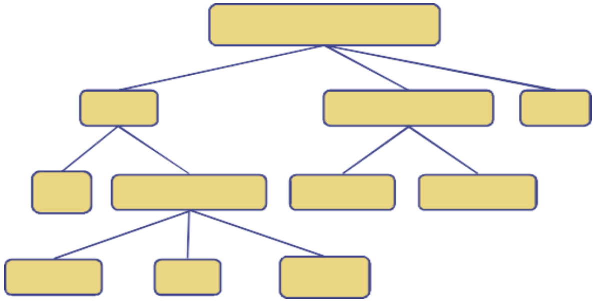
Tree and Tree Traversal

Charles Aunkan Gomes
Lecturer, Dept. of CSE
United International University
charles@cse.uiu.ac.bd

What is a Tree?

- A connected acyclic graph is a tree
- In computer science, a tree is an abstract model of a hierarchical structure
- A tree consists of nodes with a parent-child relationship
- Applications:
  - Organizational charts
  - File systems
  - Programming environments



What is a Tree?

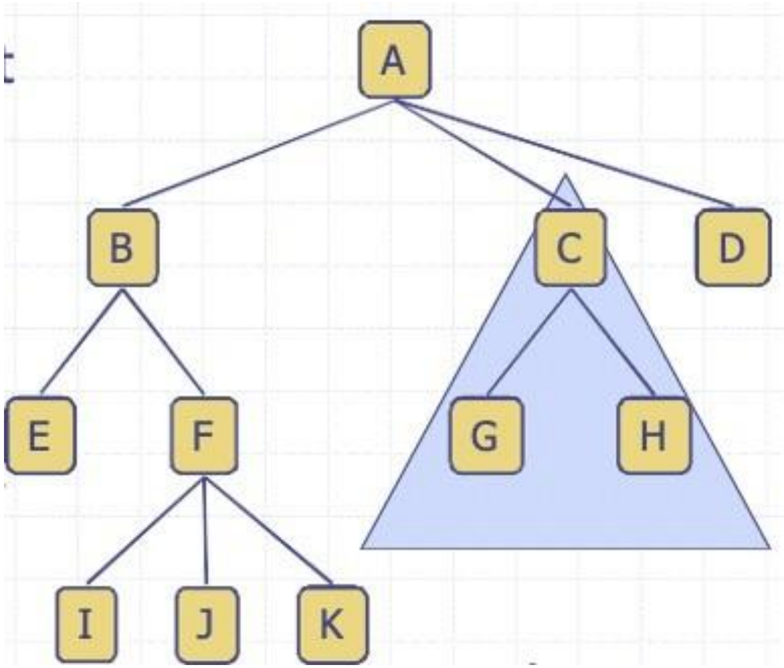
- *A connected acyclic graph is a tree.*

- A tree T is a set of nodes in a parent-child relationship with the following properties:

  - T has a special node r, called the ***root*** of T, with no parent node
  - Each node v of T, different from r, has a unique parent node u

Tree Terminology

- Root: node without parent (A)
- Internal node: node with at least one child (A, B, C, F)
- External node (Leaf): node without children (E, I, J, K, G, H, D)
- Ancestors of a node: parent, grandparent, grand-grandparent, etc.
- Descendants of a node: child, grandchild, grand-grandchild, etc.



Depth and Height

- The depth of a node v can be recursively defined as follows
  - If v is the root, then the depth of v is 0.
  - Otherwise, the depth of v is one plus the depth of the parent of v

Algorithm

```
depth(T, v)
if T.isRoot(v) then
      return 0
  else
     return 1 + depth(T, T.parent(v))
```

- Running time: $O(1 + d_v)$, d_v is depth of v in T
- In worst case $O(n)$, n is the number of nodes in T

Depth and Height

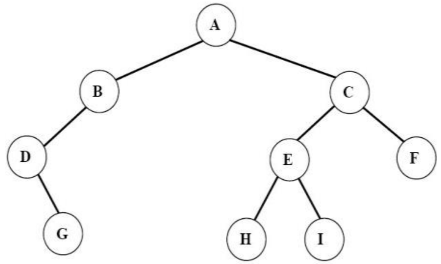
- The ***height*** of a node v can be recursively defined as follows
  - If v is a leaf node, then the height of v is 0.
  - Otherwise, the height of v is one plus the maximum height of a child of v
- The height of a tree T is the height of the root of T
- The height of a tree T is equal to the maximum depth of a leaf node of T

