1. Trees :: Introduction and Definitions

A tree is a data structure that consists of nodes, with each node storing data and zero or more **edges** linking the node to other nodes. Although this may sound similar to a linked list, a tree is a hierarchical data structure with nodes being designated **parent** and **child** nodes. A node may simultaneously be both a parent (i.e., have child nodes) and a child (i.e., be the child node of a parent).

The node at the top of the hierarchy is designated the **root** node of the tree and is the only node with no parent.

1. Trees :: Introduction and Definitions (continued)

Child nodes with the same parent are called **siblings**.

A **path** is a series of contiguous edges that connects one node to another node.

The **ancestors** of a node are all of the nodes that are on the path connecting the node to the root, i.e., the parent of the node, the parent of the parent (a **grandparent node**), the parent of the parent of the parent, and so on, all the way up to the root.

The **descendants** of a node are the the children of the node, the children of the children, and so on.

A node with no children is called a **leaf node** (or **external node**).

A node with children is called an **internal node**.

A **subtree** is a tree consisting of a node and all of its descendants; the node is the root of the subtree.

An **ordered tree** is a tree such that all of the children of an internal node are somehow ordered.

The **level** of a node is the length of the unique path from the node to the root of the tree. The root is at level 0.

The **height** of a tree is the maximum of the levels of each node, i.e., it is the length of the longest path from the root to a leaf node.