CSCI 204: Data Structures & Algorithms

Revised by Xiannong Meng based on textbook author's notes

Binary Tree ADT

- So far the ADTs we studied are linear
  - Lists
  - Arrays
  - Stacks
  - Queues
- Some applications require non-linear ADTs. Examples may include ADTs
  - To represent an organization
  - To represent class inheritance in OOP
  - To represent a complex algebraic expression
  - Or even to represent a collection of sorted numbers!

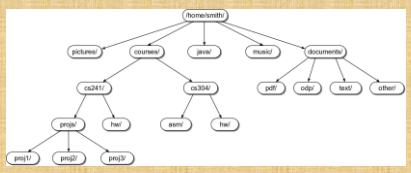
The Tree Structure

- Consists of nodes and edges that organize data in a hierarchical fashion.
  - **nodes** – store the data elements.
  - **edges** – connect the nodes.
- The organization of the nodes form relationships among the data elements.

Definition of a tree

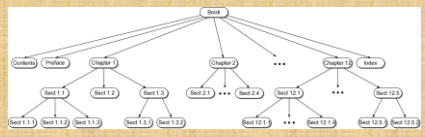
- A tree data structure is defined as follows.
  - A tree consists of a node called **root**
  - The root may have zero or more children
  - Each child itself is a **tree**
- A **recursive** definition!

Tree Example #1



Linux file system is a tree!

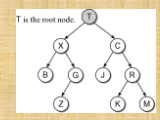
Tree Example #2



Chapters and sections in a book can be organized as a tree!

Root Node

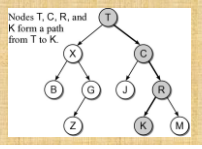
- Topmost node of the tree.
  - Provides the single access point into the tree.
  - Has no incoming edges.



7

Tree Path

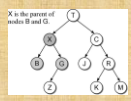
- The nodes encountered when following the edges from the root node to the destination node.
- Access to all other nodes must start with the root.



8

Parent Node

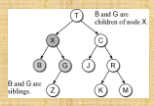
- The node from which an incoming edge originates.
  - Every node, except the root, has a parent node.
  - A node can only have one parent.



9

Child Node

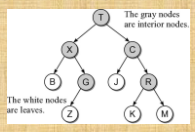
- The nodes to which outgoing edges are connected.
  - Each node can have one or more child nodes.
  - Results in a parent-child relationship.
  - **sibling nodes** – all nodes that have the same parent.



10

Types of Nodes

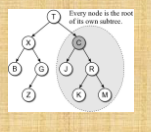
- Nodes can be classified as either:
  - **interior node** – a node that has at least one child.
  - **leaf node** – a node that has no children.



11

Subtree

- A tree is by definition a recursive structure.
  - Every node can be the root of its own subtree.
  - A **subtree** consists of a subset of nodes and edges of the larger tree.



12

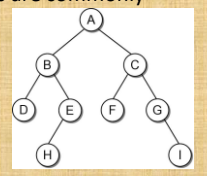
Relatives

- **descendants**
  - All nodes of a subtree are the descendants of the subtree's root.
  - Every node in the tree is a descendant of the root.
- **ancestors**
  - The ancestors of a node include all of the nodes along the node's path, excluding the node itself.
  - The root is the ancestor of all the other nodes.

13

The Binary Tree

- A tree in which each node can have at most two children. The nodes are commonly labeled:
  - left child
  - right child



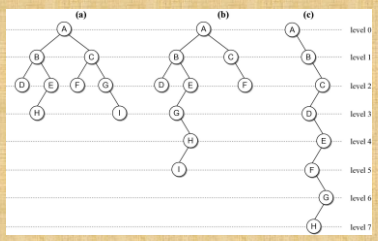
14

Binary Tree Properties

- There are several properties associated with binary trees that depend on the node organization.
  - **depth** – the distance of a node from the root.
  - **level** – all nodes at a given depth share a level.
  - **height** – number of levels in the tree.
  - **width** – number of nodes on the level containing the most nodes.
  - **size** – number of nodes in the tree.

15

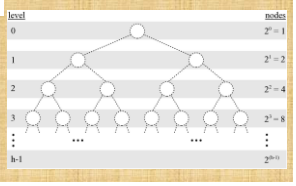
Binary Tree Properties



16

Binary Tree Properties

- Given a tree of size n:
  - max height = n
  - min height = $\lfloor \log_2 n \rfloor + 1$



17

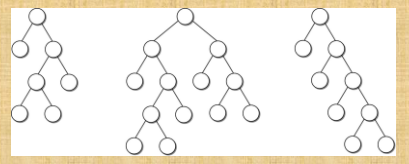
Binary Tree Structure

- Height of a tree will be important in analyzing the efficiency of binary tree algorithms.
- Structural properties can play a role in the efficiency of an algorithm.

18

Full Binary Tree

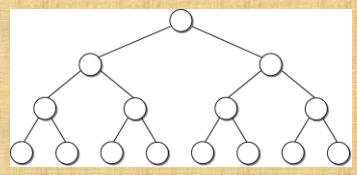
- A binary tree in which each interior node contains two children.



19

Perfect Binary Tree

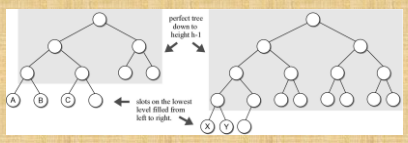
- A full binary tree in which all leaf nodes are at the same level.



20

Complete Binary Tree

- A binary tree of height **h**, is a perfect binary tree down to height **h – 1** and the nodes at the lowest level are filled from left to right (no gaps).



21