Chapter 13-[Trees](https://mfleck.cs.illinois.edu/building-blocks/version-1.3/trees.pdf)

Thursday, January 5, 2023

2:15 PM

***Trees:***

Trees are the central structure for storing and organizing data in computer science. (They're also good for the environment)

A ***Tree*** is a undirected graph with a special node called the *root*, in which every node is connected to the root by **exactly one** path. When a pair of nodes are neighbors in the graph, the node nearest the root is called the *parent* and the other node is its *child.*

A ***Leaf Node*** is a node that has no children. A node that does have children is known as an ***Internal Node***.

***Levels*** are based on how many edges away from the root nodes are.

(The root is defined to be level 0)

The ***Height*** of a tree is the maximum level of any of its nodes or, the maximum level of any of its leaves or, the maximum length of a path from the root to a leaf.

If you can get from x to y by following zero or more parent links, then y is an ancestor of x and x is a ***descendent*** of y. So x is an ancestor/descendent of itself. The ancestors/descendents of x other than itself are its ***proper ancestors/descendents***. If you pick some random node z in a tree T, the ***subtree rooted at*** z consists of z (its root), all of z’s descendents, and all the edges linking these nodes.

*A binary tree allows each node to have at most two children. An m-ary tree allows each node to have up to m children.*

In a **full** m-ary tree, each node has either zero or m children. Never an intermediate number.

In a ***complete m-ary tree***, all leaves are at the same height.

***Balanced binary trees*** are binary trees in which all leaves are at approximately the same height.

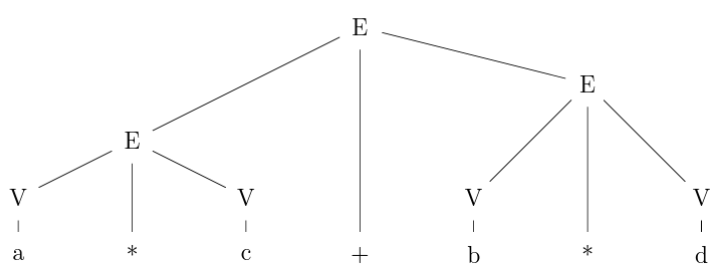
(Height is proportional to



)

***Parse Trees*** are trees that show the structure of a sequence of word/characters/symbols(***Terminal Symbols***).

Parse Tree example: (a\*c) + (b\*d)

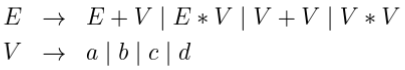


*(a, \*, C, +, b, d are all terminal symbols)*

This kind of labeling is called ***Context-free Grammar.***

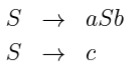
The ***Terminal Sequence*** is ***Generated*** by a specific grammar/rule.

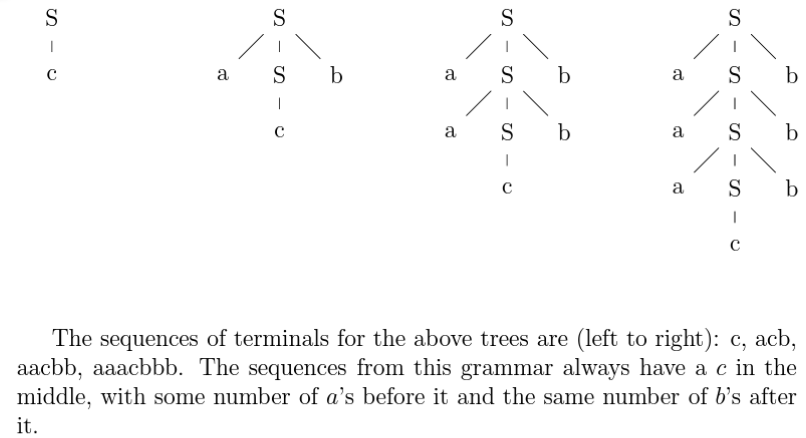
*(with the above example, a\*c + b\*d is the Terminal Sequence)*



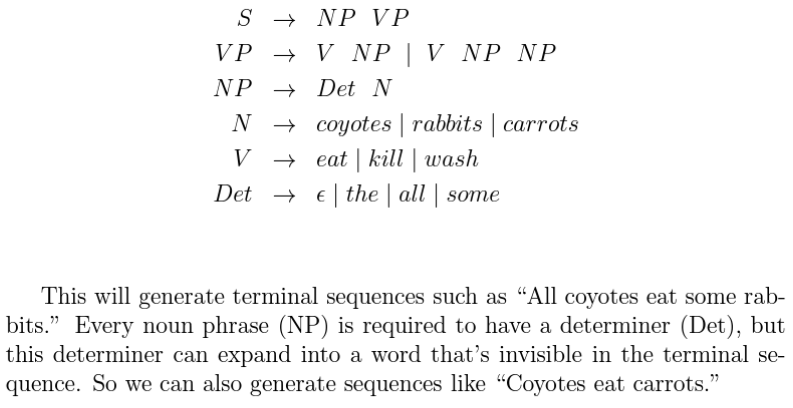
This is how you define a grammar, E and V are the ***Start Symbols***, which are allowed to appear on the root node. The nodes that are at the lower end of a path (the lowest row) are called the ***Terminals***.

