3. Trees :: Binary Trees :: Traversals

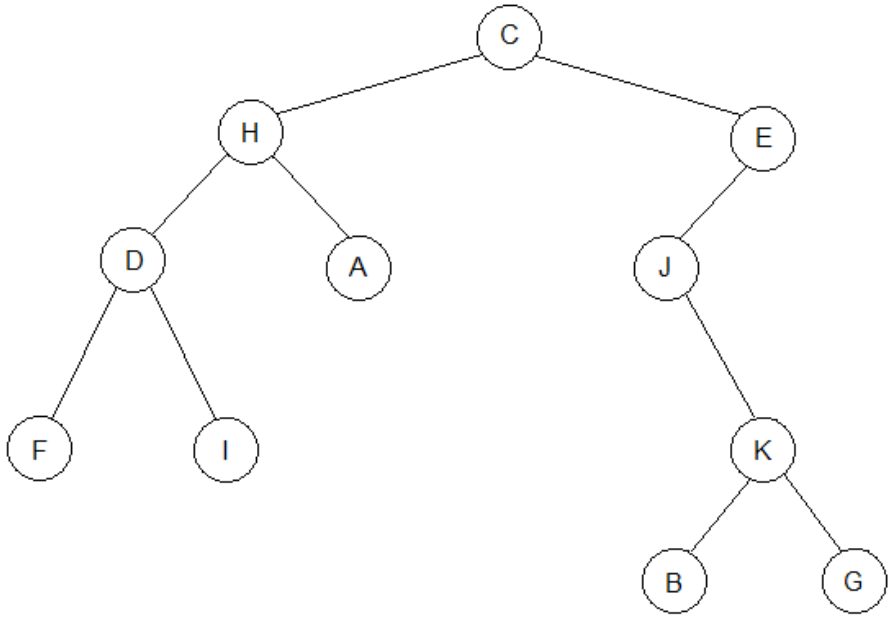
A **traversal** of a binary tree is the process of systematically **visiting** each node of the tree in order to perform some operation on the elements of the tree.

Two categories of traversals are **depth first** and **breadth first**. We will discuss depth first, first.

A depth first traversal is one where we follow one path downward emanating from a node before we follow other paths, also emanating from the same node. There are three standard depth first traversals.

1. Preorder
2. Inorder
3. Postorder

For this section, we will use this binary tree as our example:



3. Trees :: Binary Trees :: Traversals :: Depth First :: Preorder

A **preorder** traversal is one in which we **visit** the root node first, then we traverse the left subtree, and then we traverse the right subtree. For the example tree, the nodes would be visited in this order:

Note that our definition of preorder traversal is recursive in nature:

1. Visit the root node.
2. Perform a preorder traversal of the subtree rooted at the left child of the root node (if it exists).
3. Perform a preorder traversal of the subtree rooted at the right child of the root node (if it exists).

Here is the pseudocode:

```
-- Performs a preorder traversal of the subtree rooted at pRoot. pVisitor is an object which
-- implements a method named visit() which will be called as each Node is visited.
Method preorderTraversal(In: Node pRoot, In: pVisitor) Returns Nothing
   -- Visit the root node passing the data stored in the root node.
   pVisitor.visit(pRoot.data)

   -- Perform a preorder traversal of the subtree rooted at the left child pRoot if it exists.
   If pRoot has a left child Then preorderTraversal(pRoot.leftChild, pVisitor)

   -- Perform a preorder traversal of the subtree rooted at the right child pRoot if it exists.
   If pRoot has a right child Then preorderTraversal(pRoot.rightChild, pVisitor)
End Method preorderTraversal
```