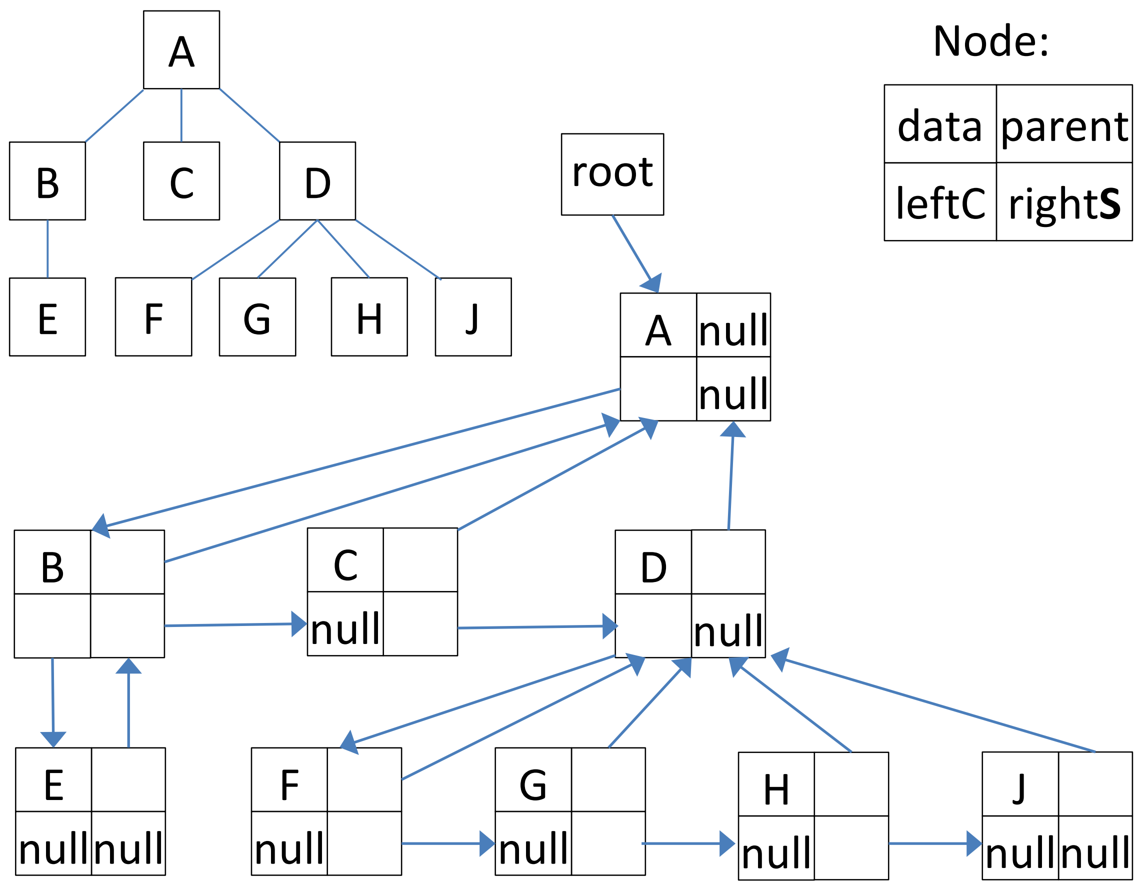
Non-Binary Trees:

```
class Node {
      ElementType data;
      Node parent, leftmostChild, rightSibling;
            // singly-linked list of each node's children
}
class Tree {
      Node root;
}
```

