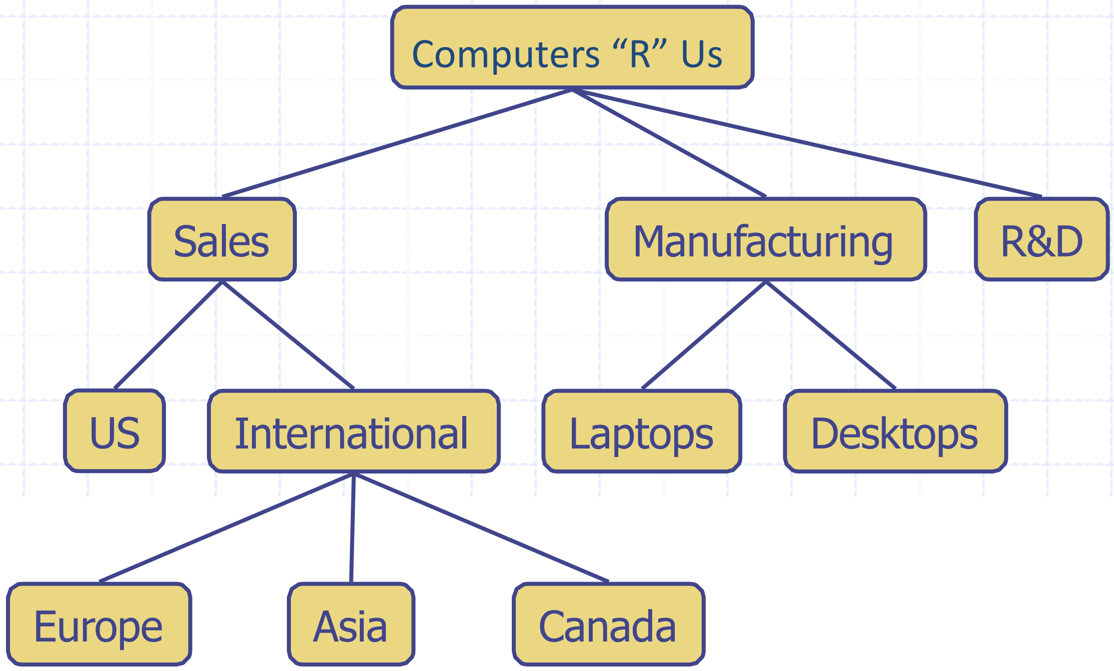
What is a Tree?

