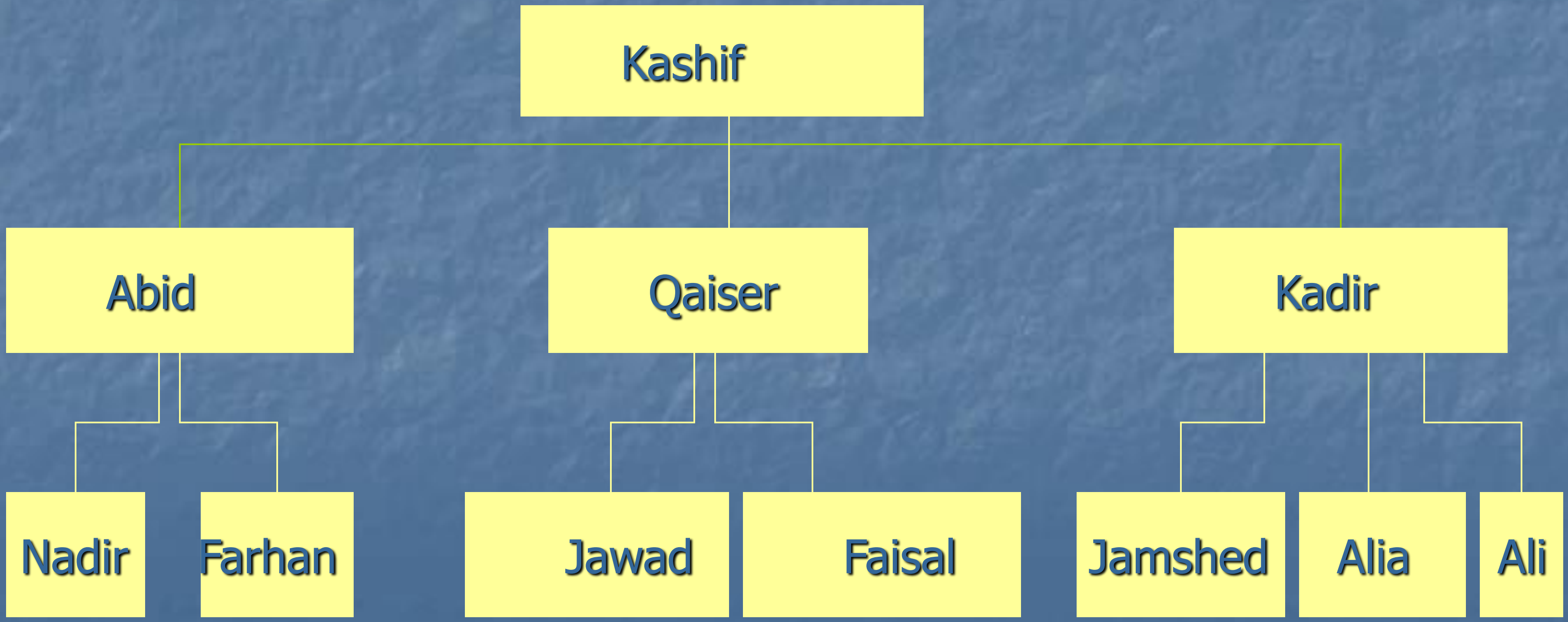
Trees

Tree Data Structures

- There are a number of applications where linear data structures are not appropriate. Consider a genealogy (family tree) tree of a family.



Tree Data Structure

- A linear linked list will not be able to capture the tree-like relationship with ease.
- Shortly, we will see that for applications that require searching, linear data structures are not suitable.
- We will focus our attention on *binary trees*.

- **Theorem:**

  A **Tree** with **n** vertices (nodes) has **n-1** edges.

