Tuesday 7/15: Trees and Grammars

Trees

tree: an undirected graph with a special node called the *root* node, where every node is connected to the root by exactly one path

- **parent:** neighboring node that is closest to the root
- **child:** neighboring node that is furthest from the root
- **siblings:** two children of the same parent
- **leaf:** a node with no children
- **internal node:** a non-leaf node
- **level:** of a node is the path length of that node to the root
- **height:** of a tree is the maximum level of any of its leaves
- **ancestor:** y of x means x is reachable by "child" relationships from y
- **descendant:** x of y means y is reachable by "parent" relationships from x
- **m-ary tree:** each node can have between 0 and m children
- **full m-ary tree:** each node has either 0 or m children
- **complete tree:** all leaves are at the same level
- **subtree:** like a subgraph relationship but the subgraph must have a tree structure

counting nodes example: How many nodes are there in a full and complete binary tree of height h?

At level 0, we have 1 node, at level 1, 2 nodes, then 2^2, 2^3, etc. Thus, the total number of nodes is given as $\sum_{l=0}^{h} 2^l = 2^{h+1} - 1$.

Grammars

Before we get into grammars, let's review strings.

Strings

string: a finite-length sequence of characters
length: of a string is the number of characters it contains
e.g., **pineapple** is a string of length 9.
ϵ: (epsilon, or the empty string) is used for the string containing no characters, which has length 0

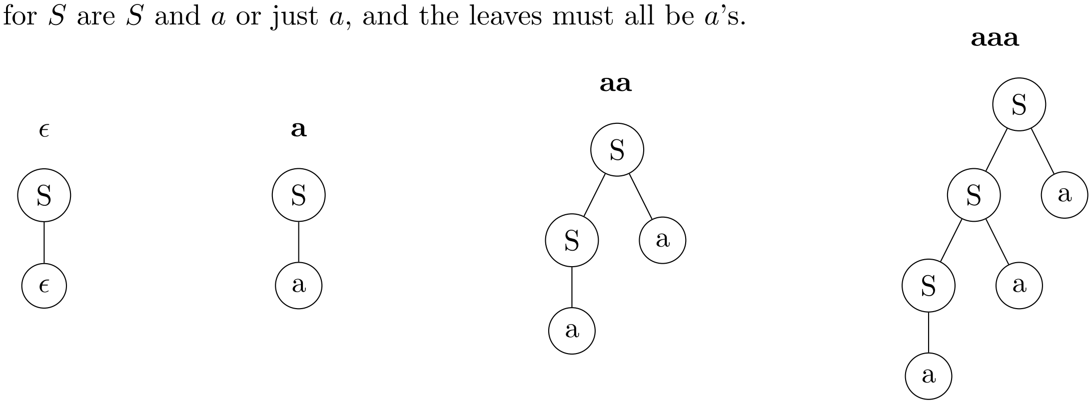
We specify concatenation of strings by writing them next to each other. For example, suppose $\alpha =$ **blue** and $\beta =$ **cat**. Then $\alpha\beta$ would be the string **bluecat** and βs is the string **cats**.

bit strings: consist of characters 0 and 1

Context-free Grammars

context free grammar: is a set of rules for structure of a tree

e.g., Let's define a grammar G with start symbol S and terminal symbol a with the following rules: $S \rightarrow \quad Sa \quad | \quad a \quad | \quad \epsilon$. This means we are creating trees rooted at S, where the possible children for S are S and a or just a, and the leaves must all be a's.



We read the leaves left to right in order to read the *terminal sequence*. We say that sequence was *generated* by G. This grammar G generates strings that are 0 or more a's.

grammar example: design a context-free grammar that generates strings of the form $a^n bc^n$ where $n \geq 0$. That is, 0 or more a's followed by a b and then the same number of c's.

Let's design a grammar G with start symbol V, and terminal symbols a, b, c.

$S \quad \rightarrow \quad b \quad | \quad aSc$

What if we wanted $a^* bc^*$ instead? In other words, any number of 0 or more a's and c's.

Then, we have the following rules:

$S \quad \rightarrow \quad b \quad | \quad aS \quad | \quad Sc$