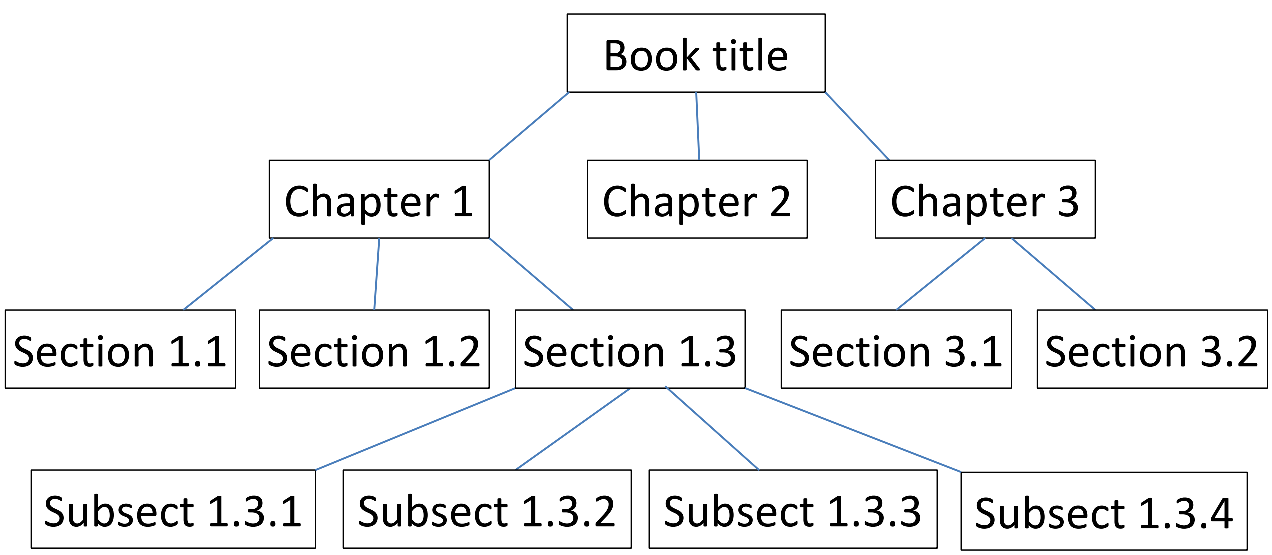


Alternative representation of Non-Binary Trees:

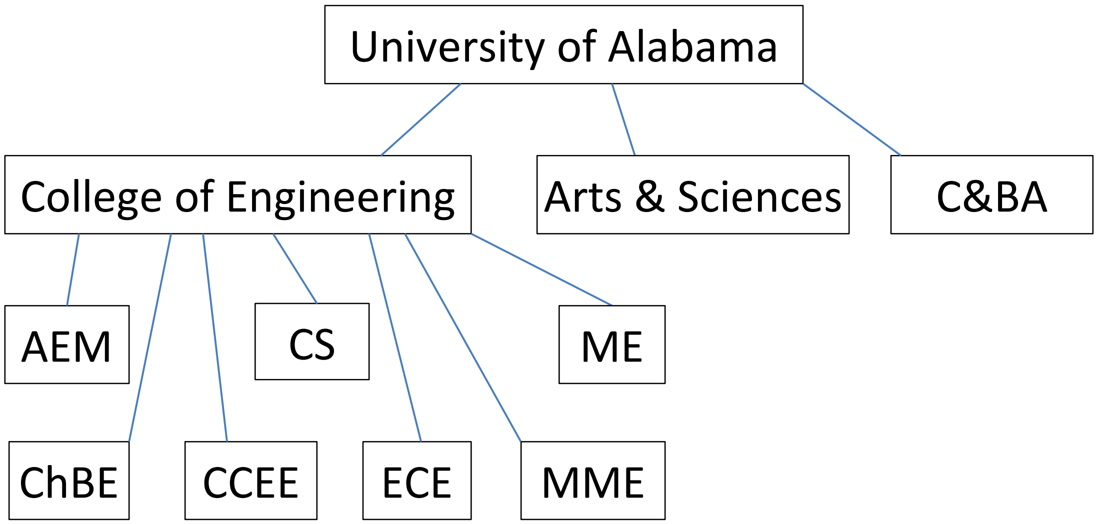
```
class Node {
    ElementType data;
    Node parent, leftmostChild, rightmostChild,
                        leftSibling, rightSibling;
        // doubly-linked list of each node's children
}
class Tree {
    Node root;
}
```

Applications:

