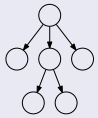
An Introduction to Trees

Prof. Susan Older

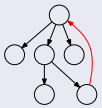
23 October 2019

Trees: An Ubiquitous Data Structure in Computing

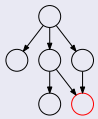
Tree



Not a tree



Not a tree



Basic idea:

A tree is a collection of linked nodes such that:

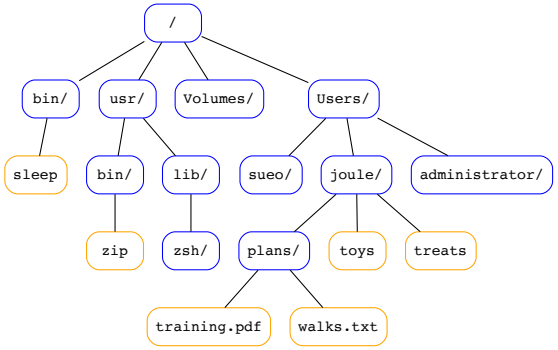
1. There are no cycles: there is no sequence of links that connects a node to itself.
2. If there is a link (called an edge or branch) from node A to node B, then:

   A is the parent of B, and B is the child of A.
3. Each node has 0 or more children and at most one parent.
4. If the tree has at least one node, then exactly one node (called the root) has no parent.

Depending upon context, the nodes and/or edges may contain values, labels, or conditions.

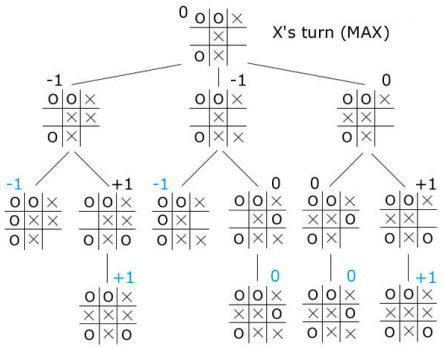
The arrows on branches are often omitted.

Tree Example: Unix File System Hierarchy



- The nodes are labeled by file/directory names.
- The node labeled joule/ has three children; the node labeled toys has no children.
- The node labeled / is the root node.

Tree Example: Game Tree



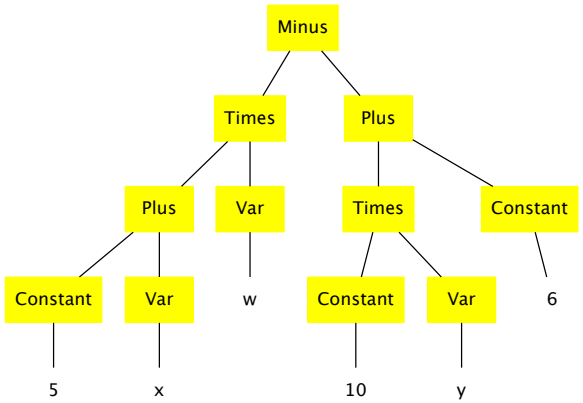
This image is from Yosen Lin's discussion of game trees at `https://www.ocf.berkeley.edu/~yosenl/extras/alphabeta/alphabeta.html`.

- Nodes are labeled by game states, combined with a numeric "favorability" rating.
- The children of a node represent possible "next" game states.

Tree Example: Abstract Syntax

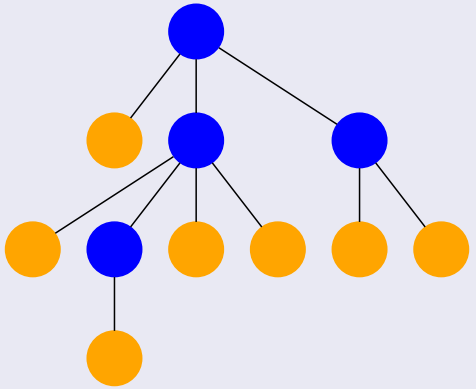
An internal (programmer-friendly) representation of the expression:

```
((5+x)*w) - ((10*y) + 6)
```



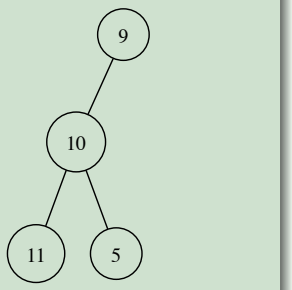
More Terminology: Leaves and Interior Nodes

- A leaf is a node with no children.
- An interior node is a node with at least one child.

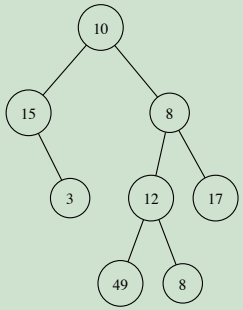


Binary Trees

A binary tree is a tree in which every node has **at most two children**. The children are often ordered (i.e., specified as left or right).



The node with label 9 has no right child.



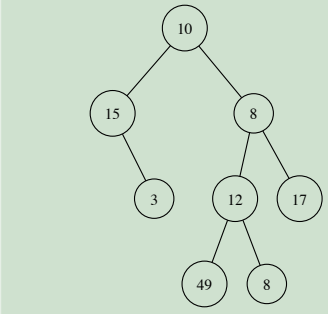
The node with label 15 has no left child.

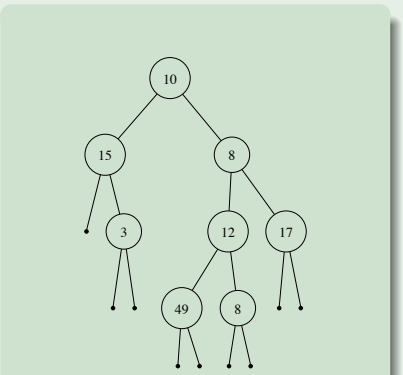
Binary Trees: Towards an Implementation

One way to define binary trees recursively:

A binary tree is one of the following:

- An "empty" tree (i.e., no nodes)
- A node that has two children, both of which are binary trees





Binary Trees: A Haskell Implementation

One way to define binary trees recursively:

A binary tree is one of the following:

- An "empty" tree (i.e., no nodes)
- A node that has two children, both of which are binary trees

```
data BTree a = Empty
             | BNode a (BTree a) (BTree a)
               deriving (Show)
```

Nodes are labeled by values of type a.

Let's Look at the Datatype Definition More Closely

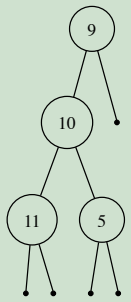
```
data BTree a = Empty
             | BNode a (BTree a) (BTree a)
               deriving (Show)
```

This definition has the following effects:

- It creates a family of types of form `BTree a`
  - `BTree Float`
  - `BTree (Int,Char)`
  - `BTree [Bool]`
  - `BTree (Char -> Bool)`
  - `BTree (BTree Int)`
  - *and the list goes on...*
- Creates a (polymorphic!) value `Empty :: BTree a`
- Creates a (polymorphic!) constructor/function `BNode :: a -> BTree a -> BTree a -> BTree a`
- Adds each `BTree a` type to the `Show` class

Haskell Binary Tree Example

```
data BTree a = Empty
             | BNode a (BTree a) (BTree a)
               deriving (Show)
```




```
BNode 9
      (BNode 10
             (BNode 11 Empty Empty)
             (BNode 5 Empty Empty))
      Empty
```