

**CSIS 3103**  
Introduction to Trees

### Trees

- Trees are used to represent a hierarchical organization of data
- There are a number of variations in tree structures
  - Binary tree, general tree, B-tree, etc.
- Recursion is often used to process trees

### Trees

Chapter 6 looks at

- binary trees
- binary search trees
- heaps

Binary search trees in particular are good for storing sorted data so that it can be efficiently retrieved

### Tree Terminology

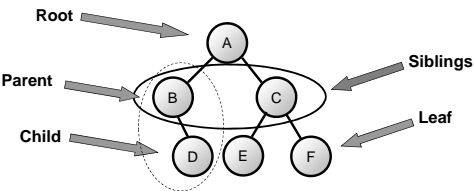
- A tree consists of a collection of elements (*nodes*). Each node is linked to its successors.
- *Root*: The node at the top of a tree
- *Branches*: The links from a node to its successors
- *Children*: The successors of a node
- *Parent*: The predecessor of a node

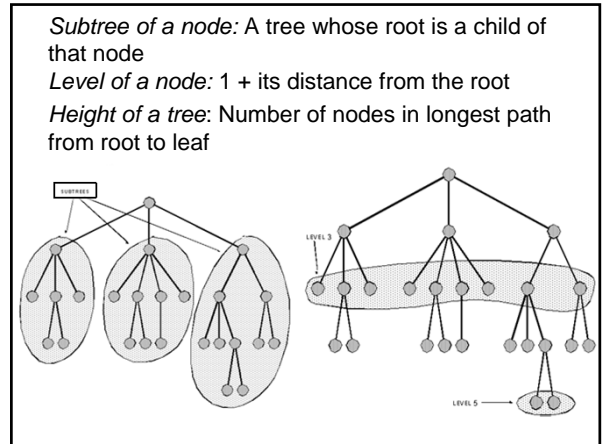
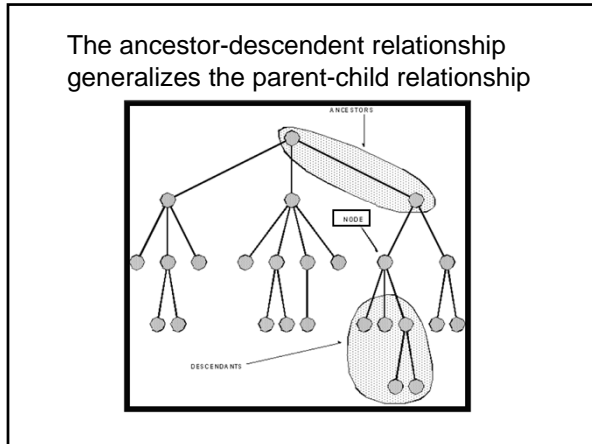


Each node in a tree has exactly one parent except for the root node, which has no parent

*Leaf node*: A node that has no children

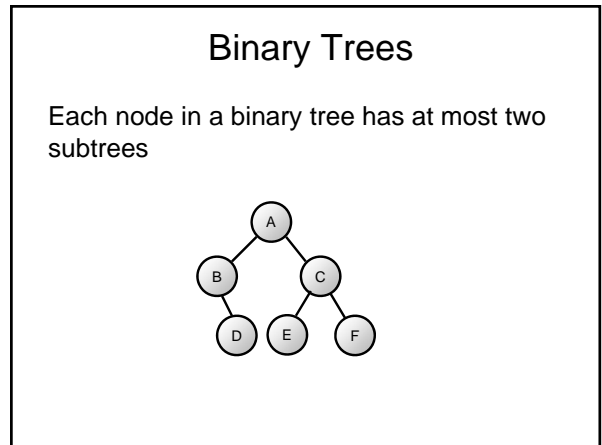
*Siblings*: Nodes that have the same parent





### Recursive Definition of Tree

A tree  $T$  is either the empty set or a pair  $T = (r, S)$ , where  $r$  is an object and  $S$  is a set of disjoint trees. The object  $r$  is called the *root* of the tree  $T$ , and the elements of the set  $S$  are *subtrees* of the root.

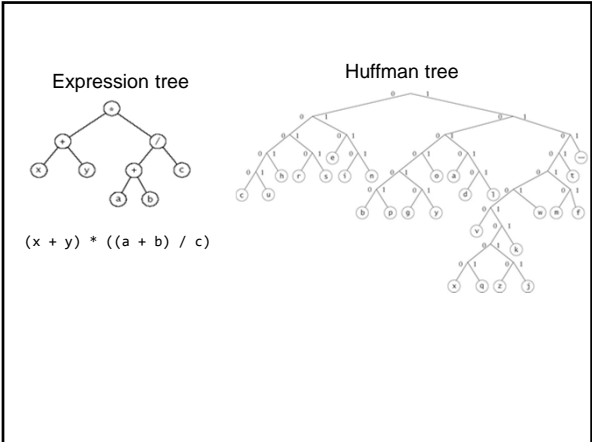


### Recursive Definition of Binary Tree

A binary tree  $T$  is either empty or consists of an object  $x$  and disjoint binary trees  $T_L$  and  $T_R$ . The object  $x$  is the *root* of the tree  $T$ ,  $T_L$  is the *left subtree* and  $T_R$  is the *right subtree* of the root

### Examples of Binary Trees

*Expression tree*: Each node contains an operator or an operand  
*Huffman tree*: Represents Huffman codes for characters that might appear in a text file (for file compression)  
*Binary search trees*: All elements in the left subtree precede those in the right subtree



### General Trees

- Nodes of a general tree can have any number of subtrees
- A general tree can be represented using a binary tree

The diagram shows two binary trees representing a general tree structure of names. The left tree has a root node **William I**. Its left child is **Robert**, which has a left child **William** and a right child **William II**. **William II** has a left child **Stephen** and a right child **Henry I**. **Stephen** has a left child **Henry** and a right child **Richard I**. **Henry** has a left child **Richard I** and a right child **Geoffrey**. **Richard I** has a left child **Arthur** and a right child **Henry III**. **Geoffrey** has a left child **Arthur** and a right child **Henry III**. **Henry III** has a left child **Edward I** and a right child **Edmund**. **Edward I** has a left child **Edward II** and a right child **Edmund**. **Edward II** has a left child **Edward III** and a right child **Thomas**. **Edmund** has a left child **Edmund** and a right child **Edmund**. The right tree is a similar binary tree with a root node **William I** and children **Robert** and **William II**. **Robert** has a left child **William** and a right child **William II**. **William II** has a left child **Stephen** and a right child **Henry I**. **Stephen** has a left child **Henry** and a right child **Manilda**. **Henry** has a left child **Henry II** and a right child **Richard I**. **Henry II** has a left child **Richard I** and a right child **Geoffrey**. **Richard I** has a left child **Arthur** and a right child **John**. **Geoffrey** has a left child **Arthur** and a right child **John**. **John** has a left child **Henry III** and a right child **Richard**. **Henry III** has a left child **Edward I** and a right child **Edmund**. **Edward I** has a left child **Edward II** and a right child **Richard**. **Edward II** has a left child **Edward III** and a right child **Edmund**. **Richard** has a left child **Edmund** and a right child **Edmund**.