16 – Trees

A tree is a non-linear hierarchical structure.

- Tree is comprised of a set of nodes in which elements are stored and edges connect one node to another.

- A node can have only one parent, but may have multiple children.
- Nodes that have the same parent are siblings.
- The root is the only node which has no parent.
- A node that has no children is a leaf node.
- A node that is not the root and has at least one child is an internal node.
Tree Terminology

- A **subtree** is a tree structure that makes up part of another tree.
- We can follow a **path** through a tree from parent to child, starting at the root.
- The **path length** is determined by counting the number of edges that must be followed to get from the root to the node.
- A node is an **ancestor** of another node if it is above it on the path from the root.
- Nodes that can be reached by following a path from a particular node are the **descendants** of that node.

<table>
<thead>
<tr>
<th>Level</th>
<th>Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>B,C</td>
</tr>
<tr>
<td>2</td>
<td>D,E</td>
</tr>
<tr>
<td>3</td>
<td>F,G</td>
</tr>
</tbody>
</table>

- The **level** of a node is the length of the path from the root to the node.
- The **height** of a tree is the length of the longest path from the root to a leaf.

Tree Classifications

- We classify trees by the **maximum number of children** any node in the tree may have.
- **General trees** have no limit to the number of children a node may have.
- **n-ary trees** limits each node to no more than *n* children.
- **binary trees** are those in which nodes may have at most two children.
- A tree is **balanced** if all of the leaves of the tree are on the same level or at least within one level of each other.
- A **balanced binary** tree with *n* nodes has a height of $O(\log_2 n)$.
- A **balanced n-ary** tree with *m* nodes will have a height of $O(\log_n m)$. 
**Balanced, Complete and Full**

- A tree is *balanced* if all of the leaves of the tree are on the same level or at least within one level of each other.
- An n-ary tree is *full* if all leaves of the tree are at the same height and every non-leaf node has exactly n children.
- A tree is *complete* if it is full, or full to the next-to-last level with all leaves at the bottom level on the left side of the tree.

**Tree Traversals**

- Traversing a tree (visiting all nodes in a sequence) is generally more interesting than traversing a linear structure.
- A particular type of traversal simply dictates the order in which the elements of a collection are assessed.
Nodes are visited **before** any subtrees are visited
Visit Node
Traverse (left)
Traverse (right)

Visit the root in **between** the traversals of the left and right subtrees
Traverse (left)
Visit Node
Traverse (right)

Visit the root node **after** the traversals of the left and right subtrees
Traverse (left)
Traverse (right)
Visit Node

**Level-Order Traversal**

- Visit the nodes on each level, left to right, top to bottom starting at the root
Enqueue the root node of the tree
While the queue is not empty{
    Dequeue node
    Visit node
    Enqueue left child of node
    Enqueue right child of node
}
Strategies for Implementing Trees

- There are methods that use arrays or links
- Array-based implementations are the less obvious choice, but sometimes useful
  - Computed Links in Array
  - Stored Links in an Array

- Link-based implementations are more powerful and efficient, but also more complicated
- You should know about all options!

Computed Links in an Array

- Place tree nodes in specific indices of the array
- A node’s index can be used to calculate the indices of its parent and children

D is in location 3. Where can you find D’s children? How about D’s parent?
Stored Links in an Array

- Array positions are allocated on a first-come, first-served basis
- Each element of the array is an object that stores a reference to the tree element and the array index of each child

D is in location 4.
Where can you find D’s children? How about D’s parent?