A tree is an acyclic connected graph. In a tree, we designate one node (any node) to be the root node. This is conventionally drawn at the top of the tree: all other nodes are measured by their distance from the root.

1. Because a tree is connected we know there is a path from every node to the root.
2. Because a tree is acyclic we know that there is only one path to the root.
3. Note that an acyclic disconnected graph is known as a forest: it is composed of many trees.
4. The parent of a given node v is the node that is connected directly to v but is closer to the root. (Thus the root has no parent).
5. u is a parent of v if and only if v is a child of u.
6. If a node has no children, it is called a leaf node.

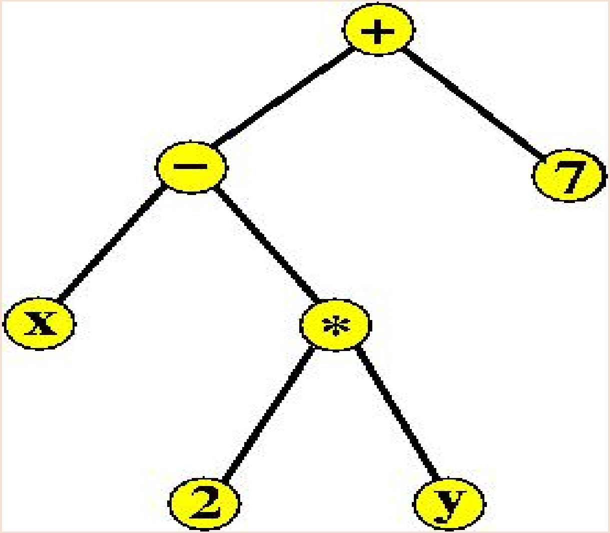
7. The distance between two nodes u and v on a tree is the length of the (unique) path between them (i.e. the number of edges in that path).
8. A node is at level *n* if its distance from the root is *n*.
9. The height (or depth) of a node is its distance from the root node.
10. The height (or depth) of a tree is the maximum depth or height of any of its nodes.
11. An n-ary tree is one where each node has a maximum of *n* children. For n=1, this is called a unary tree, for n=2 a binary tree, for n=3 a ternary tree.
12. A complete n-ary tree is one where at each level we have the maximum possible number of nodes (i.e. 2^L where L is the level).

Example: Expression Tree

In a simple mathematical expression such as $a + b$, we call a and b the operands and the sign $+$ the operator. Here $+$ is a binary operator, because we add <u>two</u> things together: we can view $+$ as a “black box” with two inputs and one output.
A general mathematical expression (involving $+, -, \times, /, \ldots$) can be represented as a binary tree: In such a tree, the operands are the leaves (not necessarily at the same level), while the operators are internal nodes. Since (standard) mathematical operators are binary, these trees are (normally) binary.

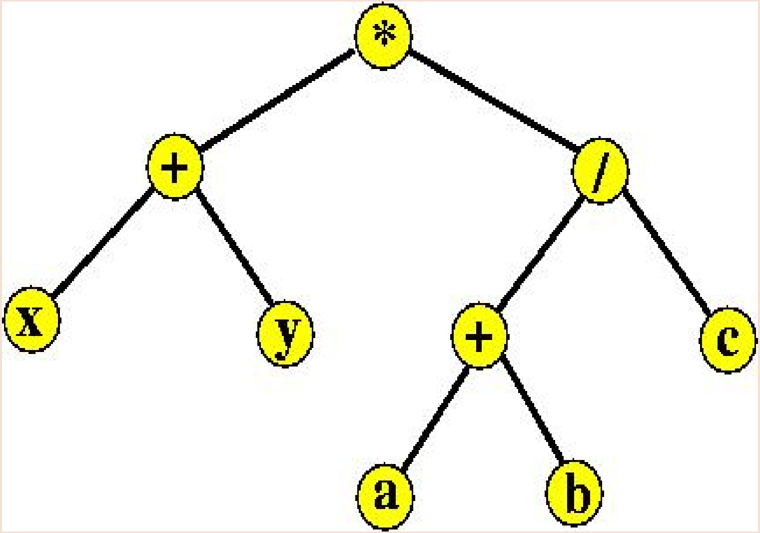
Example: Expression Tree

$x - 2y + 7$



Example: Expression Tree

$(x + y) \times ((a + b)/c)$



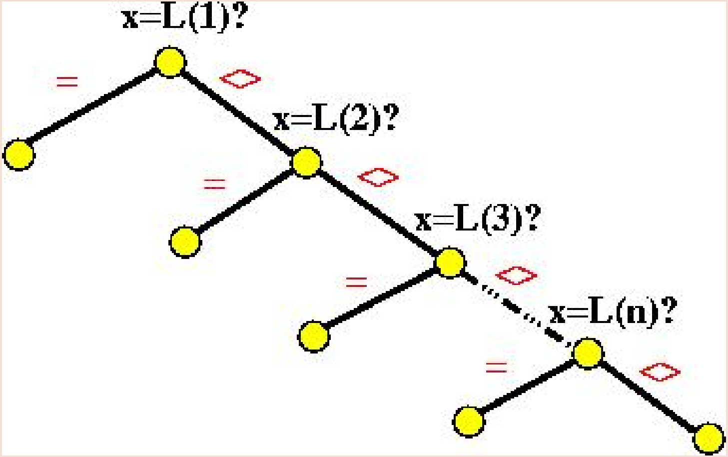
Example: Decision Tree

In an algorithm, the execution path is dictated by the evaluation of conditions (`IF` statements, conditions in `WHILE` loops, etc.). Since a condition is a boolean expression (i.e. it is either TRUE or FALSE), we can represent the condition as a node, and the result of its evaluation as two edges (TRUE/FALSE) leading to the next step in the algorithm. In the sequential search algorithm:

1. Nodes correspond to comparisons.
2. Edges correspond to the the resulting decisions.
3. Leaves correspond to (output) results, and program termination.

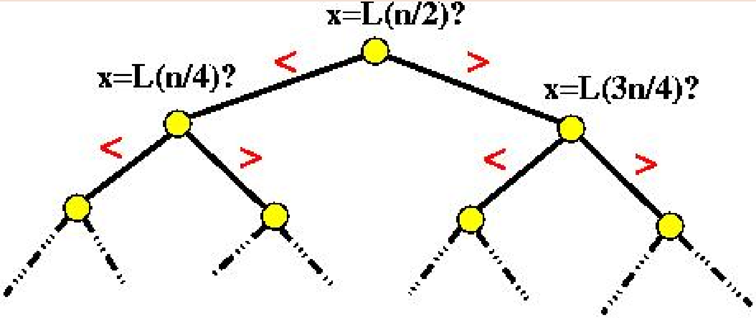
Example: Decision Tree

Linear/Sequential Search



Example: Decision Tree

Binary Search



Here all nodes correspond to comparisons, and also to (output) results, and program termination.