Depth and Height

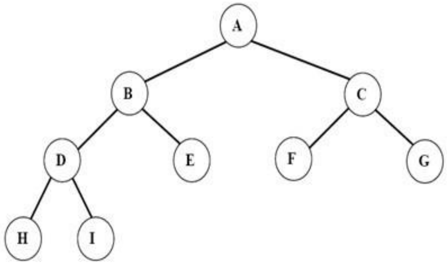
```
Algorithm height(T, v)
    if T.isLeaf(v) then  return
        0
    else
        h = 0
        for each w εT.children(v) do
                h = max(h, height(T, w))
    return 1 + h
```

Ordered Trees

- A tree is ordered if there is a linear ordering defined for each child of each node.
- A binary tree is an ordered tree in which every node has at most two children.
- If each node of a tree has either zero or two children, the tree is called a proper binary tree.



Binary Tree



Proper Binary Tree

Traversal of Trees

- A traversal of a tree T is a systematic way of visiting all the nodes of T
- Traversing a tree involves visiting the root and traversing its subtrees
- There are the following traversal methods:
  - Preorder Traversal
  - Postorder Traversal
  - Inorder Traversal (of a binary tree)

Construct a Binary Tree only from Preorder, Inorder or Postorder

- Preoder: $f \rightarrow b \rightarrow g \rightarrow i \rightarrow h$

- No Unique Tree if only one sequence is specified.
- Need at least two traversal sequences to construct a unique tree.

Construct a Binary Tree from *Preorder & Inorder*

- Preoder: f → b → a → d → c → e → g → i → h
- Inorder: a → b → c → d → e → f → i → g → h

Construct a Binary Tree from *Postorder & Inorder*

- Postoder: a → c → e → d → b → i → h → g → f
- Inorder: a → b → c → d → e → f → i → g → h

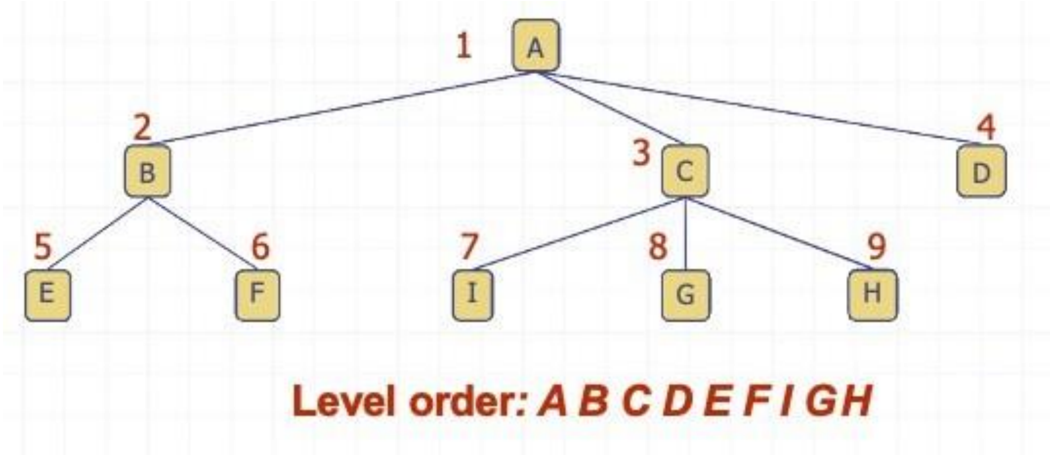
Construct a Binary Tree from *Preorder & Postorder*

- Preoder: f → b → a → d → c → e → g → i → h
- Postoder: a → c → e → d → b → i → h → g → f

- No unique Tree!!
- Need at least one Inorder traversal sequence to construct Unique tree.

