UNIVERSITI TEKNOLOGI MARA

CSC508 Data Structures

Topic 8 : Tree 1

Binary Tree

Compiled & edited by: Zahid Zainal

UNIVERSITI TEKNOLOGI MARA

Recap

- Recursion Definition
- Recursive Method
- Types of Recursion
- Recursion Application
- Recursion vs Loop

Topic Structure

- Tree Definition
- Tree Terminologies
- Binary Tree
- Binary Tree Representation
- Binary Tree Traversal

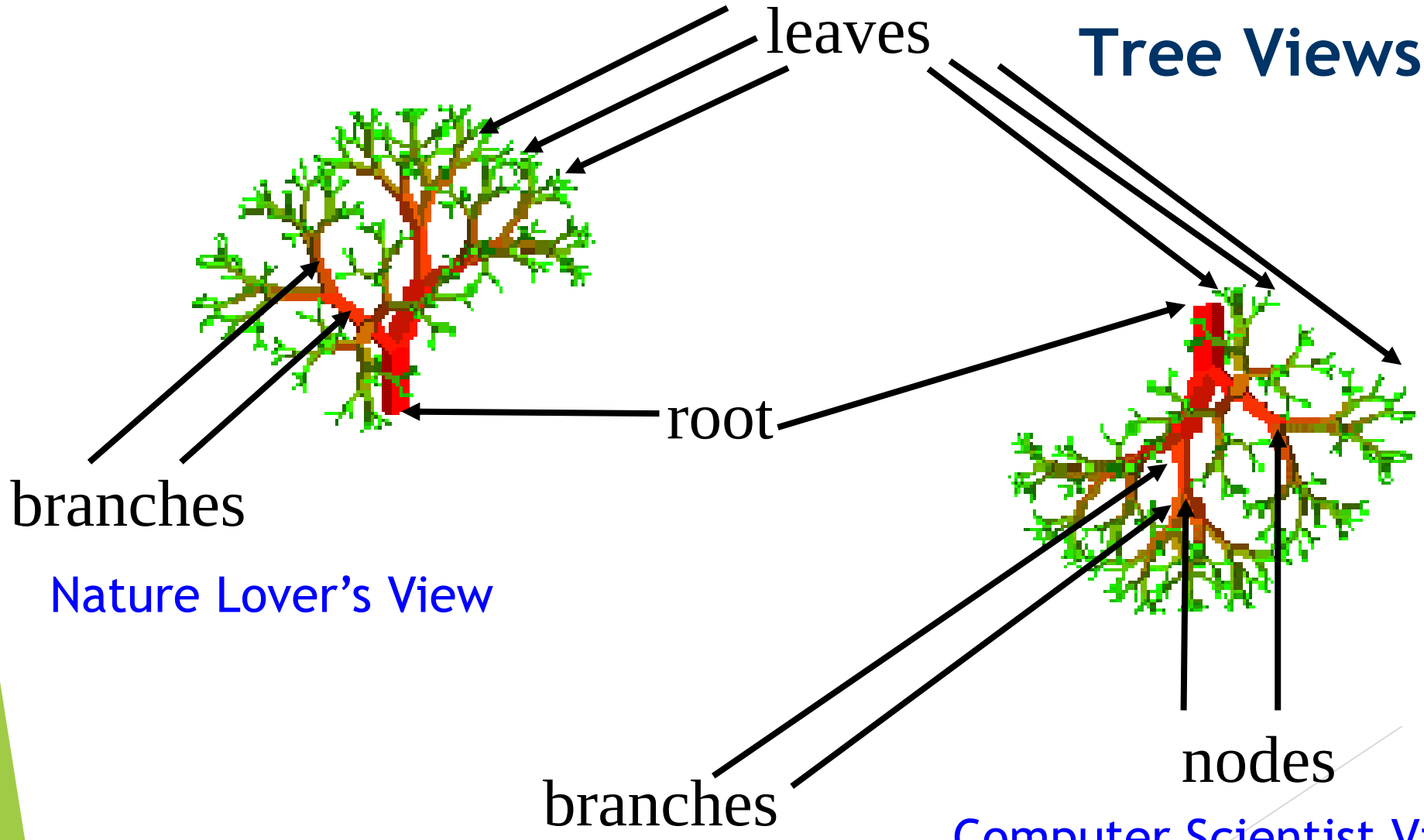
Learning Outcomes

- At the end of this lesson, students should be able to:

List vs Tree

- Linear lists are useful for serially ordered data
  - (e0, e1, e2, …, en-1)
  - Days of week, months in a year, etc.
  - Purchased items in online shopping cart
  - Students in a class
- Trees are useful for hierarchically organized data items, which have ancestors and descendants
  - Corporation structure or organization chart
  - Family tree
  - File-system directories and files