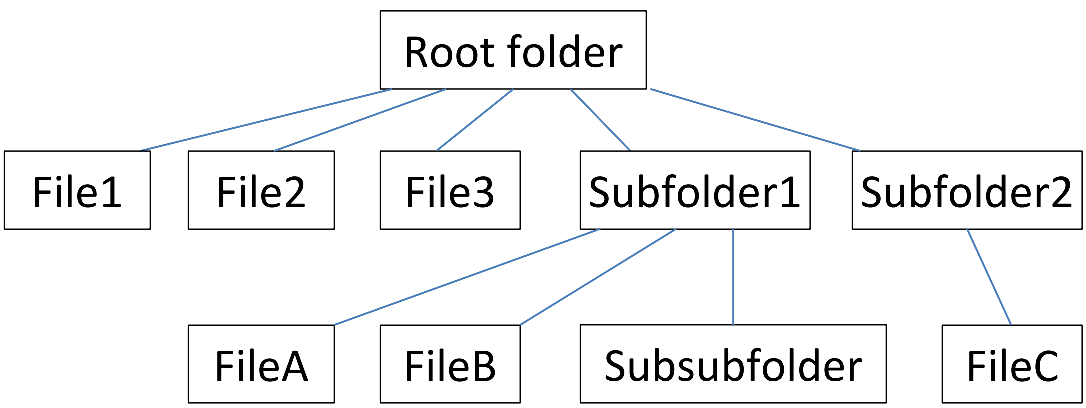
Contents of a book



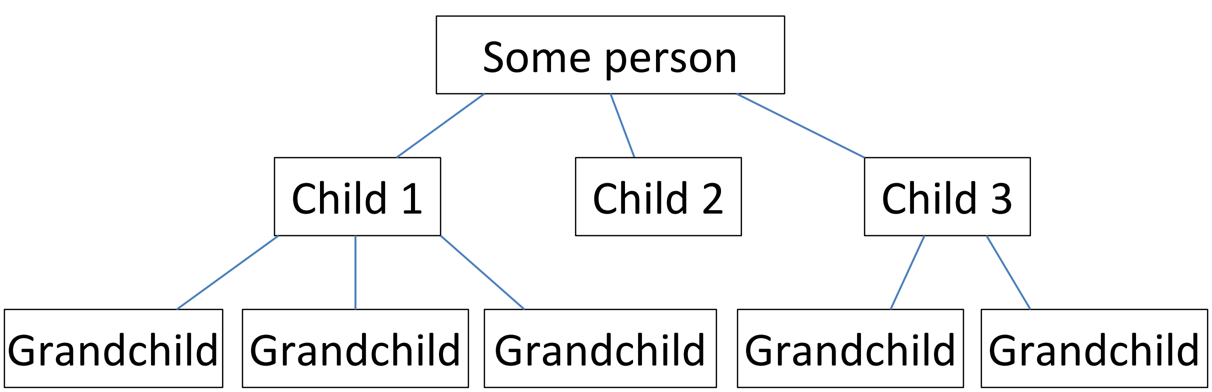
Hierarchy of an organization



File system



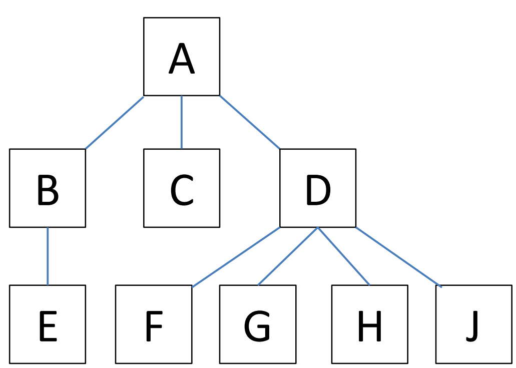
Family tree (descendants)



Traversals of a non-binary tree:

Preorder traversal

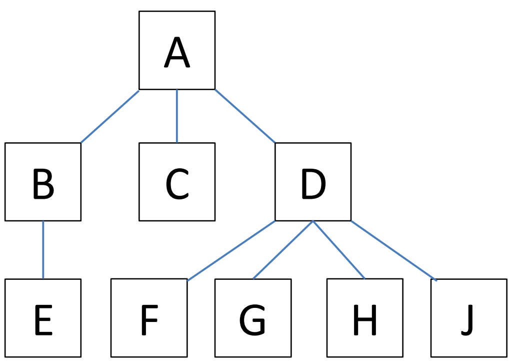
```
void preorder (Tree T) {
    preorder (T.root);
}
void preorder (Node p) {
    if (p==null) return;
    visit (p);
    preorder (p.leftmostChild);
    preorder (p.rightSibling);
}
```



A <u>B E</u> <u>C</u> <u>D F G H J</u>

Postorder traversal

```
void postorder (Tree T) {
    postorder (T.root);
}
void postorder (Node p) {
    if (p==null) return;
    postorder (p.leftmostChild);
    visit (p);
    postorder (p.rightSibling);
}
```



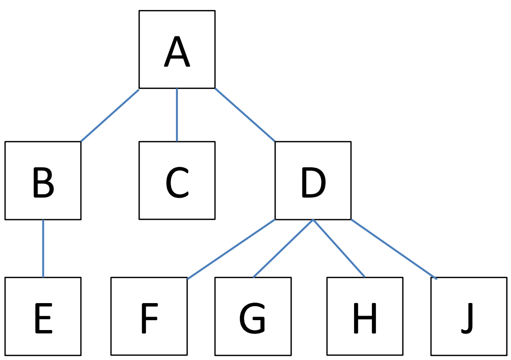
E B C F G H J D A

Inorder traversal

No natural definition of inorder for non-binary trees

Level-order traversal

```
void levelOrder (Tree T) {
    Queue Q( );
    Q.enqueue (T.root);
    while (not Q.isEmpty( )) {
            Node p = Q.dequeue( );
            visit (p);
            for (Node c=p.leftmostChild;  c!=null;  c=c.rightSibling)
                    Q.enqueue (c);
    }
}
```



A <u>B C D</u> <u>E F G H J</u>

Analysis: Let n = number of nodes in the tree. Each kind of traversal spends $\theta(1)$ time at each node of the tree, so each traversal has $\theta(n)$ total running time. [same as for binary trees]