Non Linear Data Structures

The Trees

In this chapter we will discuss one of the important non-liner data structure in computer science, Trees. Many real life problems can be represented and solved using trees.
Trees are very flexible, versatile and powerful non-liner data structure that can be used to represent data items possessing hierarchical relationship between the grand father and his children and grandchildren as so on.

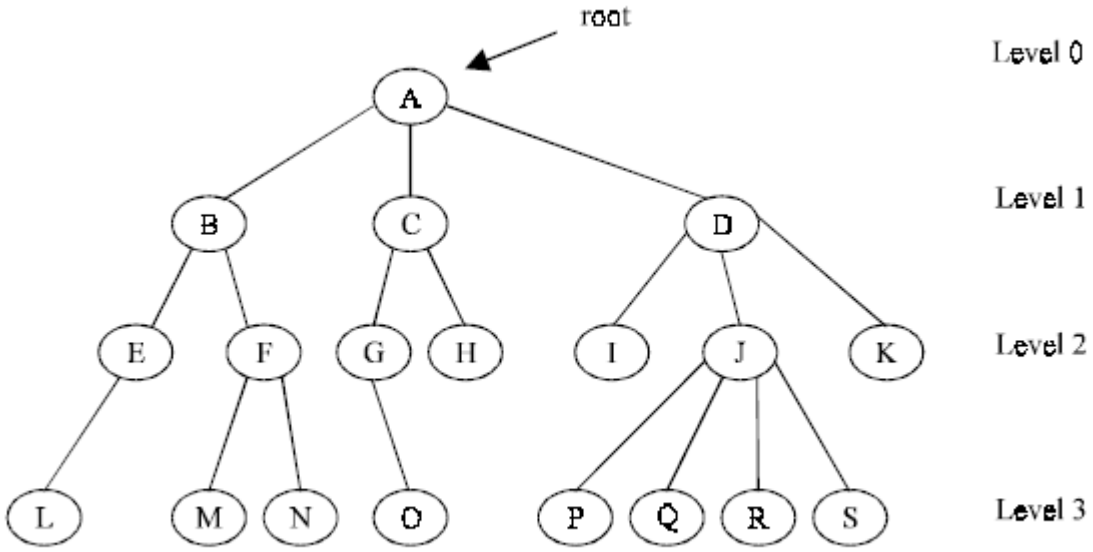

Fig. 8.1. A tree

A tree is an ideal data structure for representing hierarchical data. A tree can be theoretically defined as a finite set of one or more data items (or nodes) such that:

1. There is a special node called the root of the tree.
2. Removing nodes (or data item) are partitioned into number of mutually exclusive (i.e., disjoined) subsets each of which is itself a tree, are called sub tree.

BASIC TERMINOLOGIES

Root is a specially designed node (or data items) in a tree. It is the first node in the hierarchical arrangement of the data items. 'A' is a root node in the Fig. 8.1. Each data item in a tree is called a node. It specifies the data information and links (branches) to other data items.

Degree of the node is the number of sub trees of each node in a given tree. In fig. 8.1.

The degree of node A is 3
The degree of node B is 2
The degree of node C is 2
The degree of node D is 3

The degree of a tree is the maximum degree of node in a given tree. In the above tree, degree of node J is 4. All other nodes have less or equal degrees. So the degree of the above tree is 4. A node with degree zero is called a terminal node or a leaf. For example in the above tree fig. 8.1. M, N, I, O etc.,

are leaf nodes. Any node whose degree is non zero is called non terminal node. They are intermediate nodes in traversing the given tree from its root to the terminal nodes.

The tree is structured in different levels. The entire tree is leveled in such a way that the root node is always of level 0. Then, its immediate children are at level 1 and their immediate children are at level 2 and so on up to the terminal nodes. That is, if a node is at level n, then its children will be at level n+1.