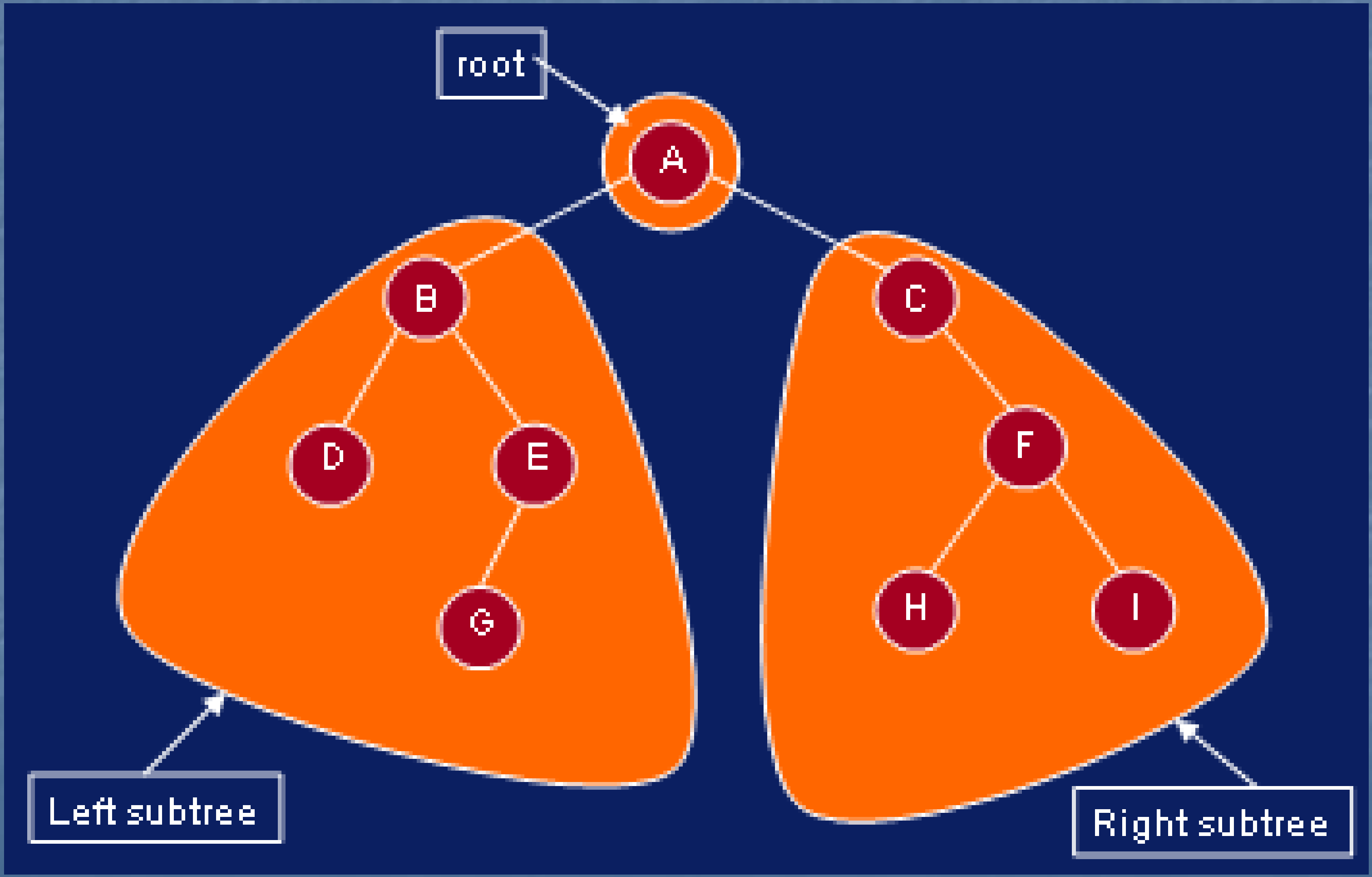
Binary Tree

- A *Binary Tree* is a finite set of elements that is either empty or is partitioned into three disjoint subsets. The first subset contains a single element called a root of the tree. The other two subsets are themselves binary trees, called the *left* and *right* sub trees of the original tree.
- A left or right sub tree can be empty.
- Each element of a binary tree is called a node of the tree.

- Following is an example of binary tree. This tree consists of nine nodes with A as its root. Its left sub tree is rooted at B and its right sub tree is rooted at C.

Figure 1

Binary Tree



- The absence of a branch indicates any *empty sub tree*. For example in previous figure, the left sub tree of the binary tree rooted at C is empty.
- Figures below shows that are ***not*** *Binary Trees*.

