**Trees**

- **Tree (or "Free Tree")**
  - undirected
  - connected
  - acyclic graph

- **Directed graph**
- **Undirected graph**

**Rooted Tree**: A free tree with root node

**Formal definition**:
- **Rooted tree**: is either
  - a single node (root)
  - a set of one or more rooted trees ("subtrees") joined to a common root

**Graph**:
- \( G = (V, E) \)
  - \( V \) = finite set of vertices (nodes)
  - \( E \) = set of edges (pairs of vertices)

**Depth**:
- path length from root

**Height**:
- (of tree) max depth

**Degree (of node)**: number of children

**Degree (of tree)**: max. degree of any node

**Family Relations**

- grandparent
- parent
- child
- sibling
- grandchild
- leaf: no children
Representing rooted trees: Each node stores a (linked) list of its children.

Node structure:
- data
- firstChild
- nextSibling

Wasted space?

Theorem: A binary tree with n nodes has \( n+1 \) null links.

In Java: class BTNode\{E\} {
  data;
  BTNode\<E\> left;  // left
  BTNode\<E\> right; // right
}...
Traversals: How to (systematically) visit the nodes of a rooted tree?

Binary Tree Traversals (can be generalized)

- Preorder
  - process/visit v
  - traverse v.left
  - traverse v.right

- Inorder
  - traverse v.left
  - process/visit v
  - traverse v.right

- Postorder
  - traverse v.left
  - traverse v.right
  - process/visit v

Complete Binary Tree: All levels full (except last)

Challenge: Non-recursive traversals

Binary Trees: Traversals, Extension, and More

Thm: An extended binary tree with n internal nodes (black) has n+1 external nodes (blue)

Observation: Every extended binary tree is full

Extended binary tree: Replace each null link with a special leaf node: external node

Those wasteful null links...

Another way to save space...

Threaded binary tree:
Store (useful) links in the null links. (Use a mark bit to distinguish link types.)

Eg. Inorder Threads:
Null left \rightarrow inorder predecessor
Null right \rightarrow "successor

Preorder: \(* + a b c - d e\)
Postorder: \((a b + c) * d e -\)
Inorder: \(a + b * c / d - e\)