A tree is an abstract model of a hierarchical structure

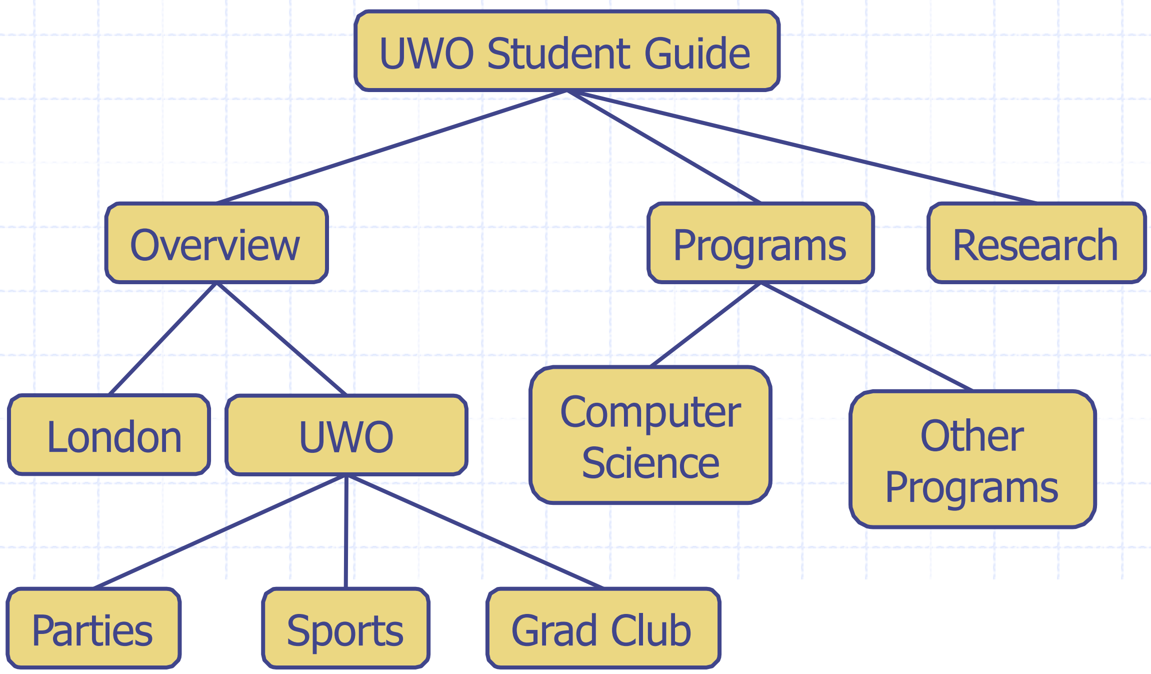
Applications

Organization of a company



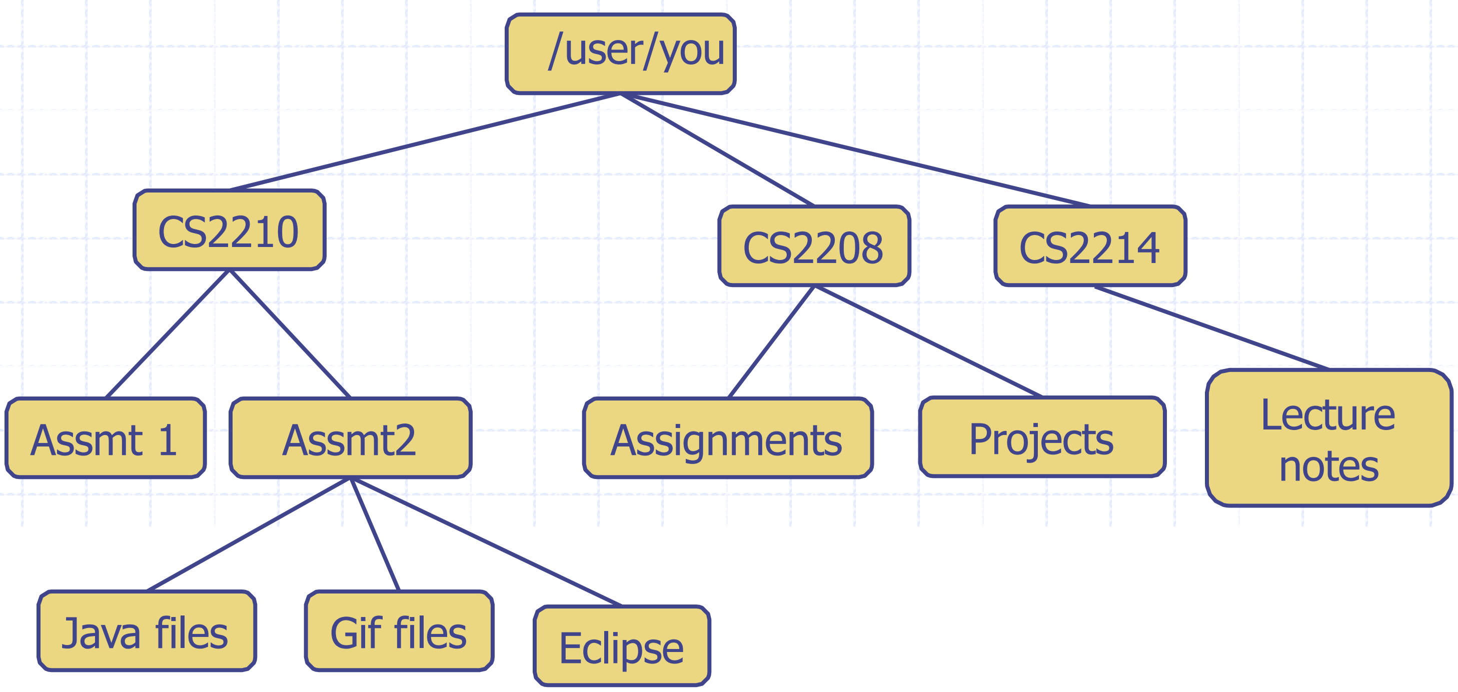
Applications

Table of contents of a book



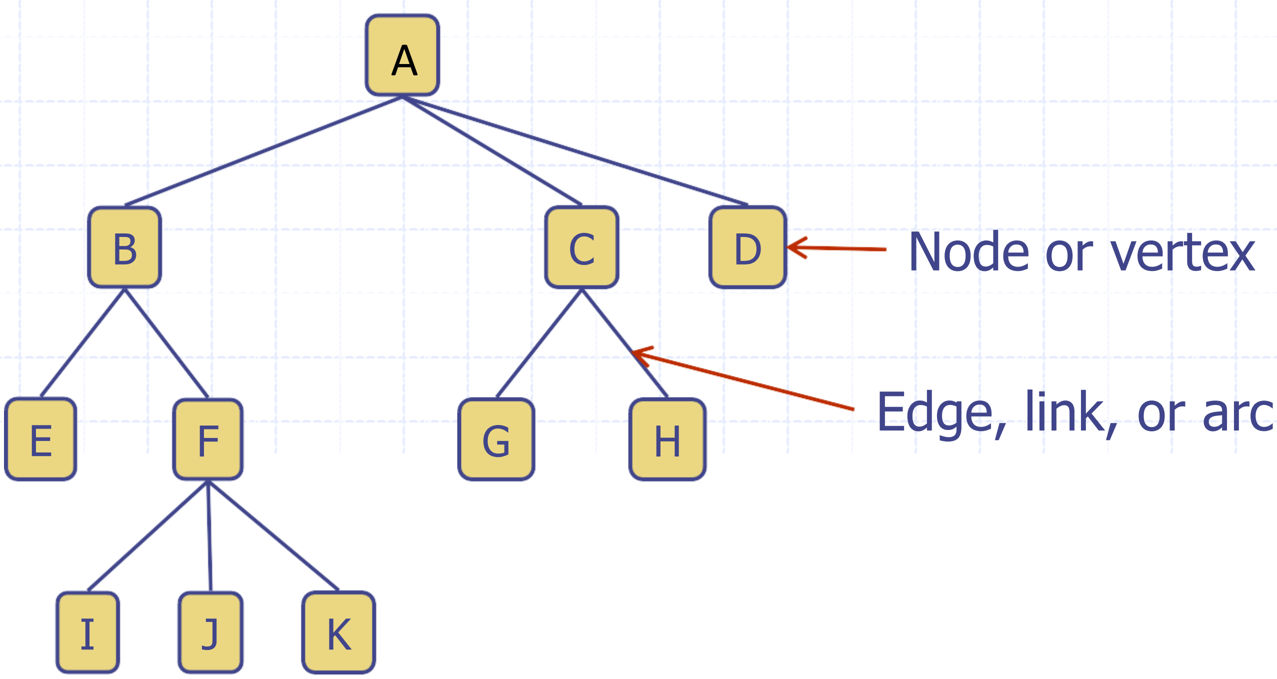
Applications

File system



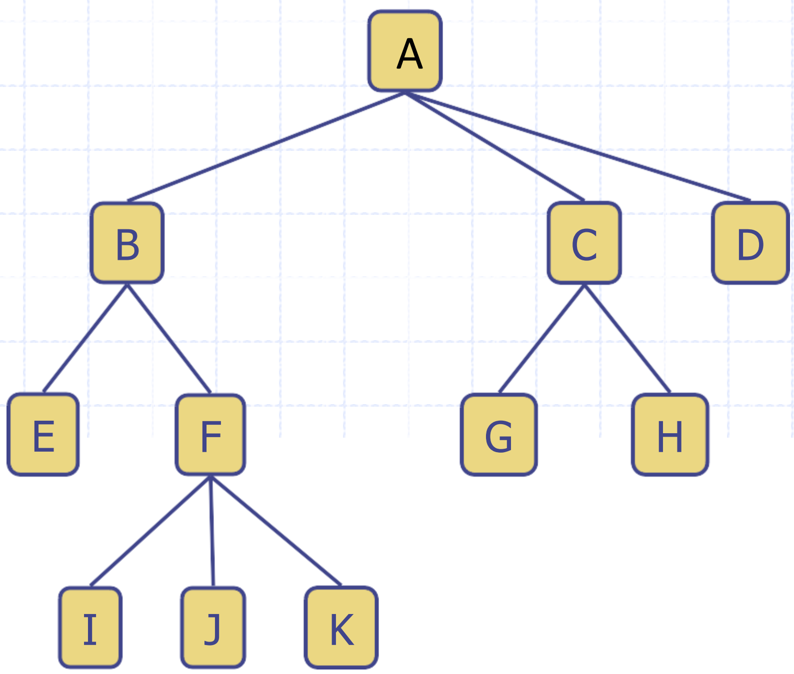
Tree Terminology

A is the root node
B is the parent of E and F
C is the sibling of B and D
B, C, and D are children of A