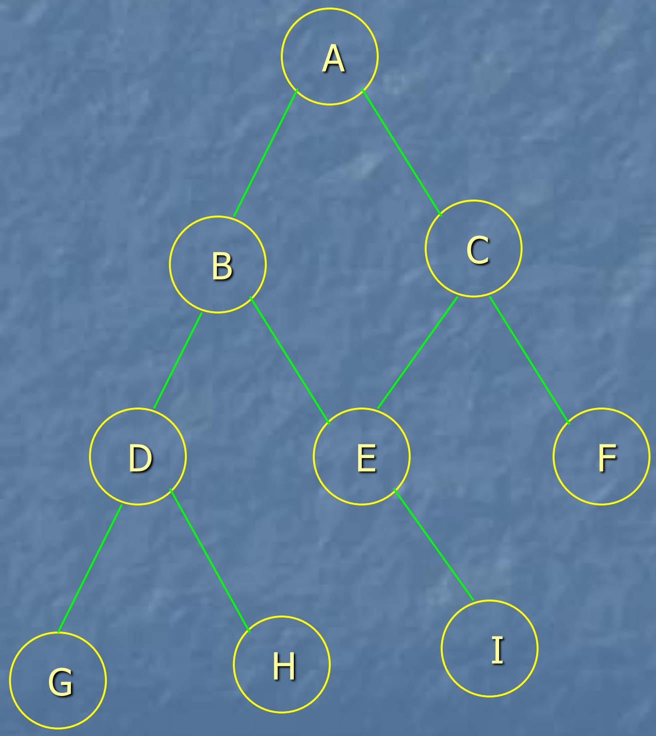


Figure 2

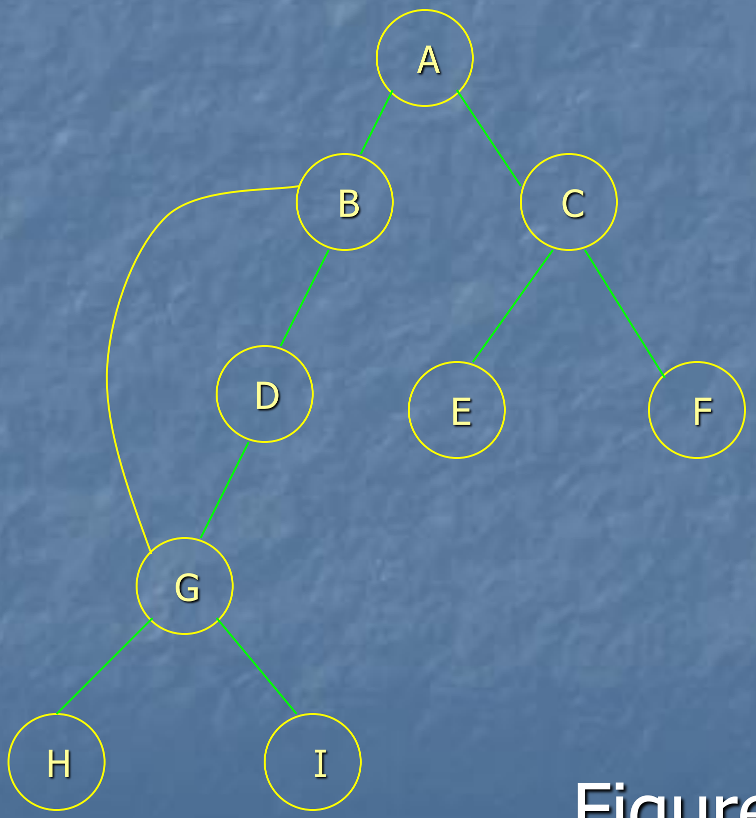


Figure 3

- If A is the root of binary tree and B is the root of its left or right sub tree, then A is said to be father or parent of B and B is said to be left or right son (child) of A.

- A node that has no child (son) is called *leaf*.

- Node n1 is an ancestor of node n2 (and n2 is a descendent of n1) if n1 is either the parent of n2 or the parent of *some* ancestor of n2. For example in previous Figure 1 A is an ancestor of G and H is a descendent of C, but E is neither a descendent nor ancestor of C.