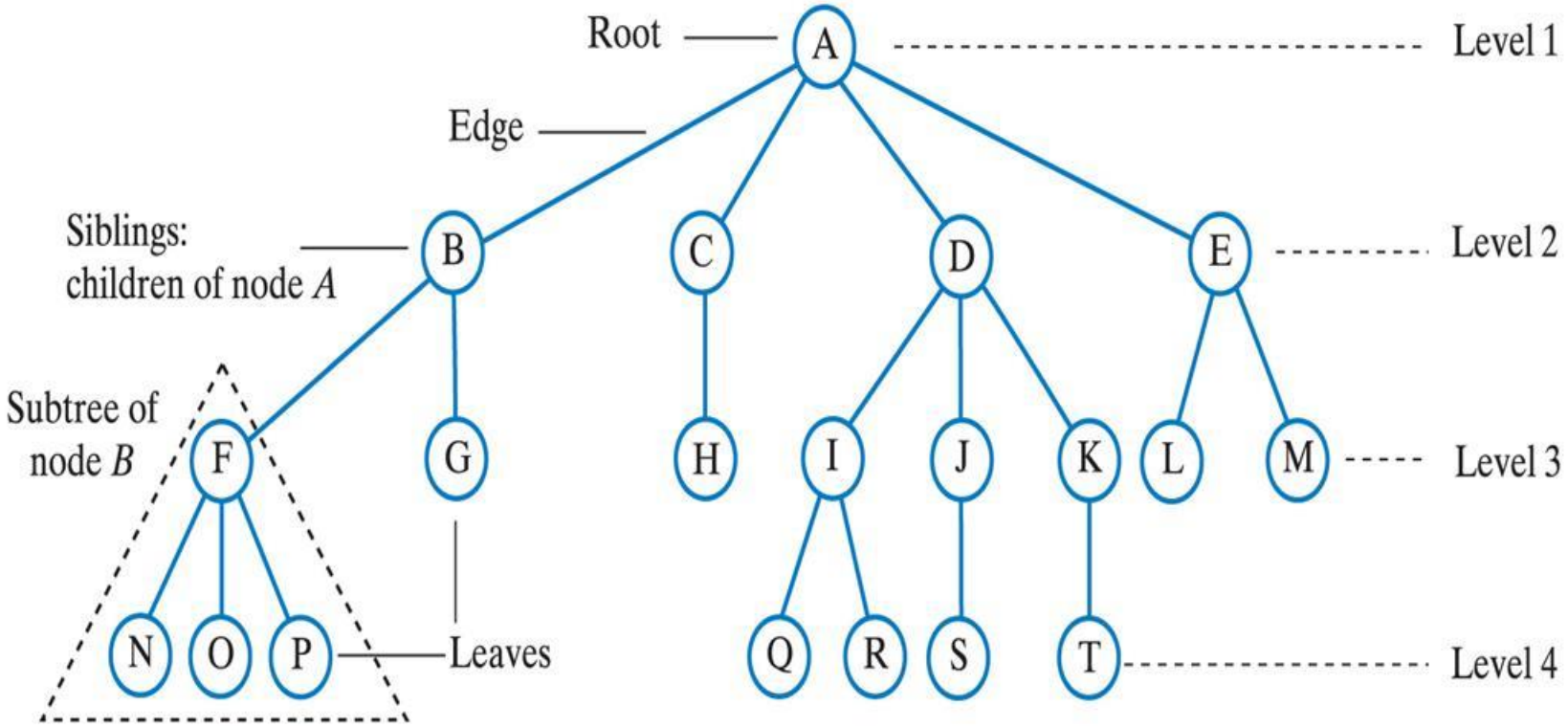
Tree



Tree Views

Tree Definition

- A tree is a set of nodes connected by edges to indicate a hierarchical relationship among the nodes
- Nodes are arranged in levels
  - Top level is a single node called the root
  - Nodes at a given level are children of nodes of the previous level
  - A node that has children is called their parent (all nodes have exactly one parent, except root)
  - Nodes with the same parent are siblings
  - Nodes that have no children are leaves