Depth of a tree is the number of nodes in the maximum level in a given tree. That is a number of level one can descend the tree from its root node to the terminal node (leaves). The term height is also used to denote the depth.

Binary trees

A binary tree is a tree in which no node can have more than two children. Typically these children are described as "left child" and "right child" of the parent node.

A binary tree T is defined as a finite set of elements, called nodes, such that:

1- T is empty (i.e., if T has no nodes called the null tree or empty tree).
2- T contains a special node R, called the root of T, and the remaining nodes of T form an ordered pair of disjoined binary trees T1 and T2, and they are called left and right sub tree of R. if T1 is non empty then its root is called the left successor of R, similarly if T2 is non empty then its root is called the right successor of R.

Consider a binary tree T in Fig. 8.3. Here 'A' is the root node of the binary tree T. Then 'B' is the left child of 'A' and 'C' is the right child of 'A' i.e., 'A' is a father of 'B' and 'C'. The node 'B' and 'C' are called brothers, since they are left and right child of the same father. If a node has no child then it is called a leaf node. Nodes D, H,I,F,J are leaf node in Fig. 8.3.

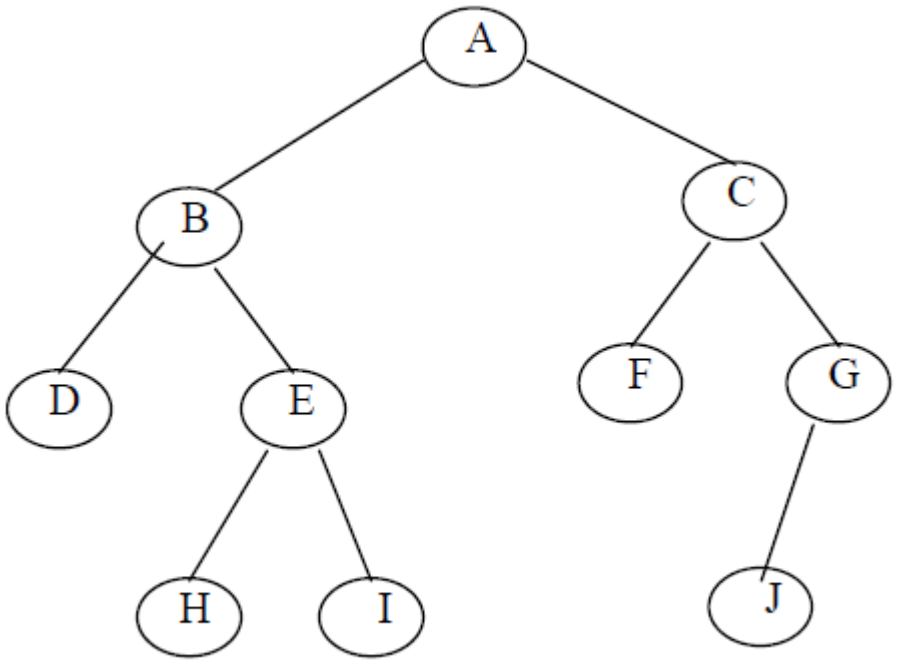

Fig. 8.3. Binary tree

The tree is said to be strictly binary tree, if every non-leaf made in a binary tree has non-empty left and right sub trees. A strictly binary tree with n leaves always contains 2n– 1 nodes. The tree in Fig. 8.4 is strictly binary tree, where as the tree in Fig. 8.3 is not. That is every node in the strictly

binary tree can have either no children or two children. They are also called 2-tree or extended binary tree.

