**Trees**

Formal Tree Definition and some concepts

Formally, we define a tree T as a set of nodes storing elements such that the nodes have a parent-child relationship that satisfies the following properties:

* If T is nonempty, it has a special node, called the ***root*** of T, that has no parent.
* Each node v of T different from the root has a unique parent node w; every node with parent w is a child of w.

Note that according to our definition, a tree can be empty, meaning that it does not have any nodes.

This convention also allows us to define a tree recursively such that a tree T is either empty or consists of a node r, called the root of T, and a (possibly empty) set of subtrees whose roots are the children of r.

Two nodes that are children of the same parent are ***siblings***. A node v is ***external*** if v has no children. A node v is ***internal*** if it has one or more children. External nodes are also known as ***leaves***.

A node u is an ***ancestor*** of a node v if u = v or u is an ancestor of the parent of v.

Conversely, we say that a node v is a descendant of a node u if u is an ancestor of v.

The ***subtree*** of T rooted at a node v is the tree consisting of all the descendants of v in T (including v itself).

**Edges and Paths in Trees**

An **edge** of tree T is a pair of nodes (u,v) such that u is the parent of v, or vice versa. A path of T is a sequence of nodes such that any two consecutive nodes in the sequence form an edge.

A tree is ***ordered*** if there is a meaningful linear order among the children of each node; that is, we purposefully identify the children of a node as being the first, second, third, and so on. If order is unimportant, the tree is ***oriented***. A ***forest*** is a set of zero or more disjoint trees.

**Computing Depth and Height**

Let p be a position within tree T. The depth of p is the number of ancestors of p, other than p itself.

The depth of p can also be recursively defined as follows:

* If p is the root, then the depth of p is 0.
* Otherwise, the depth of p is one plus the depth of the parent of p.

The level (or depth) of a node is the number of branches that must be traversed on the path to the node from the root. The root has level 0.

The running time of depth(p) for position p is O(dp +1), where dp denotes the depth of p in the tree, because the algorithm performs a constant-time recursive step for each ancestor of p.

Thus, algorithm depth(p) runs in O(n) worst-case time, where n is the total number of positions of T, because a position of T may have depth n−1 if all nodes form a single branch. Although such a running time is a function of the input size, it is more informative to characterize the running time in terms of the parameter dp, as this parameter may be much smaller than n.

**Height:** The height of a tree to be equal to the maximum of the depths of its positions (or zero, if the tree is empty).

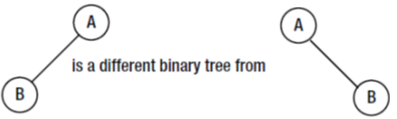
The ***moment*** of a tree is the number of nodes in the tree. The ***weight*** of a tree is the number of leaves in the tree.

If the relative order of the subtrees T1, T2, …, Tm is important, the tree is an ordered tree.

The ***degree*** of a node is the number of subtrees of the node.

Binary Trees

A binary tree is empty or consists of a root and two subtrees—a left and a right—with each subtree being a binary tree. A consequence of this definition is that a node always has two subtrees, any of which may be empty. Another consequence is that if a node has one nonempty subtree, it is important to distinguish whether it is on the left or right. Here’s an example:



A binary tree where each node, except the leaves, has exactly two subtrees is called a ***complete*** binary tree