Tree Terminology

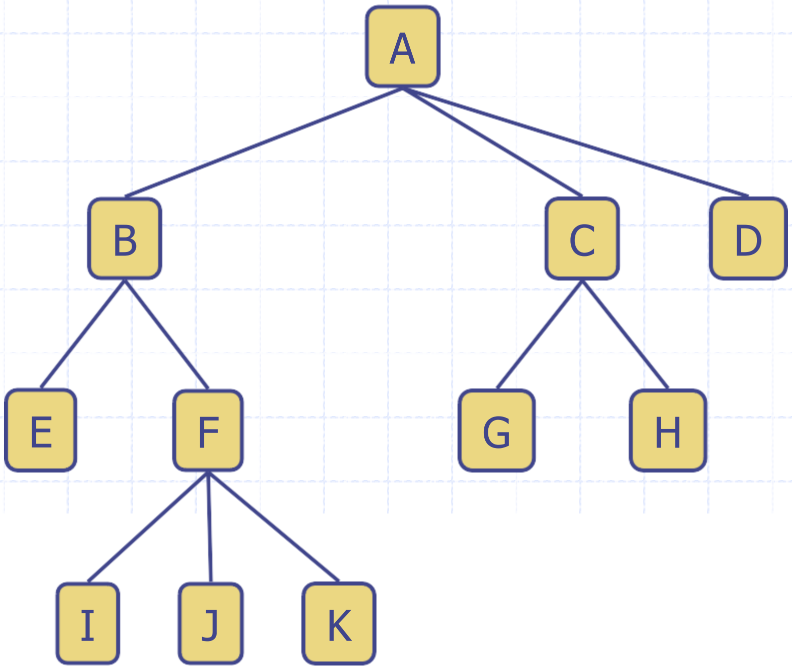
E, F, I, J, K are descendants of B
All nodes, except A, are descendants of A
A, B, F are ancestors of J
A has no ancestors



Tree Terminology

Internal node:
node with at least one child (A, B, C, F)

External node or leaf:
node without children (E, I, J, K, G, H, D)



Tree Terminology

Depth or level of a node:

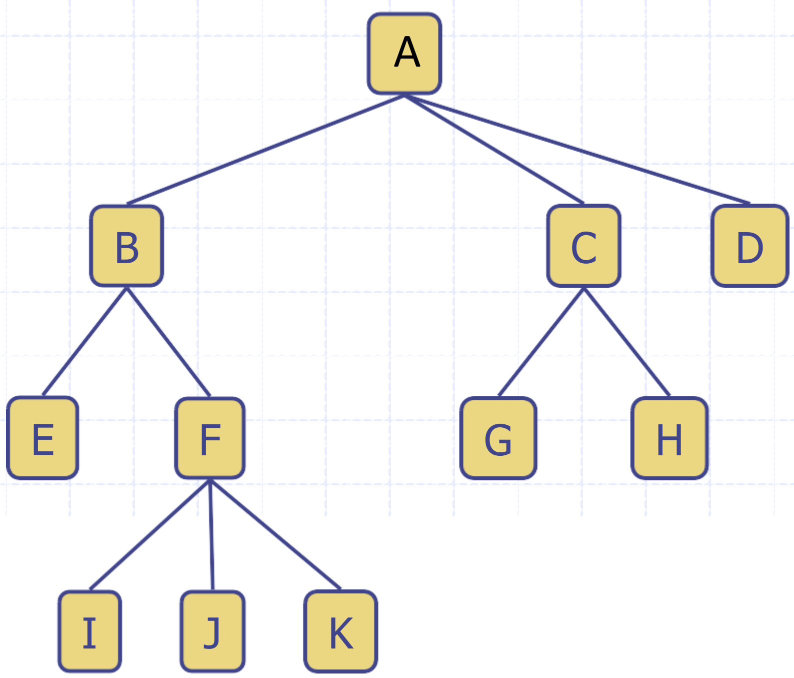
number of ancestors. Depth of E is 2.

Height of tree

maximum depth of any node. Tree has height 3.

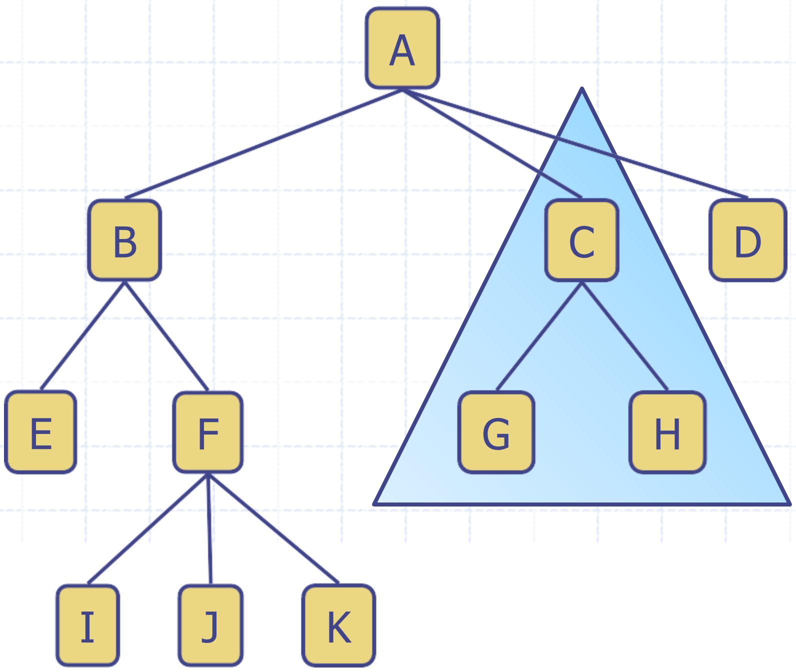
Degree of a node:

number of children. Degree of F is 3.



