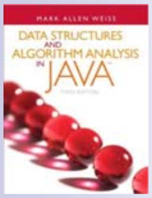
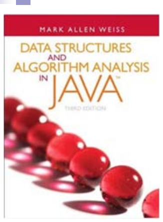


FLORIDA INTERNATIONAL UNIVERSITY

Trees
(II)

Dr. Antonio L. Bajuelos

FIU School of Computing & Information Sciences



COP-3530 - Data Structures

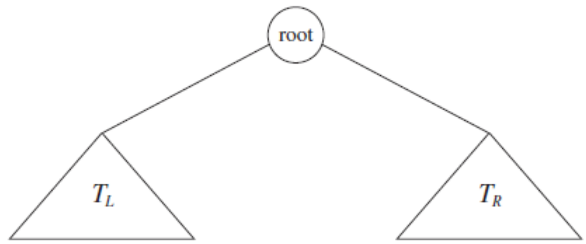
Module #3: Trees (part II)

Outline:

- Binary trees.
- Some bounds on binary trees.
- An application of binary trees.
- Binary tree traversals.

Binary Trees

- **Definition:** A **binary tree** is a **tree** in which **every node has at most two children**.

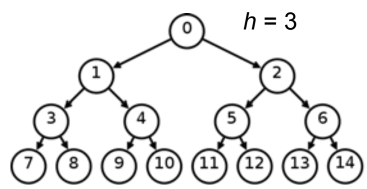


3

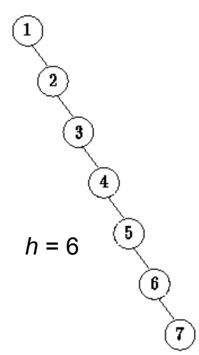
Binary Trees. Some bounds

- **height** - longest path from root to leaf (count edges)

h = 3



- For **binary tree** of height ***h***:
  - **max** number of **leaves**: 2^h
  - **max** number of **nodes**: 2^{h+1} -1
    ($\sum_{k=0}^{n-1} 2^k = 1 + 2 + 4 + \ldots + 2^{n-1} = 2^n - 1$)
  - **min** number of **leaves**: 1
  - **min** number of **nodes**: h +1



h = 6

- For N nodes, **min height** is **O(logN)** and we want to avoid height = **O(N)** (**max height**)

Implementation of Binary Trees

- The declaration of **tree nodes** is similar in structure to that for doubly linked lists.
- A **node** is a **structure** consisting of the element information plus two references (**left** and **right**) to other **nodes**.
- **Binary Tree Node Class**

```
class BinaryNode
{
    Object element;        // The data in the node
    BinaryNode left;       // Left child
    BinaryNode right;      // Right child
}
```

5

Binary Search Trees. Java Code

- **Binary Node Class**

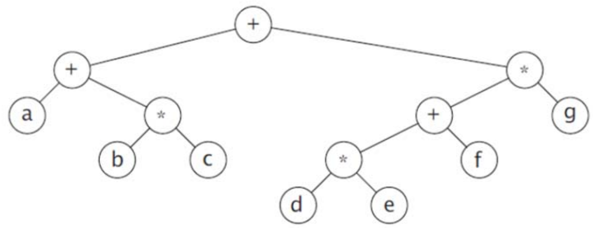
```
private static class BinaryNode<AnyType>
{
    AnyType element; // The data in the node
    BinaryNode<AnyType> left; // Left child
    BinaryNode<AnyType> right; // Right child

    // Constructors
    BinaryNode( AnyType theElement )
    {  this( theElement, null, null );  }

    BinaryNode( AnyType theElement, BinaryNode<AnyType> lt,
                    BinaryNode<AnyType> rt )
    { element = theElement; left = lt; right = rt; }
}
```

6

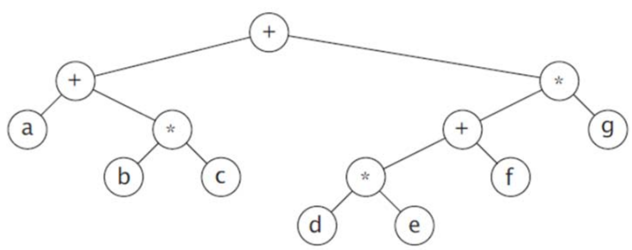
Expressions Trees (example of Binary Trees)



- The **leaves** of an expression tree are **operands** and the other (internal) nodes contain **operators**.
- The **expression trees are binary**, (in the majority of cases all the operators are binary).
- It is also possible for a node to have only one child, as is the case with the **unary minus** operator.

7

Expressions Trees (example of Binary Trees)

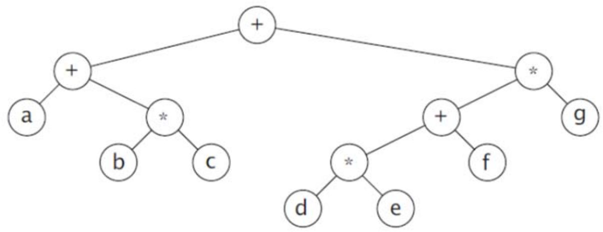


- How to evaluate an expression tree?
  - Applying the operator at the root to the values obtained by **recursively evaluating the left and right subtrees**.
  - Left subtree: a + (b * c)
  - Right subtree: ((d * e) + f) * g
  - The full tree represents:

    (a + (b * c)) + (((d * e) + f) * g)

8

Binary Trees Traversals (inorder, postorder, and preorder traversals)



- ***Inorder* traversal**: (left, node/root, right)

  (a + (b * c)) + (((d * e) + f) * g) ***infix notation***

- ***Postorder* traversal**: (left, right, node/root)

  a b c * + d e * f + g * + ***postfix notation***

- ***Preorder* traversal**: (node/root, left, right)

  + + a * b c * + * d e f g ***prefix notation***

9