- If every non leaf node in a binary tree has non empty left and right sub trees, the tree is termed as *strictly binary tree*. Following Figure 4 is strictly binary tree while Figure 1 is not.
- A strictly binary tree with **n** leaves always contain **2n – 1** nodes.

Figure 4

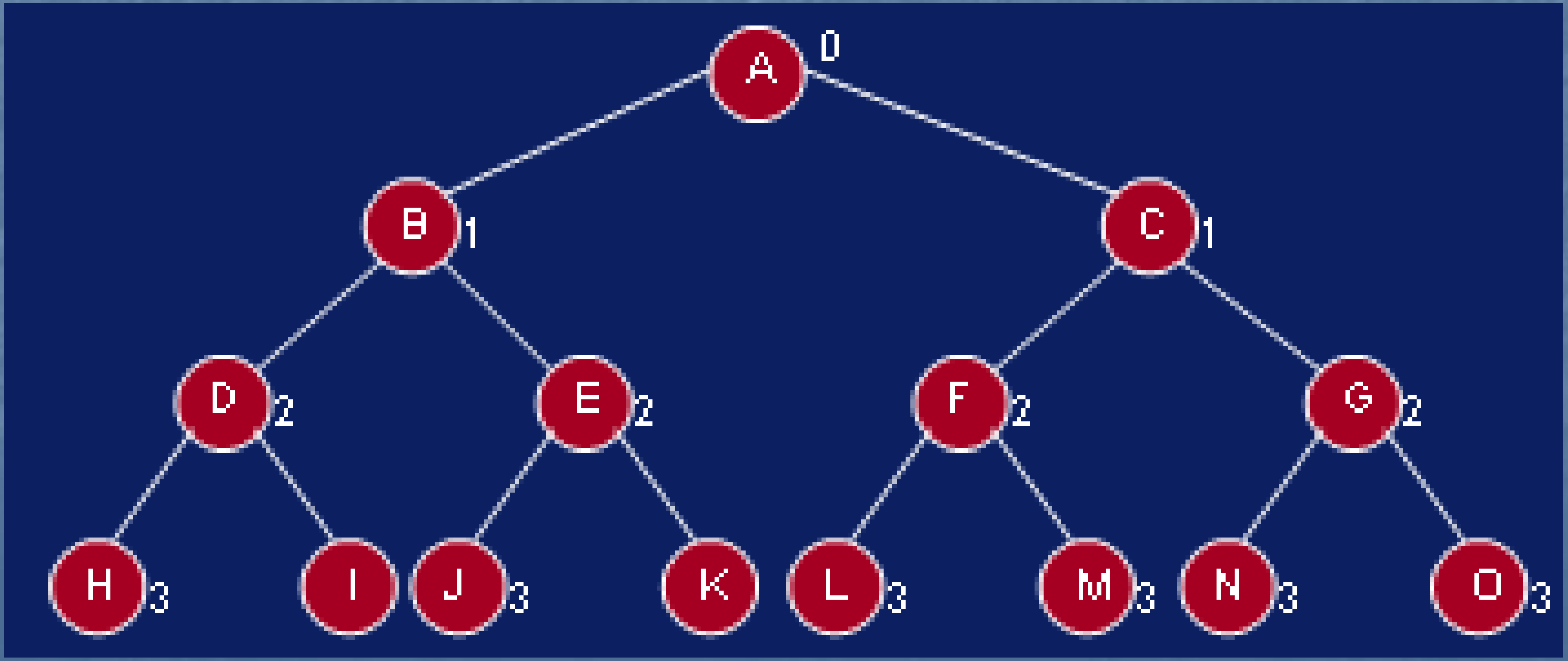
- The level of a node in a binary tree is defined as follows :-> The root of the tree has level 0, and the level of any other node in the tree is one more than the level of its parent. For example in Figure 1 node E is at level 2 and node H is at level 3.

- The depth of a binary tree is the maximum level of any leaf in the tree. Thus depth of Figure 1 is **3**.

