Introduction to Trees

We often write a list of values 6, 7, and 3 like this: (6, 7, 3). The list could also be depicted as shown to the right, with each element having a pointer to its successor in the list.

If each element can have 0 or more successors, as shown to the right, we have what we call a tree. Here, the first element, 6, has three successors: 2, 7, and 8. The first element’s successor 2 has no successors. The first element’s successor 7 has one successor.

Thus, a list is a special form of a tree.

To the right, we draw a tree as computer scientists generally draw trees. No arrow heads are used; it is assumed that all lines point down. And, think of elements as nodes, with edges—the lines—connecting certain nodes. For now, think of the letters not as values but as names of the nodes. Node A at the top is called the root of the tree, and nodes B, H, I, and J, which have no successors, are called leaves. Computer scientists generally draw trees upside down!

It’s important to realize we are not discussing a data structure. Remember: a data structure is an organization or format for storing and managing data, usually to make some operations efficient. Here, we’re not showing how to implement trees, or how to store them in a data structure. We’re just introducing type tree—without any operations at the moment.

These are not trees

The first image to the right is not a tree. A tree cannot have a node that has an edge leading from itself back to itself. The second image is also not a tree because node C has two parents, H and I. A node can have at most one parent, and only the root node can have no parents.

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.

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

©David Gries, 2018
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

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. By definition, an empty tree has height -1.

The height of a tree is the height of its root.

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.

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.