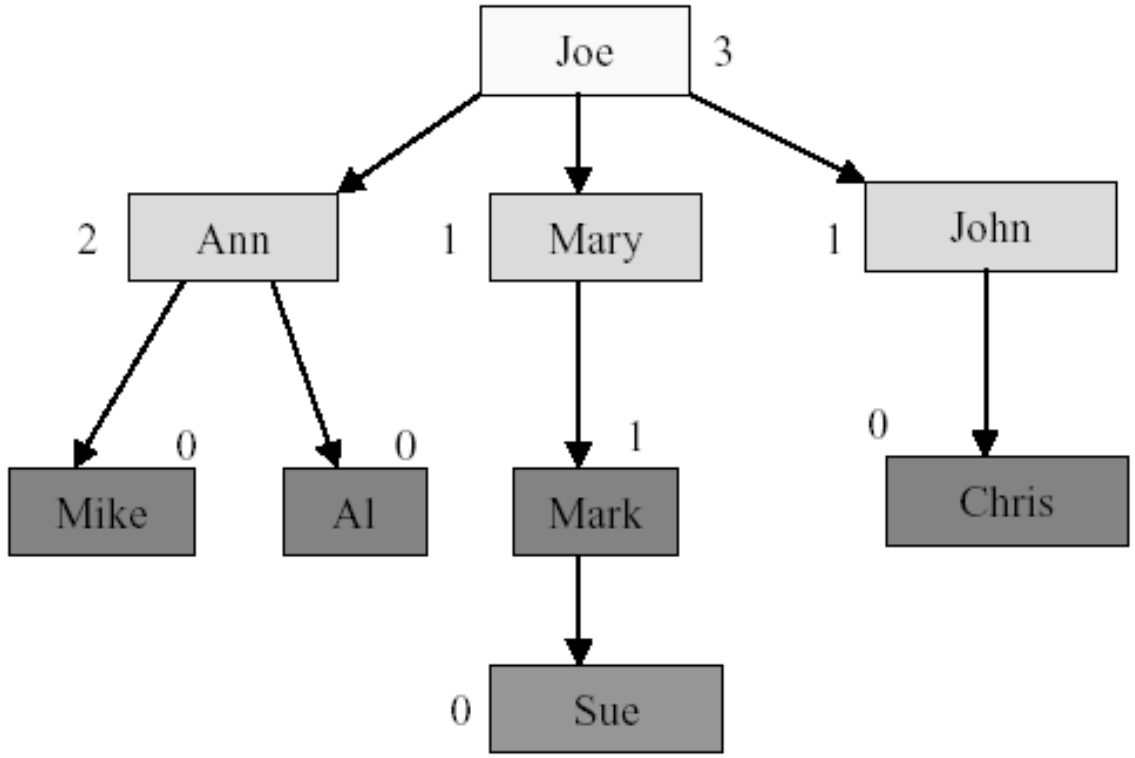
Some reference may start the level at 0

Tree Terminologies

- A subtree of a node is a tree rooted at a child of that node
- A node is reached from the root by a path
  - The length of a path is the number of edges that compose it
- The height of a tree is the number of levels in it
  - Another definition: number of nodes on the longest path from the root to a leaf
  - Height of tree T = 1 + height of tallest subtree of T

Tree Terminologies (cont.)

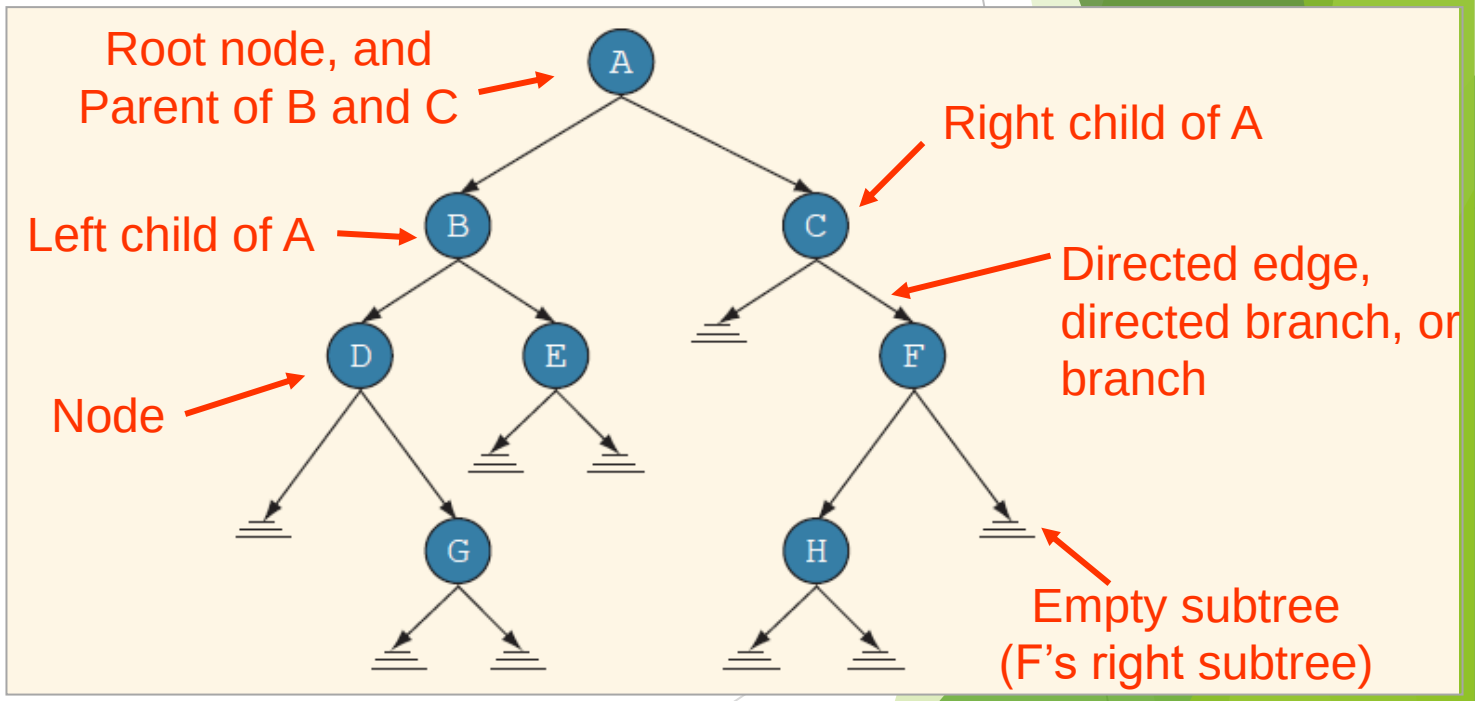
- Node degree is the number of children it has
- Tree degree is the maximum degree of all its nodes' degrees



