

CSC148 – Introduction to Computer Science

Lecture 6: Trees

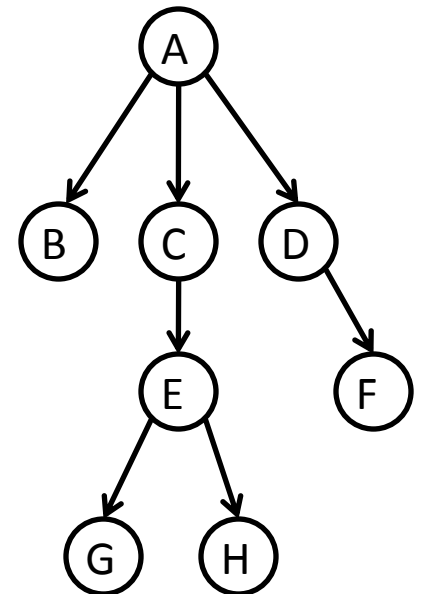
Sean Henderson

Trees

- Extremely useful data structure
- Defined recursively
 - A tree contains a *root* node
 - The root node has a collection of *child* nodes
 - Each child node is essentially another tree

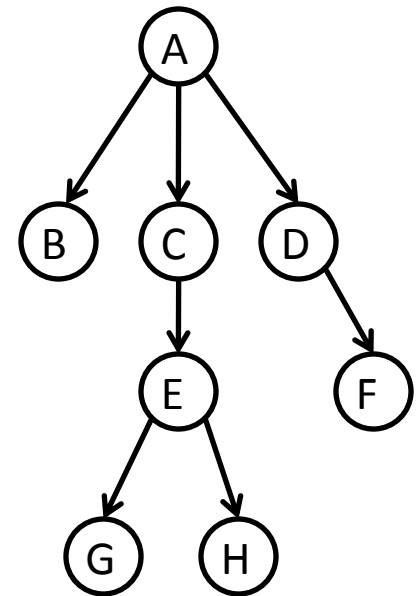
Example/Terminology

- The image on the right represents a tree
- Tree root is labeled A
- A has 3 children
 - B, C, D
- C has one child, as does D



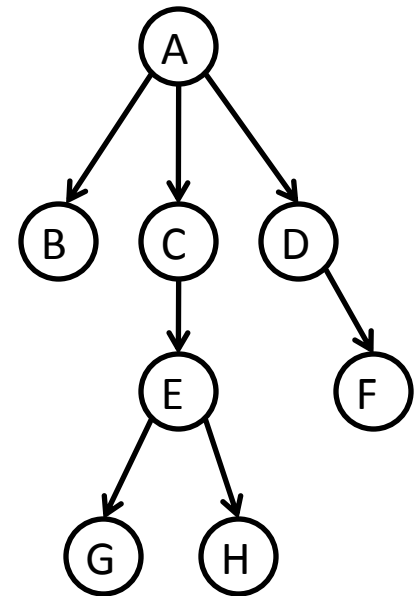
Example/Terminology

- We say that B's *parent* is A
- Nodes with no children are called *leaf nodes*
 - 4 leaf nodes in this tree (B,G,H,F)
- Nodes with the same parent are called *siblings*
- Connections from parents to children are called *edges*
- This tree has 7 edges



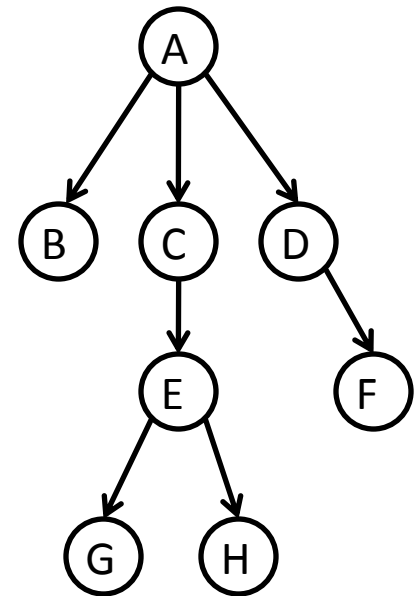
Example/Terminology

- A *path* is a sequence of edges one would follow to get to one node from another
- The *length* of a path is how many edges are on it
- The *depth* of a node is the length of the path from the root to that node
- What is the depth of D? H?



Example/Terminology

- The *height* of the tree is the maximum depth of any leaf node
- The height of this tree is 3
 - A-G and A-H both have this length



Examples of Trees

- Genealogy (family trees)
- Parse trees (A1, part 3)
- Expression trees (mathematical expressions)
- Some organizational charts
- The basis of many, many interesting data structures!

Binary Trees

- Each node has at most 2 children
- Each child is labeled either *left* or *right*
- Question: for a binary tree of height n , what is the maximum number of nodes it can contain?
- Let's work on an example...

Tree Traversal

- Can recursively iterate over the nodes in a tree
- Visit the root node, then recursively visit its children
- Three types of traversals:
 - *Inorder*: left child, root, right child
 - *Preorder*: root, left child, right child
 - *Postorder*: left child, right child, root
- Each traversal has its own use