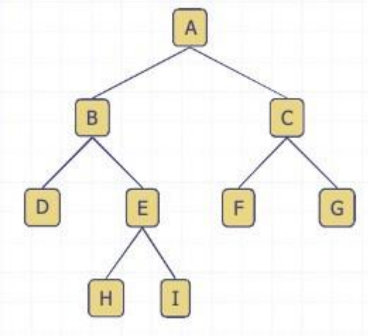
Level Order Traversal

•In a level order traversal, every node on a level is visited before going to a lower level



(Proper) Binary Tree

- A (proper) binary tree is a tree with the following properties:
  - Each internal node has two children
  - The children of a node are an ordered pair



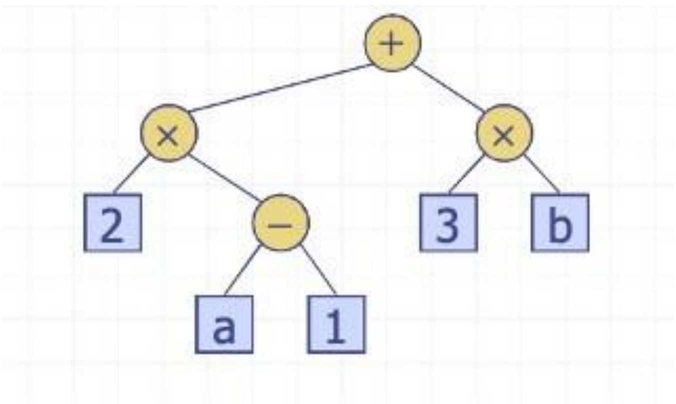
- We call the children of an internal node left child and right child
- Alternative recursive definition:
  - A (proper) binary tree is either
    - a tree consisting of a single node, or
    - a tree whose root has an ordered pair of children, each of which is a proper binary tree

Application:

- arithmetic expressions
- decision processes
- searching

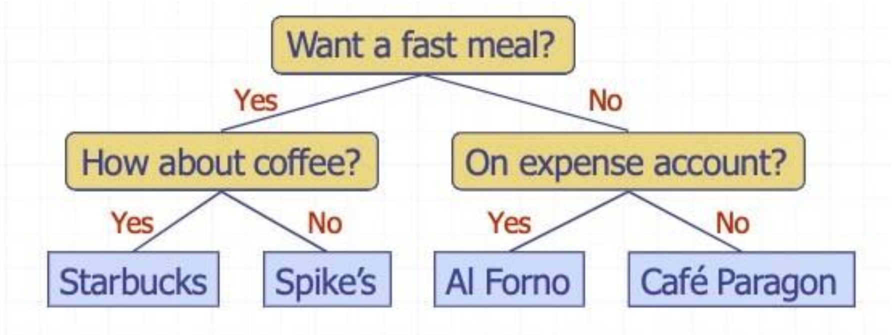
Arithmetic Expression Tree

- Binary tree associated with an arithmetic expression
  - internal nodes: operators
  - external nodes: operands
- Example: arithmetic expression tree for the expression (2 * (a - 1) + (3 * b))



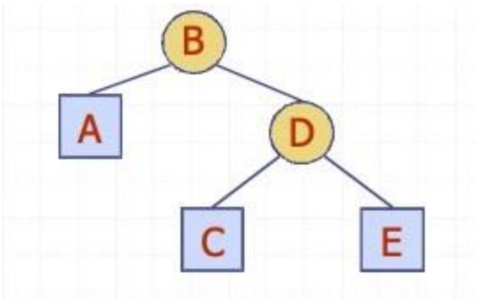
Decision Tree

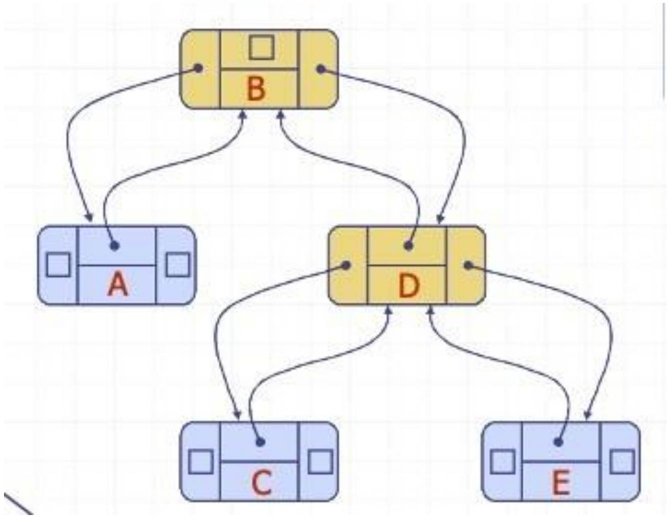
- Binary tree associated with a decision process
  - internal nodes: questions with yes/no answer
  - external nodes: decisions
- Example: dining decision