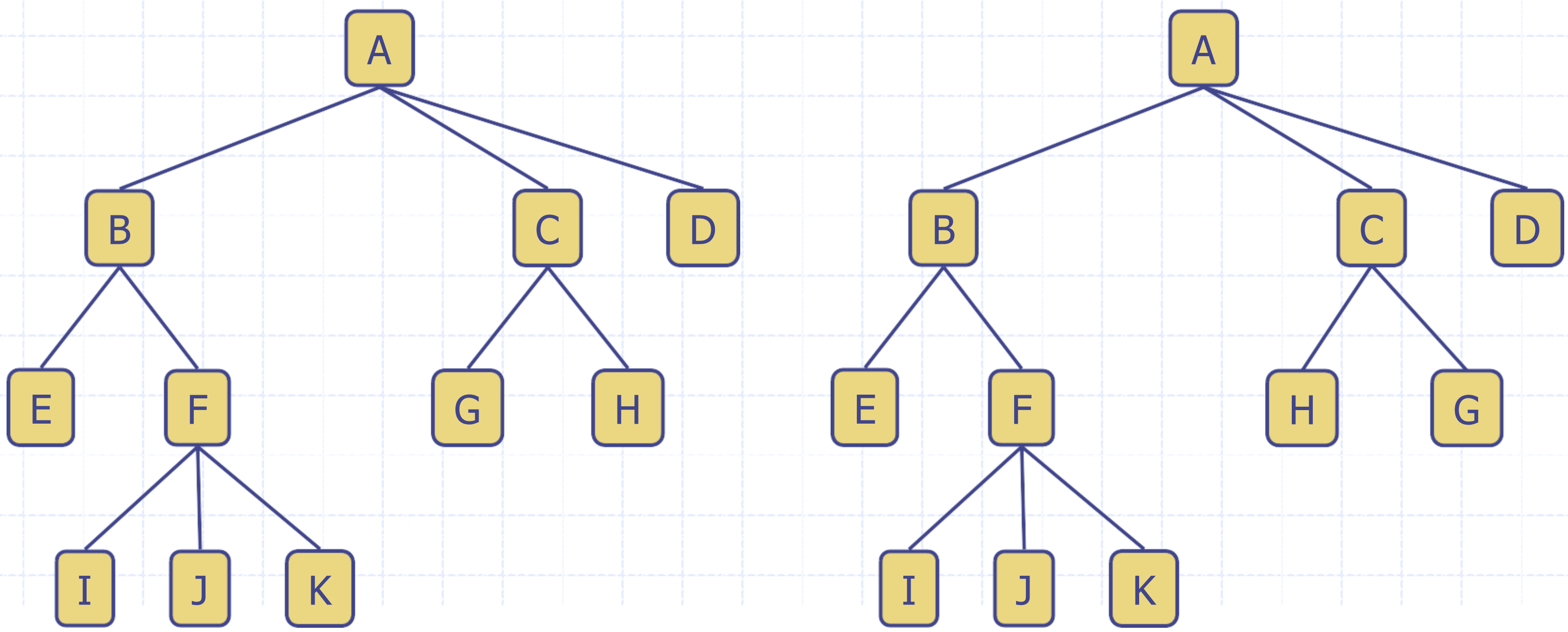
Tree Terminology

Subtree: tree consisting of a node and its descendants



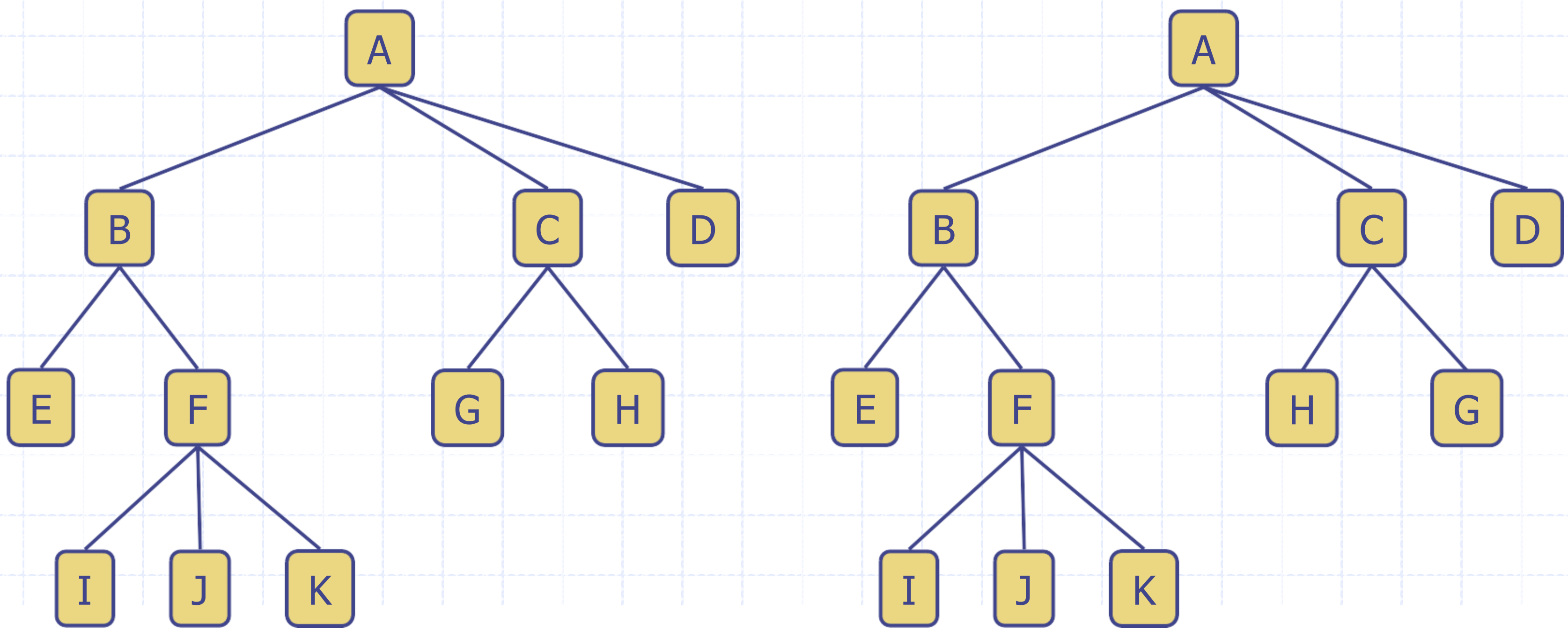
Tree Terminology

Ordered tree: The children of a node are ordered.