tree degree = 3

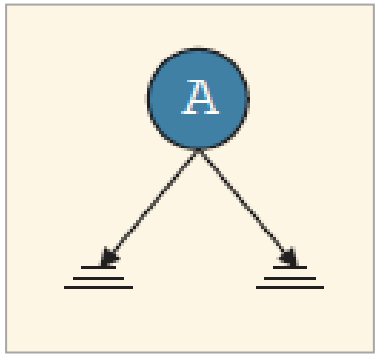
Binary Tree

- Binary tree is a type of tree where every node has at most two children
- A binary tree, T is either empty, or
  - *T* has two sets of nodes, L_T and R_T, called the left subtree and right subtree of *T*, respectively
  - L_T and R_T are binary trees themselves



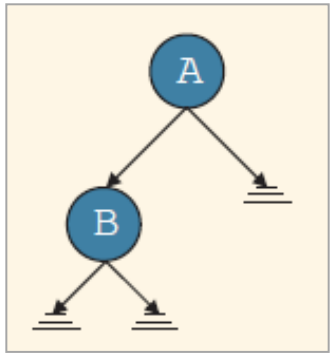
Binary Tree Cases

- Binary tree with one node
  - The root node of the binary tree = A
  - L_A = empty
  - R_A = empty

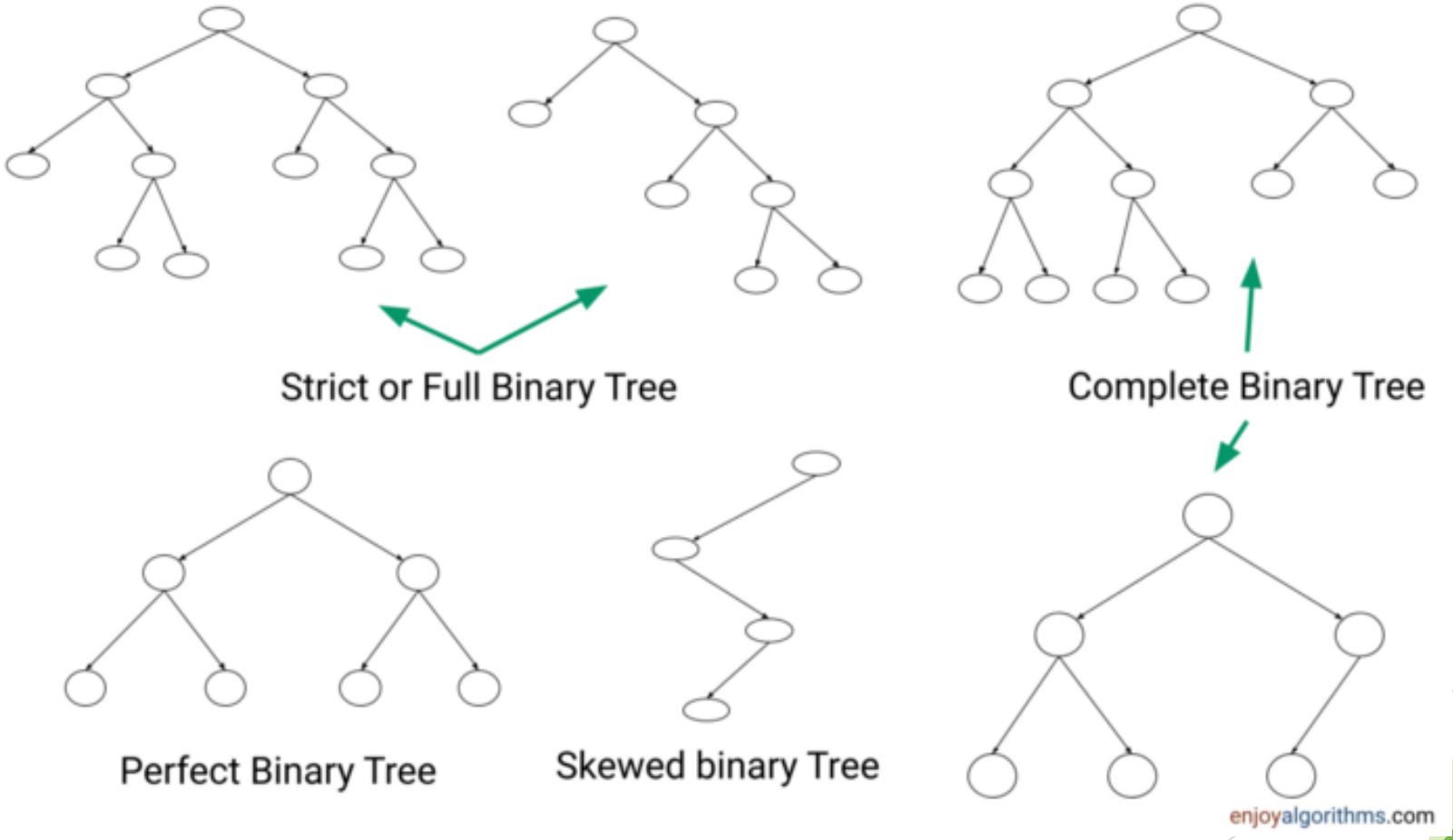


Binary Tree Cases

- Binary tree with two nodes.
  - The root node of the binary tree = A
    - L_A = B
    - R_A = empty
  - The root node of L_A = B
    - L_B = empty
    - R_B = empty



