Trees
Terminology

- Trees are hierarchical
  - “parent-child” relationship exits
    - A is the parent of B
    - B is a child of A
    - B and C are siblings
    - Generalized to ancestor and descendant
  - root: the only node without parent
  - leaf: a node has no children
  - Subtree: any node together with all of its descendents
General Tree v.s. Binary Tree

• A general tree $T$ is a set of one or more nodes such that $T$ is partitioned into disjoint subsets:
  – A node $r$, the root
  – Sets that are general trees, called subtrees of $r$

• A binary tree is a set $T$ of nodes such that
  – $T$ is empty, or
  – $T$ is partitioned into 3 disjoint subsets:
    • A node $r$, the root
    • 2 possibly empty sets that are binary trees, called left and right subtrees of $r$
General Tree v.s. Binary Tree

(a) President
   - VP Marketing
     - Director Media Relations
   - VP Manufacturing
     - Director Sales
   - VP Personnel

(b) Caroline
   - John
     - Joseph
   - Jacqueline
     - Rose
Represent Algebraic Expressions using Binary Tree

\[ a - b \]  
(a)  

\[ a - b / c \]  
(b)  

\[ (a - b) \times c \]  
(c)
Binary Search Tree

- A binary search tree is a binary tree that is sorted according to the values in its node.
- For each node n
  - n’s value is greater than all values in its left subtree
  - n’s value is less than all values in its right subtree
A Binary Search Tree of Names

Jane

Bob

Alan

Ellen

Tom

Nancy

Wendy
Height of Trees

- Trees come in many shapes

- Height of any tree: number of nodes on the longest path from the root to a leaf
Full Binary Trees

- Full binary tree
  - All nodes that are at a level less than h have two children, where h is the height

- Each node has left and right subtrees of the same height
Full Binary Tree

• If the height is $h > 0$
  – The number of leaves is $2^{(h-1)}$
  – The number of nodes is $2^h - 1$

• If the number of nodes is $N > 0$
  – The height is $\log(N+1)$ with base 2
  – The number of leaves is $(N+1)/2$

• Has many leaves as possible among binary trees of height $h$
Complete Binary Trees

- Complete binary tree
  - A binary tree full down to level $h-1$, with level $h$ filled in from left to right
Balanced Binary Trees

• Balanced binary tree
  – The height of any node’s right subtree differs from the height of the node’s left subtree by no more than 1
  – A complete binary tree is balanced
ADT Binary Tree

• Operations
  – Create/destroy a tree
  – Determine/change root
  – Determine emptiness
  – Attach/Detach left/right subtree to root
  – Return a copy of the left/right subtree of root
  – Traverse the nodes in preorder, inorder, or postorder

<table>
<thead>
<tr>
<th>Binary tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
</tr>
<tr>
<td>left subtree</td>
</tr>
<tr>
<td>right subtree</td>
</tr>
<tr>
<td>createTree()</td>
</tr>
<tr>
<td>destroyBinaryTree()</td>
</tr>
<tr>
<td>isEmpty()</td>
</tr>
<tr>
<td>getData()</td>
</tr>
<tr>
<td>setData()</td>
</tr>
<tr>
<td>attachRight()</td>
</tr>
<tr>
<td>attachLeftSubtree()</td>
</tr>
<tr>
<td>attachRightSubtree()</td>
</tr>
<tr>
<td>detachLeftSubtree()</td>
</tr>
<tr>
<td>detachRightSubtree()</td>
</tr>
<tr>
<td>getLeftSubtree()</td>
</tr>
<tr>
<td>getRightSubtree()</td>
</tr>
<tr>
<td>preorderTraverse()</td>
</tr>
<tr>
<td>inorderTraverse()</td>
</tr>
<tr>
<td>postorderTraverse()</td>
</tr>
</tbody>
</table>
Build a Tree

Tree1.setRootData('F');
Tree1.attachLeft('G');

Tree2.setRootData('D');
Tree2.attachLeftSubTree(tree1);

Tree3.setRootData('B');
Tree3.attachLeftSubtree(tree2);
Tree3.attachRight('E');

Tree4.setRootData('C');
binTree.createBinaryTree('A', tree3, tree4);