Tree Terminology

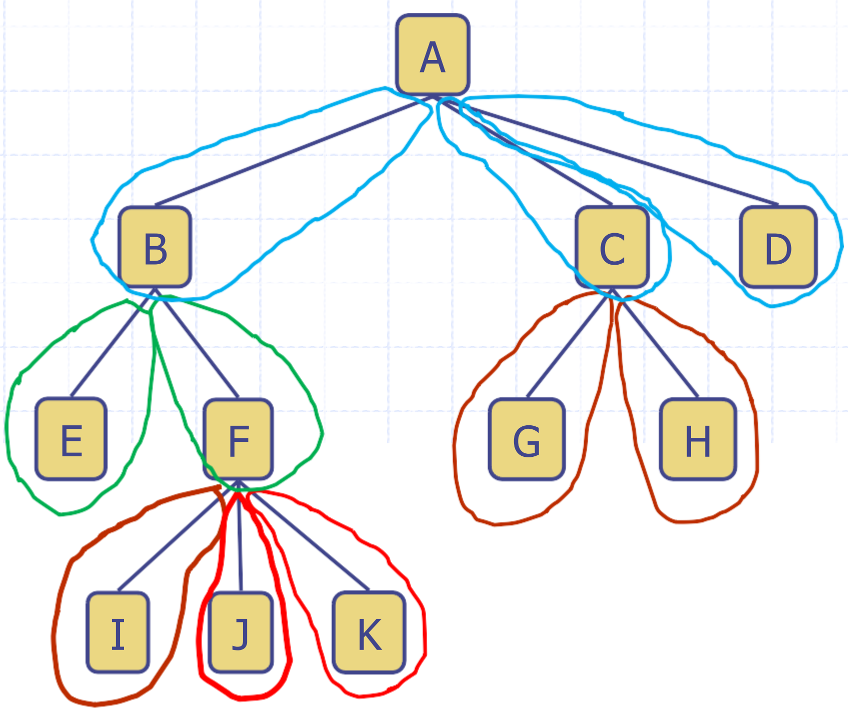
Un-Ordered tree: The children of a node are not ordered.



Tree Properties

Number of edges = Number of nodes -1

Proof. Glue every node to the edge connecting it to its parent. The root is not glued to any edges.



Tree ADT

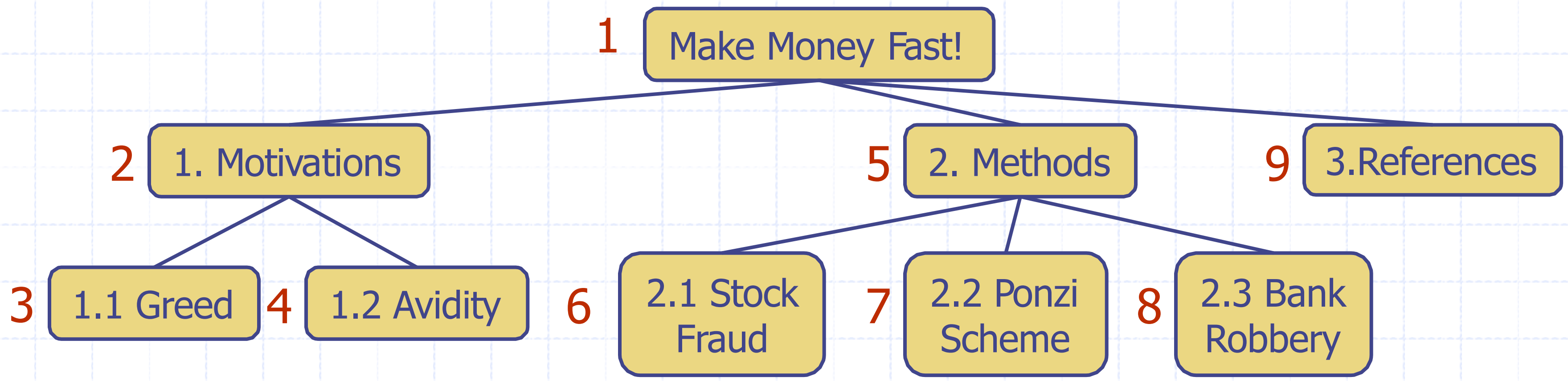
- Generic methods:
  - integer size()
  - boolean isEmpty()
  - Iterator iterator()
- Accessor methods:
  - position root()
  - position parent(p)
  - Iterable children(p)
  - Integer numChildren(p)
- Query methods:
  - boolean isInternal(p)
  - boolean isExternal(p)
  - boolean isRoot(p)
- Additional update methods may be defined by data structures implementing the Tree ADT

Preorder Traversal

- A tree traversal visits the nodes of a tree in a systematic manner
- In a preorder traversal, a node is visited before its descendants

Algorithm ***preOrder***(***v***)
visit(***v***)
for each child ***w*** of ***v*** **do**
    preOrder (***w***)

Preorder Traversal



Make Money Fast!

1. Motivations
    1. Greed
    2. Avidity
2. Methods
    1. Stock Fraud
    2. Ponzi Scheme
    3. Bank Robbery
3. References

Postorder Traversal

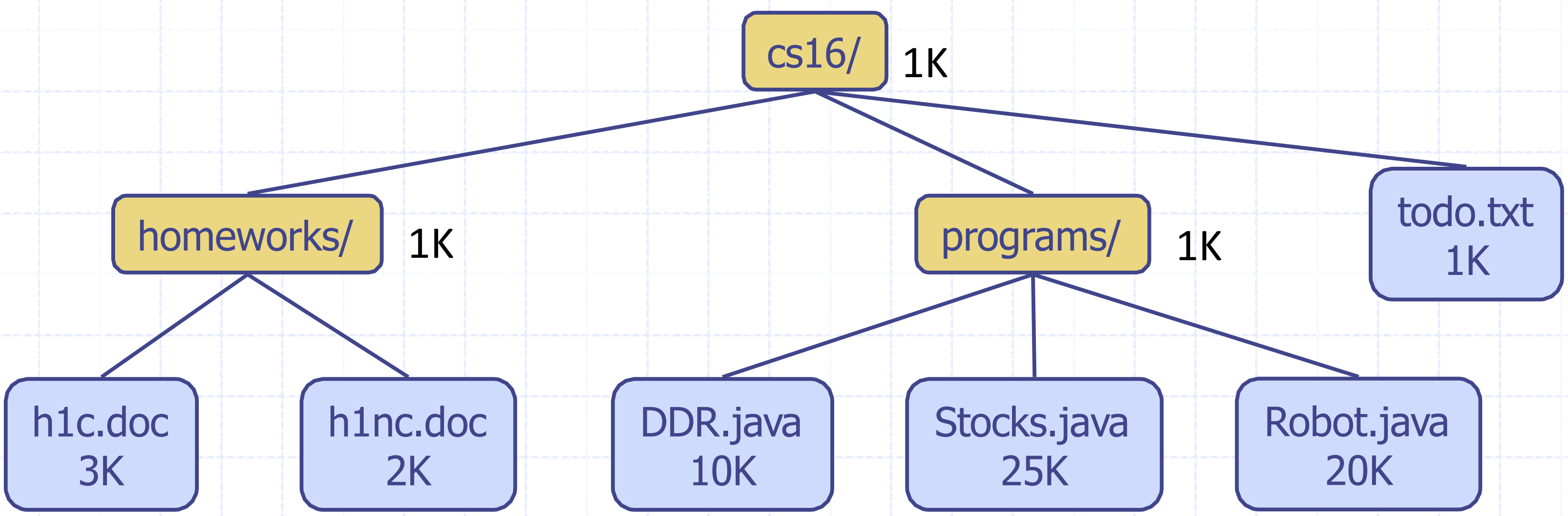
In a postorder traversal, a node is visited after its descendants