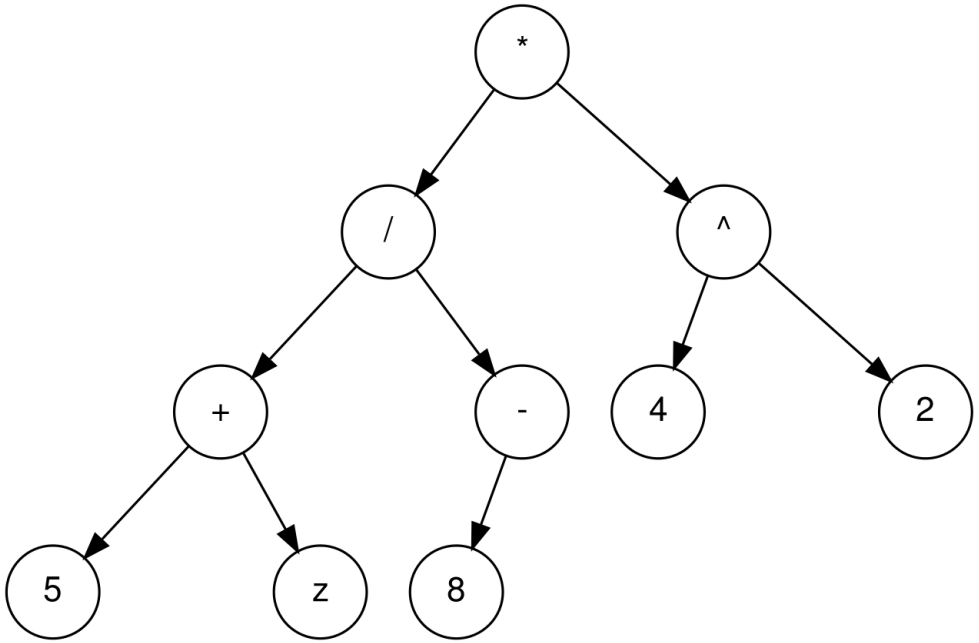
Binary Tree Types



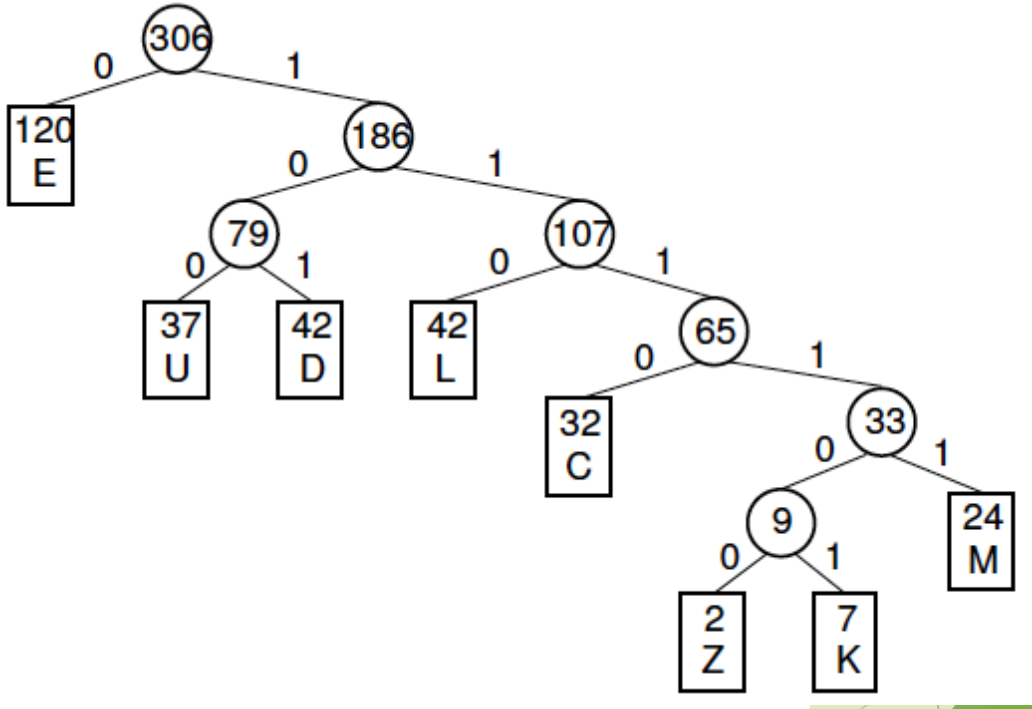
Binary Tree Types (cont.)

- A **full binary tree** (sometimes **proper binary tree** or 2-tree) is a tree in which every node has zero or two children except for the leaves.
- A **perfect binary tree** is a full binary tree in which all leaves are at the same depth.
- A **complete binary tree** is a binary tree where all the levels of the tree are filled completely, except the lowest level nodes which are filled from as left as possible

