s child, etc.

J’s **ancestors** are D and A.

C’s **descendents** are F, H, and I



A is **parent** of B, C, D

J is **child** of D

I and H are **siblings**

With those definitions, we define the root of a tree, the leaves of a tree, and the internal nodes of a tree:



**root** of tree (no parent)

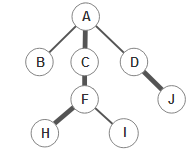
**internal node**: has a child

internal nodes are A, B, C, D, F

**leaf** of tree (no child)

The *size* of a tree is the number of nodes in it.

The *degree* of a node is its number of children. Thus, a leaf has degree 0.

**Paths**

A (downward) *path* is a sequence of nodes and edges that leads from the first node down to one of its descendents (or itself). We describe a path by giving the sequence of nodes in it. For example, in the tree to the right, we have highlighted two paths using thick edges: the path (D, J) and the path (A, C, F, H).

The *length* of the path is the number of edges in it. Note: (B) describes the path of length 0 from B to B; it contains no edges.

At times, one *does* talk about not only downward paths but paths from any node to any  
other node. For example, we could consider the path from node F to node B: (F, C, A, B).  
But for now, we will talk only about downward paths.

**Depth, level, height, and width**

level depth

1 0

2 1

3 2

4 3

The *depth* of a node is the length of the path from the root to the node. The *level* of the node is 1 + (its depth). Sorry, this can be confusing, but that’s the terminology. We won’t be using the *level*.

The height of a node is the length of a longest path from that node to a leaf. The height of a tree is the height of its root.

height of B, H, I, J: 0

height of F, D: 1

height of C: 2

height of A: 3

height of tree: 3

The width of a tree at depth d is the number of nodes at depth d.   
The width of a tree is its maximum width over all depths.

depth width at depth

0 1

1 3

2 2

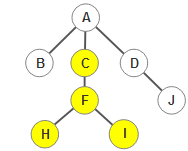
3 2

In the tree to the right, the width at depth 1 is 3 —there are three nodes at that depth: B, C, and D. That is the maximum width over all depths, so the width of the tree is 3.

**Forest**

A *forest* is just a set of 0 or more disjoint trees. By disjoint we mean that no two trees in the forest have a node in common.

**Subtrees**

Look at the first tree to the right. We can think of C simply as a node. It may contain values of some sort, and it has parent A and child F.

But we can also think of *subtree C*: That is, the tree whose root is C, as shown in yellow in the second tree to the right.

So, we have two views of C: it’s simply a node, or it’s the root of a subtree. Get used to these two ways of thinking of a node of a tree.