Algorithm ***postOrder***(v)
    for each child w of v **do**
            postOrder (w)
    visit(v)

Postorder Traversal

Application

Compute space used by the files in a directory and its subdirectories

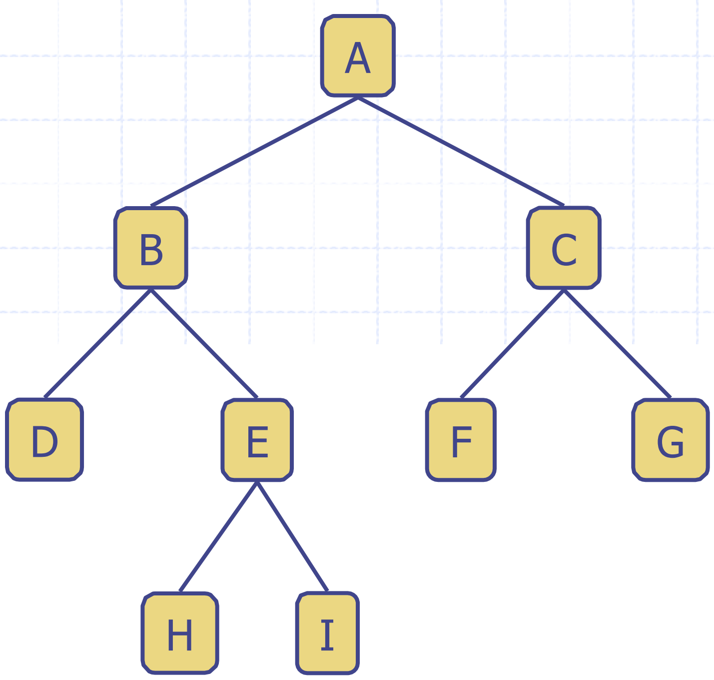


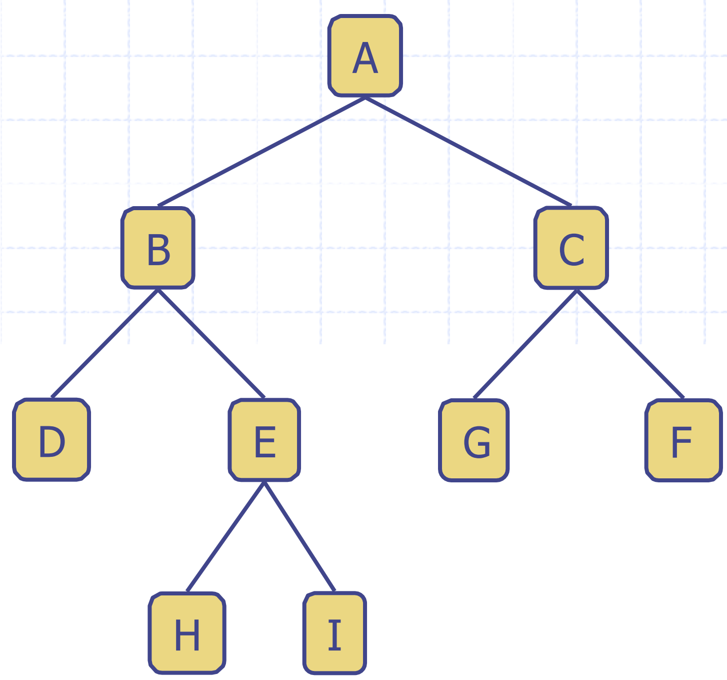
Binary Trees

A binary tree is a tree with the following properties:

- Internal nodes have ≤ 2 children (exactly two for proper binary trees)
- The children of a node are an ordered pair

We call the children of an internal node left child and right child

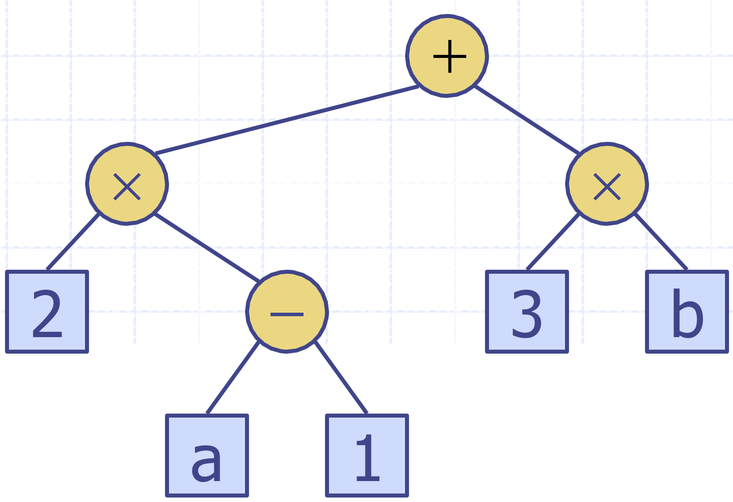




Arithmetic Expression Tree

Binary tree associated with an arithmetic expression
- internal nodes: operators
- external nodes: operands