- A Complete Binary Tree ,is a binary tree in which each level of the tree is completely filled except possibly the bottom level, and in this level the *nodes are in the left most positions*. For example,

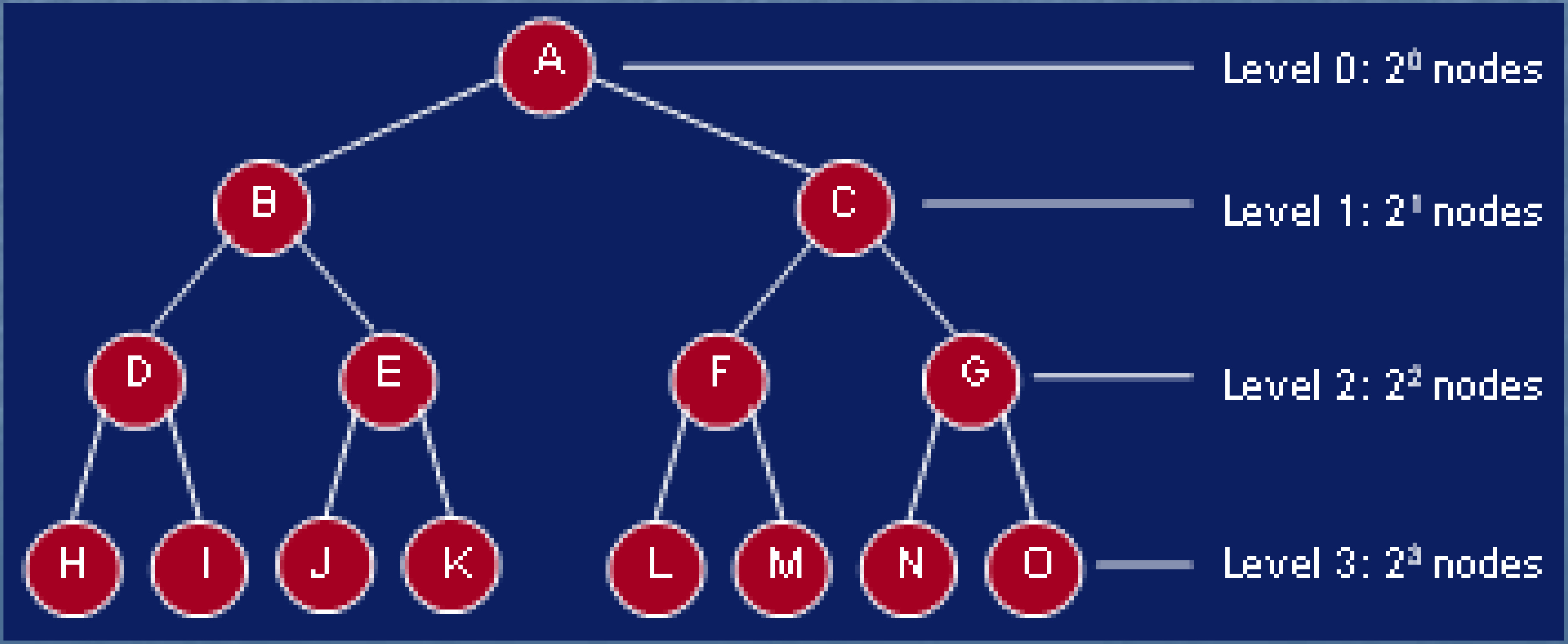
Complete Binary Tree

- A *complete binary tree* of depth *d* is the strictly Complete binary if all of whose leaves are at level *d*.