Examples of Binary Tree



Expression Tree



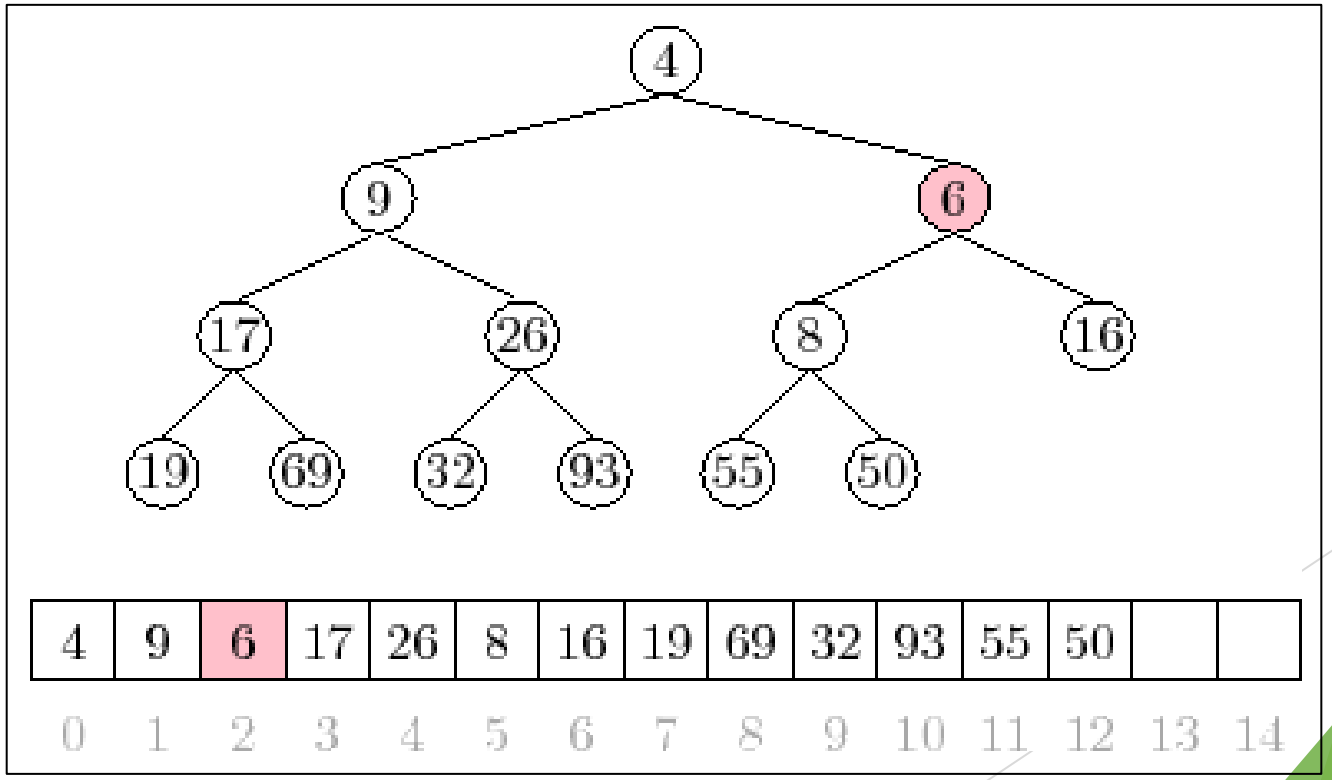
Huffman Code Tree

Binary Tree Representation

- Array representation
  - Storing tree elements in an array in a way that represents the relationship between nodes
- Linked representation
  - Using pointers to link between a parent node and its children nodes

Binary Tree - Array

- Nodes are stored in array with from index 0 until (n-1), where n is the total number of nodes.



Binary Tree - Linked

- The most popular way to represent a binary tree
- Each tree node is represented by a structure with:
  - Two link fields (left and right), to point to left and right binary subtrees
  - Node's own info in the form of data field(s)
  - May add additional fields, like subtree height for each node, pointer to parent node, etc.

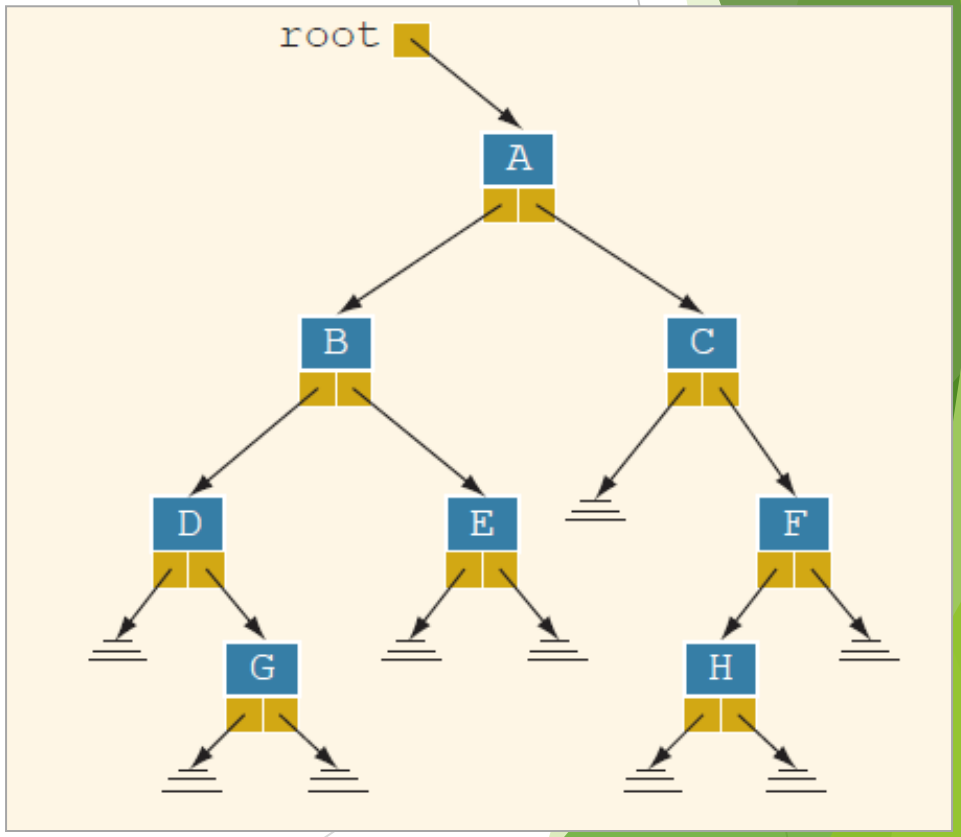
```
protected class BinaryTreeNode {
    Object info;
    BinaryTreeNode llink;
    BinaryTreeNode rlink;
}
```

Link to left child node

Link to right child node

Binary Tree - Linked

- A variable storing the address of the root node of the binary tree is used outside the tree in a pointer variable



Common Binary Tree Operations

- Determine the height
- Determine the number of nodes and leaves
- Add or delete nodes to/from the tree
- Search the tree to find particular data
- Traverse the tree in various ways
- Display (print or draw) the binary tree
- Empty the tree
- Make a copy of the tree
- Determine if two binary trees are identical
- and many other…

Height of a Binary Tree

- Recursive algorithm to find height of binary tree: (height(p) denotes height of binary tree with root p):
  - If the binary tree is empty, height is 0
  - If the binary tree is nonempty:
    - Find the height of left subtree & right subtree
    - Find the maximum of these two heights and add 1

Height of a Binary Tree (cont.)

```
private int height(BinaryTreeNode p) {
    if(p == NULL)
        return 0;
    else
        return 1 + max(height(p.llink), height(p.rlink));
}
```