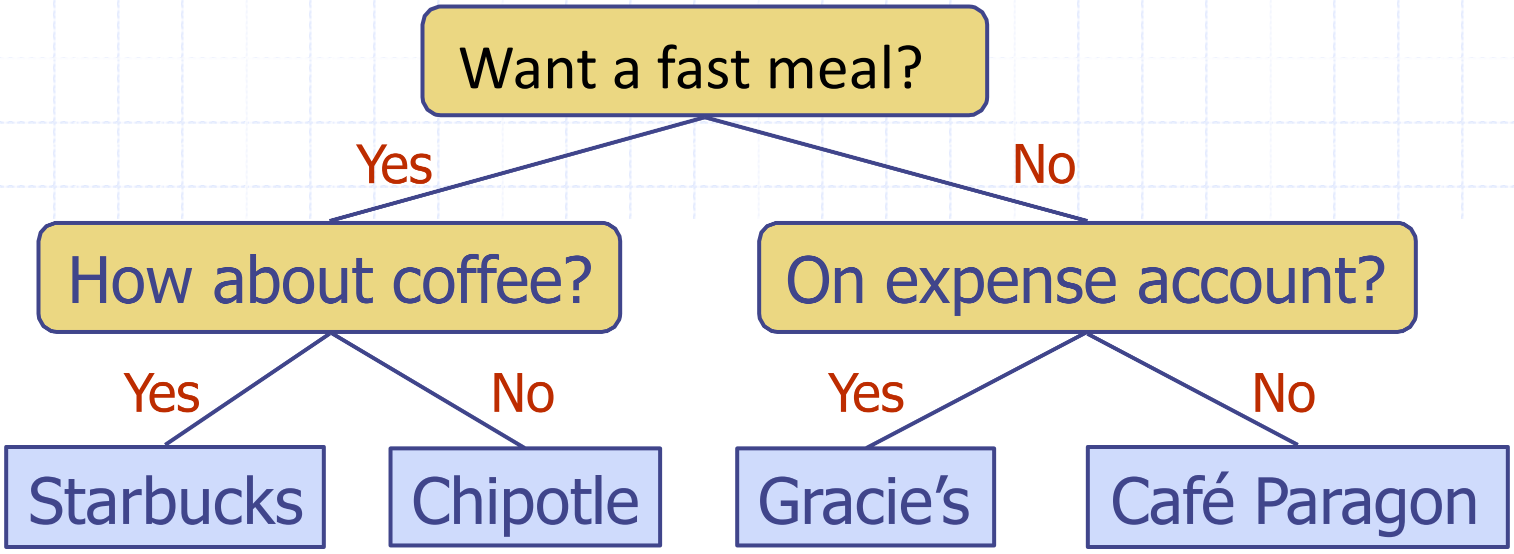
Example: arithmetic expression tree for the expression $(2 \times (a - 1) + (3 \times b))$



Decision Tree

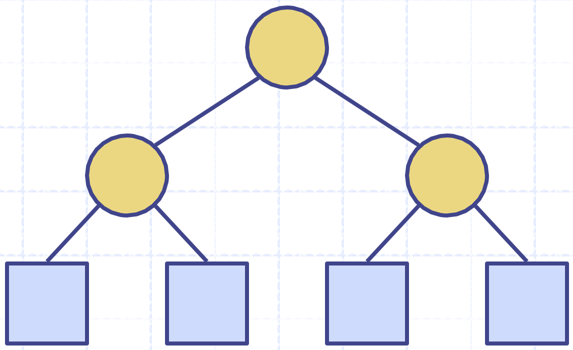
Binary tree associated with a decision process
- internal nodes: questions with yes/no answer
- external nodes: decisions

Example: dining decision



Properties of Proper Binary Trees

 # leaves = #internal nodes + 1



Proof. Let n = number of nodes

#edges = n-1, also

#edges = 2 (#internal nodes), as each internal node has 2 edges incident on it

So, n-1 = 2 (#internal nodes)

#internal nodes = (n-1)/2

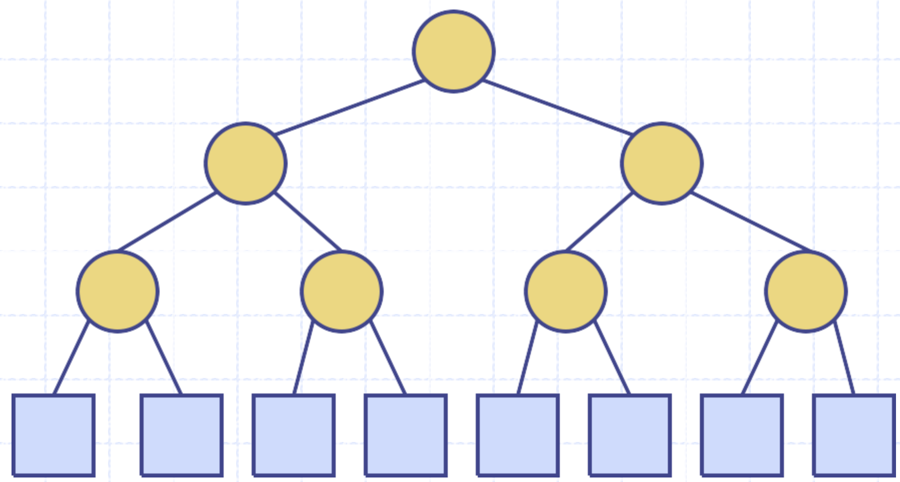
since n = #leaves + #internal nodes

then #leaves = (n+1)/2

Properties of Proper Binary Trees

- # nodes at level $i \leq 2^i$ a
- # leaves $\leq 2^{\text{height}}$
- $\log_2$ (# leaves) $\leq$ height $\leq$ #internal nodes

Level	#Nodes
0	2^0
1	2^1
2	2^2
3	2^3



Maximum level = height of the tree

BinaryTree ADT

- The BinaryTree ADT extends the Tree ADT, i.e., it inherits all the methods of the Tree ADT
- Additional methods:
  - position left(p)
  - position right(p)
  - position sibling(p)
- The above methods return null when there is no left, right, or sibling of p, respectively
- Update methods may be defined by data structures implementing the BinaryTree ADT

Inorder Traversal

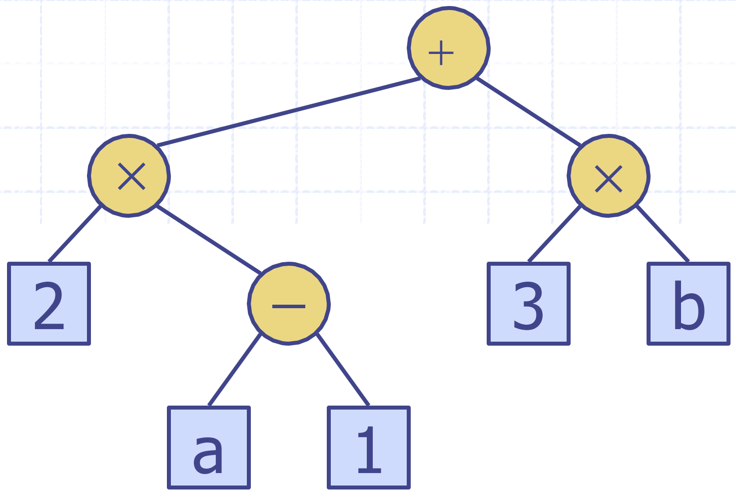
In an inorder traversal a node is visited after its left subtree and before its right subtree

