Trees

The *tree* is a fundamental structure in computer science. Almost all operating systems store files in a *file system* that is a tree or a tree-like structure. Trees are also used to represent a structured document in XML or HTML, to represent arithmetic expressions, and to analyse recursive algorithms using a recursion tree (function call tree), like we did for Merge-Sort.

Definition

Trees can be defined in two ways: nonrecursively and recursively.

Nonrecursive definition of trees A *(rooted) tree* consists of a set of nodes and a set of directed edges between nodes. A rooted tree has the following properties.

- One node is distinguished as the *root*;
- For every node c, except the root, there is exactly one edge (p, c) pointing to c;
- For every node c there is a unique path from the root to c.

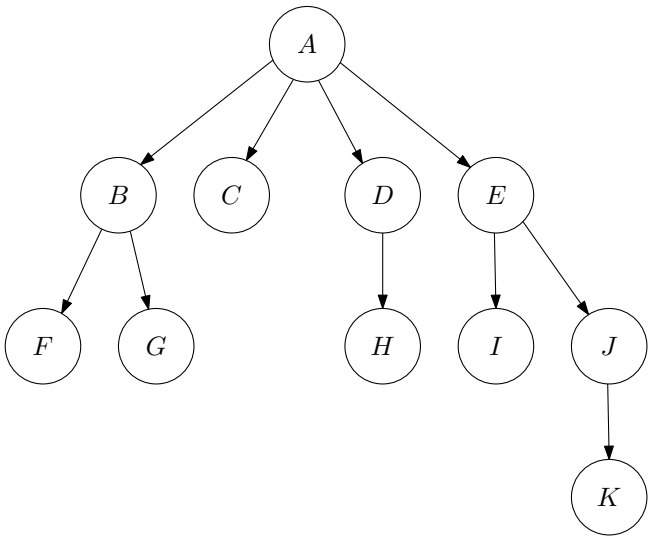
A tree on n nodes has exactly $n-1$ edges, since for every node except the root, there is exactly one edge pointing into it.

A tree can also be defined as a *connected, acyclic* graph.

Recursive definition of trees A (rooted) tree consists of a root, and zero or more subtrees $T_1, T_2, \ldots, T_k$. There is a directed edge from the root to the root of each subtree.

Here, the base case is the case of a root with zero subtrees: a tree consisting of a single node.

Notations A directed edge connects a *parent* to its *child*. A node without children is called a *leaf*. The *depth* of a node c is the length of the path from the root to c. The *height* of a node in a tree is the length of the path from the node to the deepest leaf. Nodes with the same parent are called *siblings*. If there is a path from node u to node v, then u is an *ancestor* of v and v is a *descendant* of u.



Node	Depth	Height
A	0	3
B	1	1
C	1	0
D	1	1
E	1	2
F	2	0
G	2	0
H	2	0
I	2	0
J	2	1
K	3	0

Figure 1: A tree, with height and depth information

Tree Traversals

We can access any node in a tree starting from its root node. A *Tree traversal*, also called *walking the tree*, visits the nodes of the tree sequentially to obtain some information about the tree. Tree traversals are most easily implemented recursively.

We distinguish three types of traversals, depending on when the information in a node is processed:

- *Preorder traversal* means that a node is processed *before* its children;
- *Postorder traversal* means that a node is processed *after* its children;
- *Inorder traversal* means that a node is processed between its left and right child (for binary trees).

For instance, for the tree shown in Fig. 1, the tree traversals would visit the nodes as follows:

- Preorder: A, B, F, G, C, D, H, E, I, J, K
- Postorder: F, G, B, C, H, D, I, K, J, E, A

For the tree in Fig. 2, the three possible orders are:

- Preorder: F, B, A, D, C, E, G, I, H
- Postorder: A, C, E, D, B, H, I, G, F
- Inorder: A, B, C, D, E, F, G, H, I

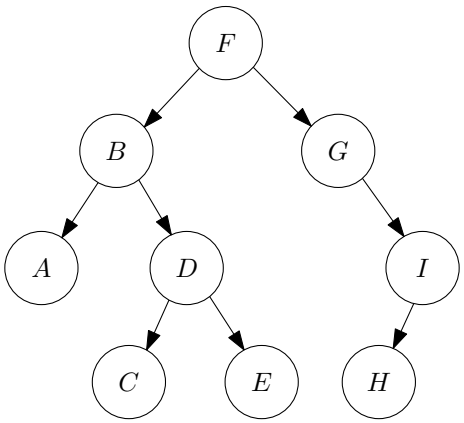


Figure 2: A binary tree