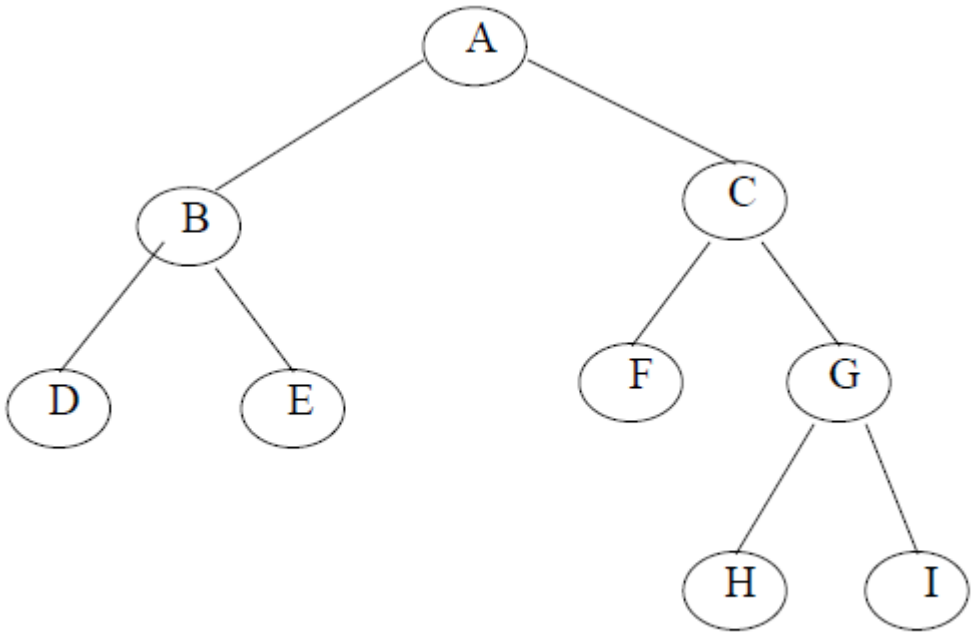

Fig. 8.4. Strictly binary tree

The main application of a 2-tree is to represent and compute any algebraic expression using binary operation.

For example, consider an algebraic expression E.

$$E = (a + b) / ((c – d)*e)$$

E can be represented by means of the extended binary tree T as shown in Fig. 8.5. Each variable or constant in E appears as an internal node in T whose left and right sub tree corresponds to the operands of the operation.

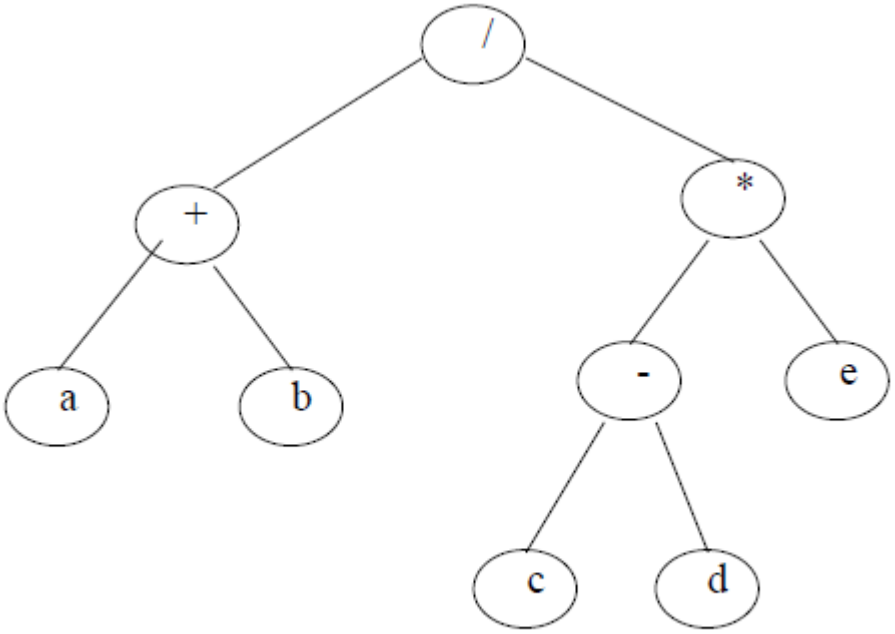

Fig. 8.5. Expression tree

A complete binary tree at depth 'd' is the strictly binary tree, where all the leaves are at level d. Fig. 8.6 illustration the complete binary tree of depth 3.