Binary Tree Traversal

- Many binary tree operations , such as item insertion, deletion, and lookup, require the tree to be traversed
- Each node of the tree is **visited** exactly once
  - During node visit, actions (such as making a copy, printing, evaluating, etc.) with respect to that node are taken
- Must start with the root, and then
  - Visit the node first, *or*
  - Visit the subtree(s) first, *or*
  - Visit siblings, level by level
- Four different traversals
  - Pre-order, In-order, Post-order, Level-order

Traversal Algorithms

- Pre-order traversal
  - Visit the node
  - Traverse the left subtree
  - Traverse the right subtree
- In-order traversal
  - Traverse the left subtree
  - Visit the node
  - Traverse the right subtree
- Post-order traversal
  - Traverse the left subtree
  - Traverse the right subtree
  - Visit the node

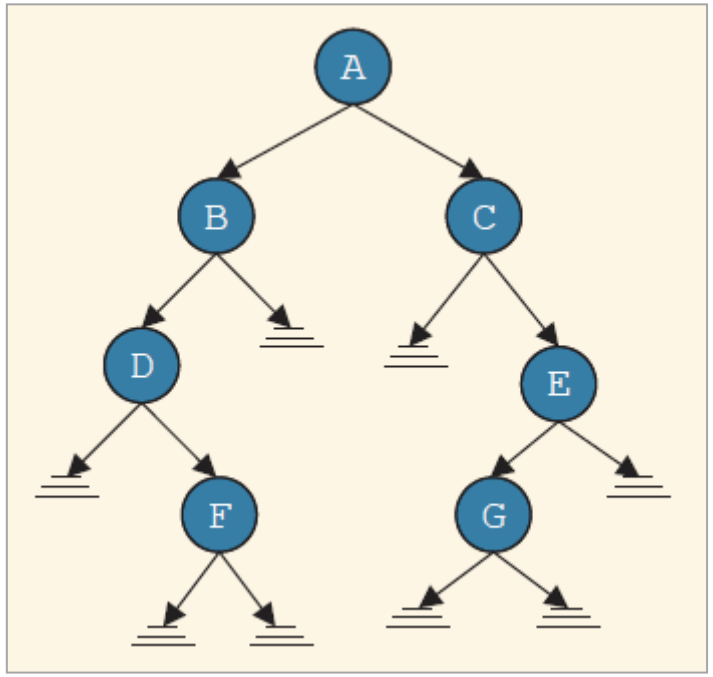
These traversal algorithms are all recursive

Traversal Algorithms (cont.)

- Level-order traversal
  - Add the root node to a queue of nodes
  - Repeat the following until the queue gets empty:

    Take out next node in the queue, visit it, and add its children to the queue (left first, then right)

Traversal Examples



- Pre-order visit sequence: A B D F C E G
- In-order visit sequence: D F B A C G E
- Post-order visit sequence: F D B G E C A
- Level-order visit sequence: A B C D E F G

Pre-order Traversal

```
private void preorder(BinaryTreeNode p){
    if(p != NULL){
        System.out.println(p.info + " ");
        preorder(p.llink);
        preorder(p.rlink);
    }
}
```

In-order Traversal

```
private void inorder(BinaryTreeNode p) {
    if(p != NULL) {
        inorder(p.llink);
        System.out.println(p.info + " ");
        inorder(p.rlink);
    }
}
```

Post-order Traversal

```
private void preorder(BinaryTreeNode p){
    if(p != NULL){
        preorder(p.llink);
        preorder(p.rlink);
        System.out.println(p.info + " ");
    }
}
```

Level-order Traversal

```
private void levelOrder(){
    Queue q;
    q.enqueue(root);
    BinaryTreeNode p;

    while(!q.isEmpty()){
        p = q.dequeue();
        System.out.println(p.Info);

        if (p.llink != null)
            q.enqueue(p.llink);
        if (p.rlink != null)
            q.enqueue(p.rlink)
    }
}
```

Summary

- A tree is a set of nodes connected by edges to indicate a hierarchical relationship among the nodes
- Binary tree is a type of tree where every node has at most two children
- Three types of binary tree : Full binary tree, Perfect binary tree, and Complete binary tree
- Two ways to represent binary tree : Array and Liked representation
- Four traversal algorithms : Pre-order, In-order, Post-order, and Level-order

Next Topic…

- Binary Search Tree

References

- Carrano, F. & Savitch, W. 2005. *Data Structures and Abstractions with Java, 2nd ed. Prentice-Hall.*
- Malik D.S, & Nair P.S., Data Structures Using Java, Thomson Course Technology, 2003.
- Rada Mihalcea, CSCE 3110 Data Structures and Algorithm Analysis notes, U of North Texas.