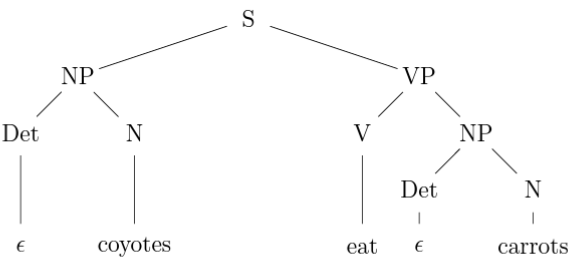
Another Example, S:





Engrish grammar example:

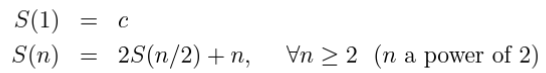




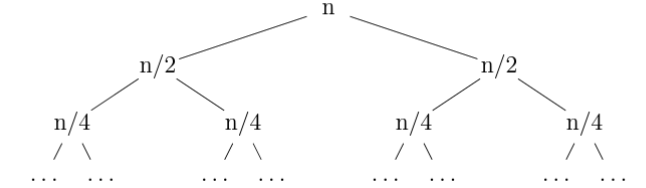
***Recursion Trees:***

The behavior of recursive functions can be written out as trees.

Example:



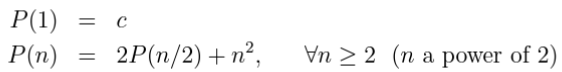
(The top node relative to each 2 rows represents S(n) and contains everything in the formula for S(n) **except the recursive part**)

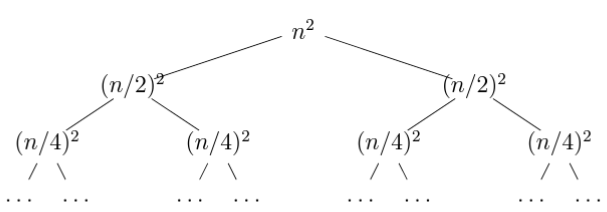


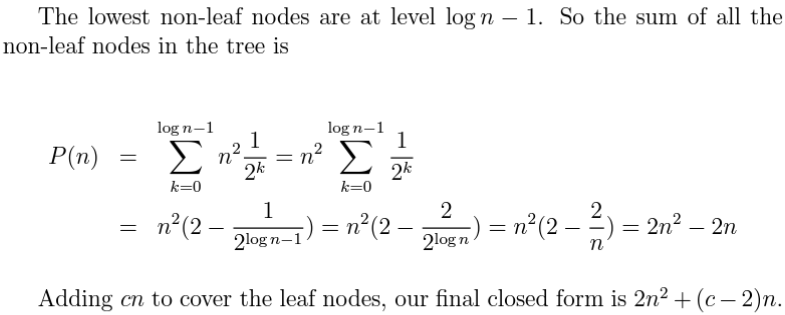
*(so here in S(n) each "top row" only has ~~2S(n/2)~~ + n, and the bottom row has the next iteration, which is n/2. Since there is a 2 multiplied to S(n/2), the bottom row has 2 nodes instead of 1)*

The final value of the function can be calculated by summing the value of every node in the tree. (The tree has height log(n) base 2, and the node values of each level happens to add up to just n. Plus each leaf node at the very bottom representing the base case is c, and there are n of them, so the final formula is n\*log(n) + cn)

Another Example:

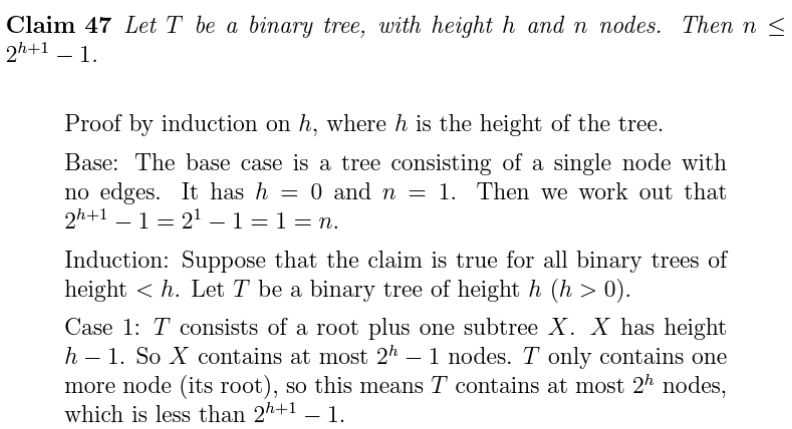


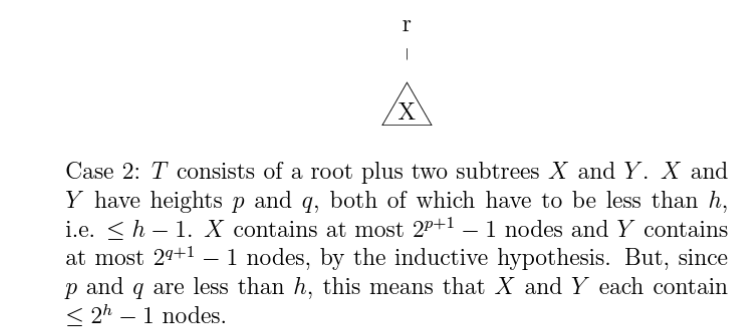


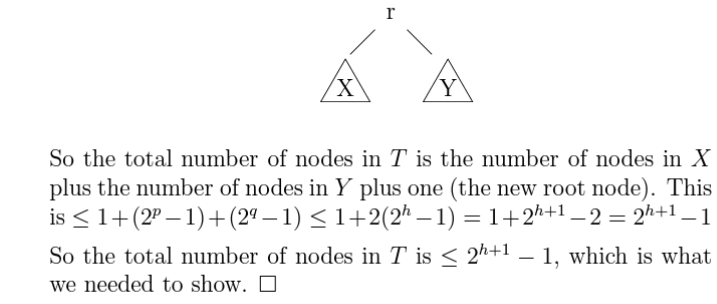


***Induction with Trees:***

We can write proofs using induction about trees by breaking up a tree into sub trees.







Same can be done with other kinds of trees, like grammar trees or trees with numerical nodes. (breaking up one tree into different sub trees and proof with different cases)