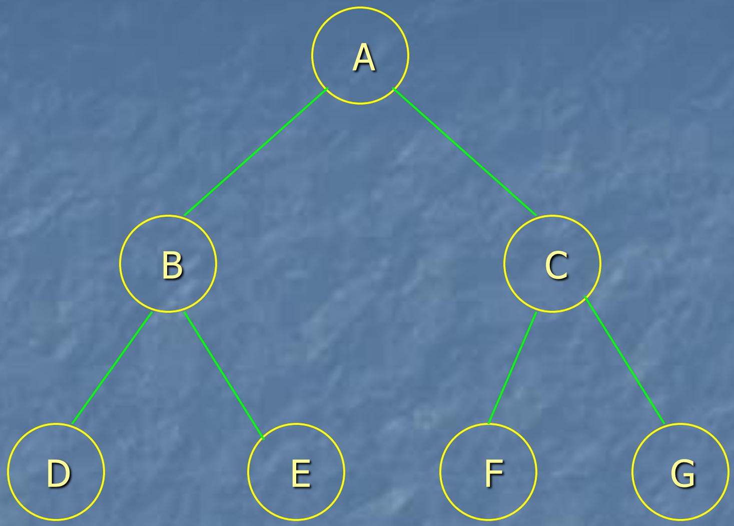


Complete Binary Tree

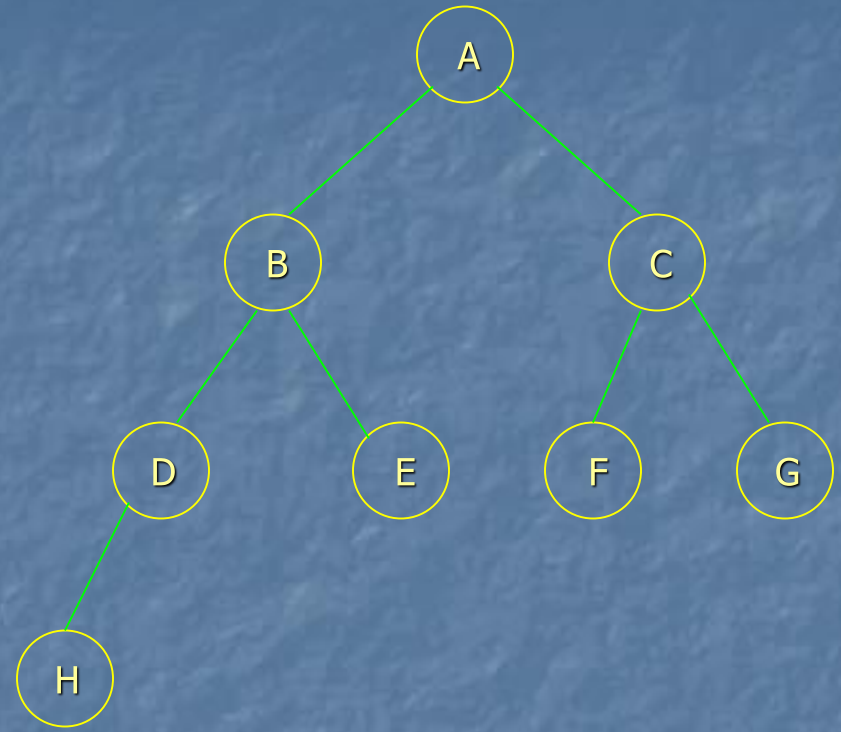
- A *complete binary tree* of depth *d* is the strictly binary all of whose leaves are at level *d*.



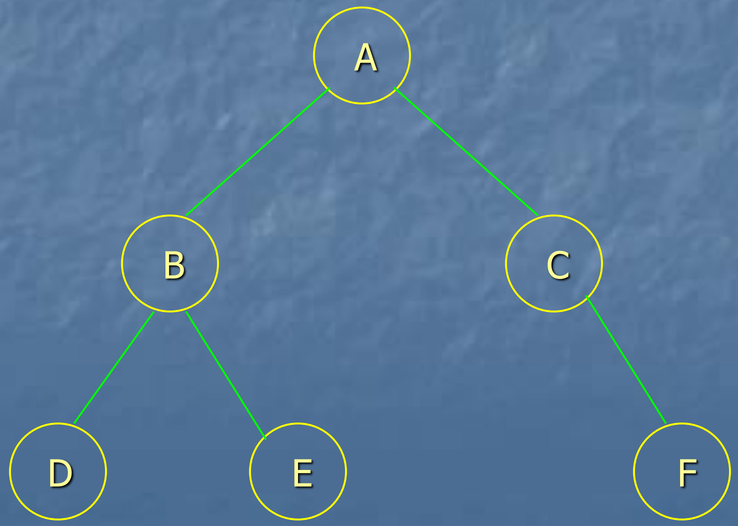
Strictly Complete Binary Tree



Complete Binary Tree



Not Complete Binary Tree



- Another common property is to traverse a binary tree i.e. to pass through the tree, i.e. enumerating or visiting each of its nodes once.

Thank You……