Trees

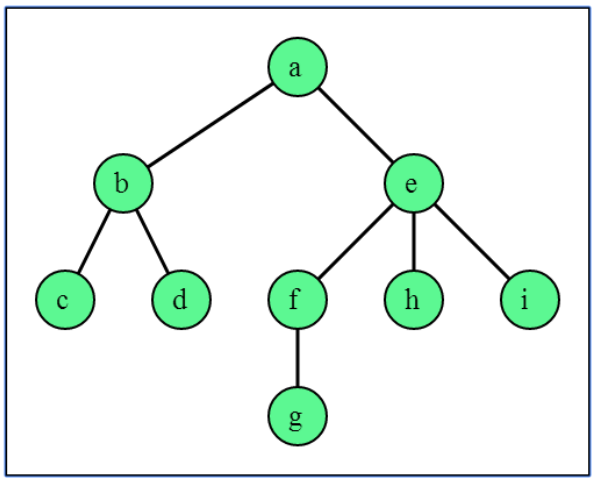
Help Center

A rooted tree is a hierarchical data structure with applications in many areas of computer science. These applications include searching, sorting, modeling biological processes (such as genealogy), and representing hierarchical programming constructs such as arithmetic expressions.

Basic definition and display

Formally, a rooted tree is a collection of nodes and edges that can be organized recursively as follows. The tree has a *root* node with an associated value and a list of references to a collection of *subtrees*. The root nodes of these subtrees are the *children* of the root node for the original tree while the root node is the *parent* of the children. Each node in the tree should have exactly one parent with the exception of the root which has no parent. This last condition ensures that the tree is hierarchical since each subtree of the root is then guaranteed to be disjoint from the other subtrees.

The image below shows a diagram of a typical rooted tree. In this diagram, each node is a green circle labeled by its value and edges are drawn connecting each parent with its children. Note that a tree can also be viewed as a special type of graph. In graph theory, an *unrooted* tree is graph with n nodes and $n - 1$ edges, with the property that any pair of nodes is connected by a sequence of edges. (For this class, we will henceforth consider only rooted trees.)



This example shows a tree with nodes labeled `"a"` to `"i"`. The tree consists of a root node (labeled with its value `"a"`) and two subtrees consisting of the nodes `"b"`, `"c"`, `"d"` and `"e"`, `"f"`, `"g"`, `"h"`, `"i"`, respectively. Observe that rooted trees are typically drawn with the root at the top of the diagram, and with subtrees drawn below the root in left-to-right order.

Structural properties of trees

The nodes of a tree are typically classified based on their number of children. A node with no children is a *leaf* node. In the tree above, the nodes labeled `"c"`, `"d"`, `"g"`, `"h"` and `"i"` are leaves. Nodes with one or more children are *internal* (or interior) nodes. In the example, the nodes labeled `"a"`, `"b"`, `"e"`, and `"f"` are internal nodes.

This distinction between leaf nodes and internal nodes allows us to specify simple recursive definitions for several important structural quantities associated with trees.

- The number of nodes in a tree satisfies:

```
If the root node of the tree is a leaf:
    return 1
else:
    return 1 + the sum of the number of nodes for each subtree
```

- The number of leaves in a tree satisfies:

```
If the root node of the tree is a leaf:
    return 1
else:
    return the sum of the number of leaves for each subtree
```

- The *height* of a tree is the length of the longest sequence of edges from the root to a leaf. The height can be computed recursively via:

```
If the root node of the tree is a leaf:
    return 0
else:
    return 1 + the maximum of the heights of the subtrees
```

In general, the nodes of a tree can have any number of children including just a single child (such as the node labeled `"f"` above). One particularly important class of trees are *binary* trees in which all nodes have two or fewer children. Binary trees in which all internal nodes have exactly two nodes are called *full* binary trees. Full binary trees often arise in applications involving searching and sorting.

For a more concrete illustration of how trees are represented and how these tree methods are implemented, please examine our implementation of a basic `Tree` class in Python.

Created Tue 8 Jul 2014 4:54 AM CST

Last Modified Thu 16 Apr 2015 9:23 AM CST