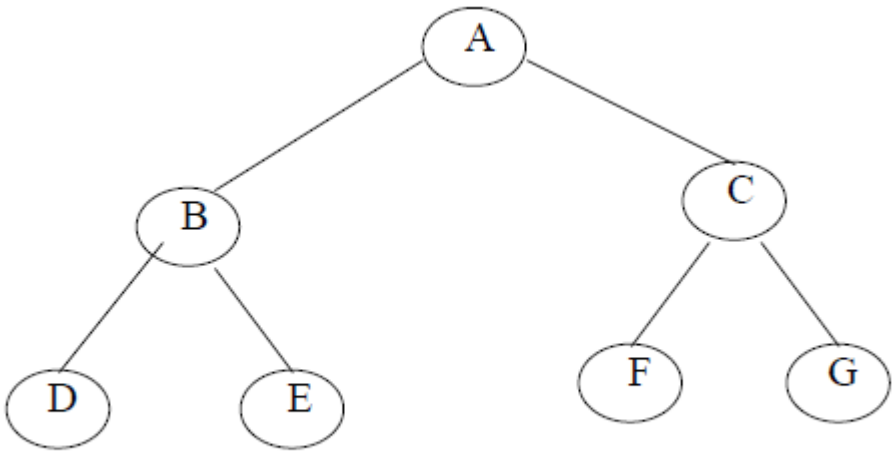

Fig. 8.6. Complete binary tree

A binary tree with n nodes, n > 0, has exactly n – 1 edges. A binary tree of depth d, d > 0, has at least d and at most 2^d – 1 nodes in it. If a binary tree contains n nodes at level l, then it contains at most $2n$ nodes at level l + 1.

Finally, let us discuss in briefly the main difference between a binary tree and ordinary tree is:

no	binary tree	ordinary tree
1	A binary tree can be empty	a tree cannot
2	Each element in binary tree has exactly two sub trees (one or both of these sub trees may be empty).	Each element in a tree can have any number of sub trees.
3	The sub tree of each element in a binary tree are ordered, left and right sub trees.	The sub trees in a tree are unordered.

If a binary tree has only left sub trees, then it is called left skewed binary tree. Fig.8.7 (a) is a left skewed binary tree.

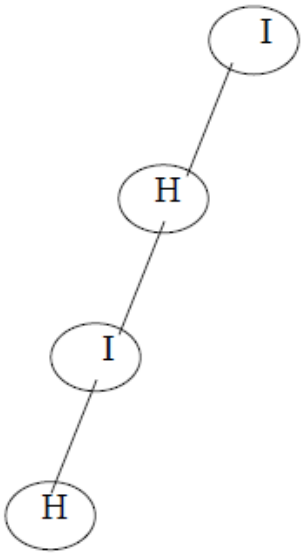

Fig. 8.7(a). Left skewed

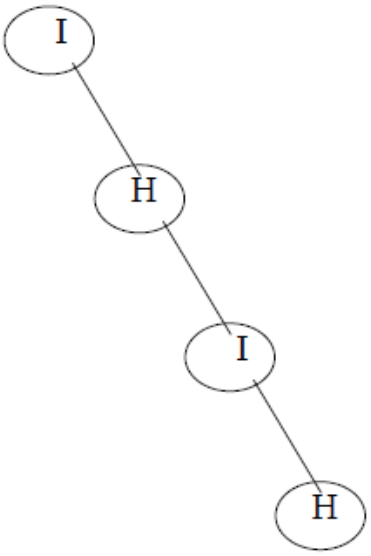

Fig. 8.7(*b*). Right skewed

If a binary tree has only right sub trees, then it is called right skewed binary tree.
Fig. 8.7(b) is a right skewed binary tree.