Linked Structure for Binary Trees

- A node is represented by an object storing
  - Element
  - Parent node
  - Left child node
  - Right child node





Codes for Creation of Binary Trees

A binary tree can be created recursively.
The program will work as follow:

- Read a data in *x*.
- Allocate memory for a new node and store the address in pointer *p*.
- Store the *data x* in the *node p*.
- Recursively create the left subtree of *p* and make it the left child of *p*.
- Recursively create the right subtree of *p* and make it the right child of *p*.

Codes for Creation of Binary Trees

```
#include<stdio.h>
typedef struct TreeNode {
	struct TreeNode *left;
	int data;
	struct TreeNode *right;
} TreeNode;
```

Codes for Creation of Binary Trees

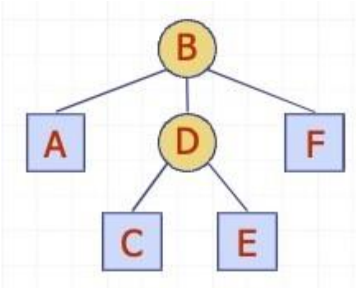
```
TreeNode * createBinaryTree( ){
        TreeNode *p;
        int x;
        printf("Enter data(-1 for no data): ");
        scanf("%d", &x);
        if(x == -1) return NULL;
        //create current node
        p = (TreeNode*) malloc(sizeof(TreeNode));
        p->data = x;
        //recursively create left and right subtree
        printf("Enter left child of %d: \n", x);
        p->left = createBinaryTree ( );
        printf("Enter right child of %d: \n",x);
        p->right = createBinaryTree ();
        return p; }
```

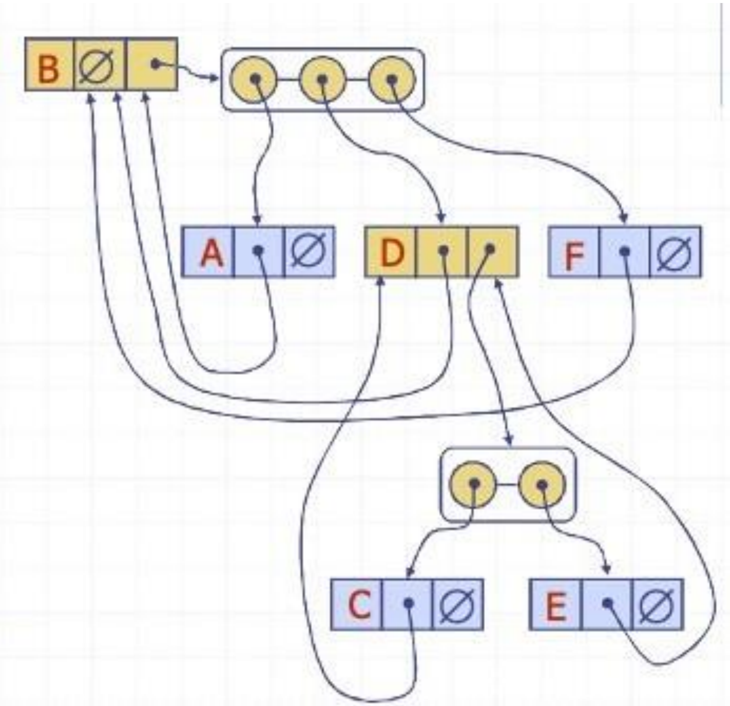
Codes for Creation of Binary Tree Traversal

```
void preorder(TreeNode *t) {
 if(t != NULL) {
        printf("\n%d", t->data);
        preorder(t->left);
        preorder(t->right);
        }
}
int main() {
 TreeNode *root;
 root = createBinaryTree ( );
 printf("\nThe preorder traversal of tree is: \n");
 preorder(root);
}
```

Linked Structure for General Trees

- A node is represented by an object storing
  - Element
  - Parent node
  - Children Container: Sequence of children nodes






THANK YOU