Algorithm ***inOrder*****(*****v*****)**
    if ***left*** **(*****v*****)** $\neq$ **null then**
        inOrder **(*****left*** **(*****v*****))**
    visit**(*****v*****)**
    if ***right*****(*****v*****)** $\neq$ **null then**
        inOrder **(*****right*** **(*****v*****))**

Print Arithmetic Expressions

- Specialization of an inorder traversal
  - print operand or operator when visiting node
  - print "(" before traversing left subtree
  - print ")" after traversing right subtree

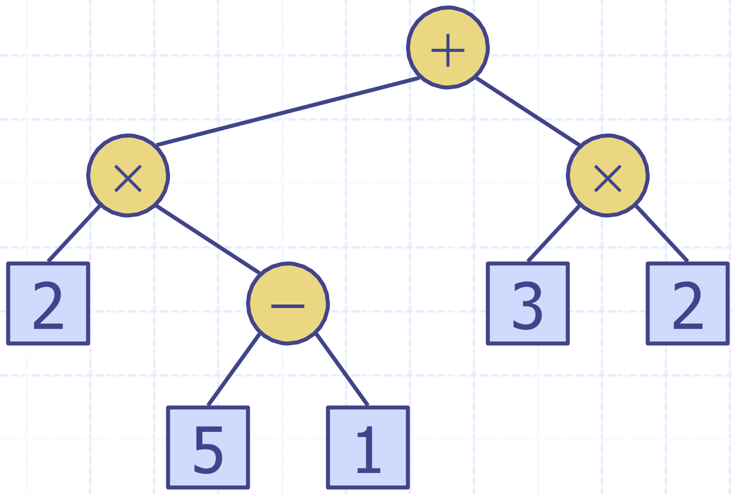
Algorithm ***printExpression***(*v*)
  if *v* is a leaf **then**
    print(v.element())
  else {
    print("(' ')
    printExpression (*left*(*v*))
    print(*v.element* ())
    printExpression (*right*(*v*))
    print (")' ')
  }



$((2 \times (a - 1)) + (3 \times b))$

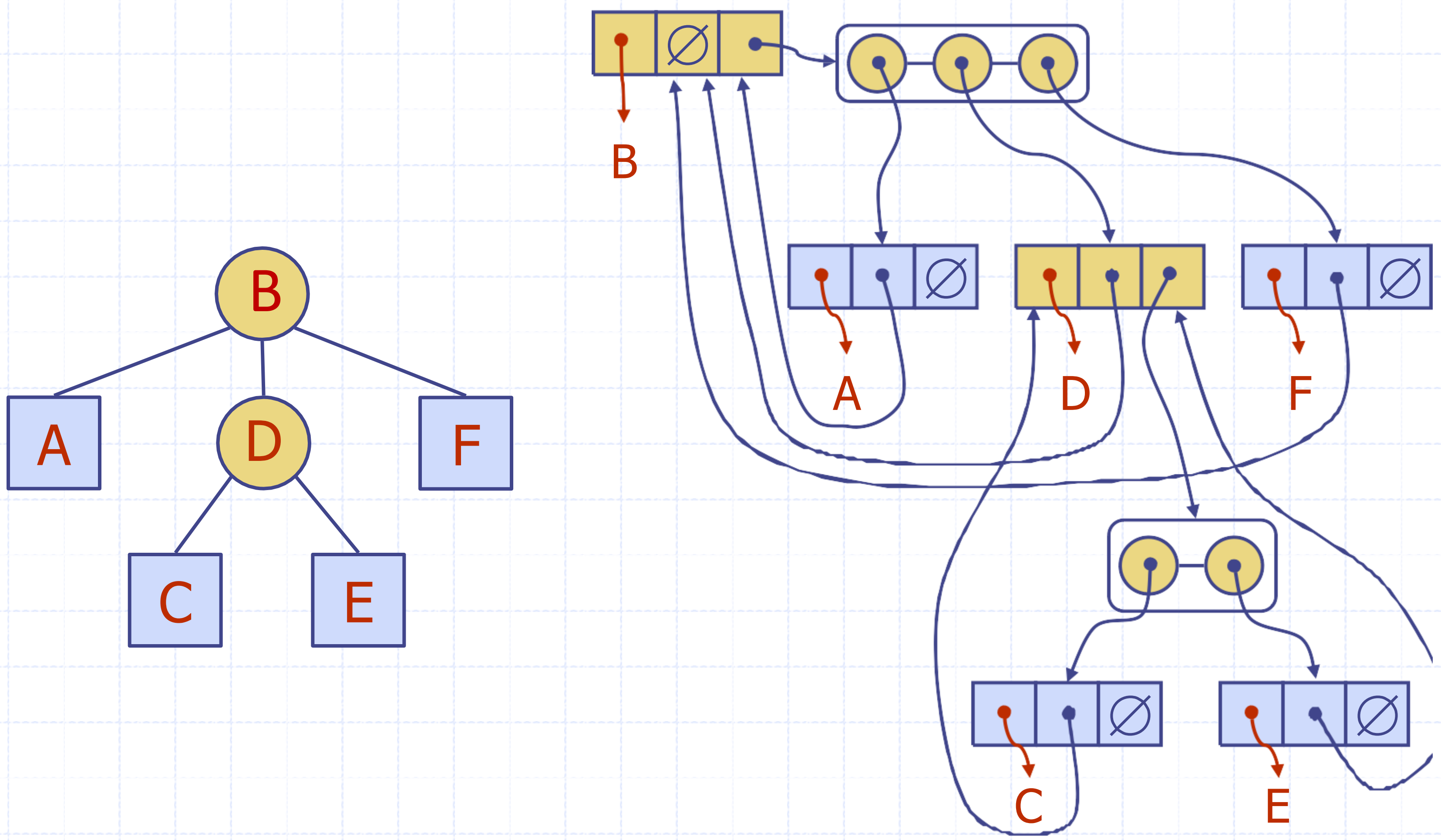
Evaluate Arithmetic Expressions

- Specialization of a postorder traversal
  - recursive method returning the value of a subtree
  - when visiting an internal node, combine the values of the subtrees



Algorithm *evalExpr*(*v*)
  if *isExternal* (*v*) **then**
    return *v.element* ()
  else {
    x ← *evalExpr*(*left*(*v*))
    y ← *evalExpr*(*right*(*v*))
    ◊ ← operator stored at *v*
    return *x* ◊ *y*
  }

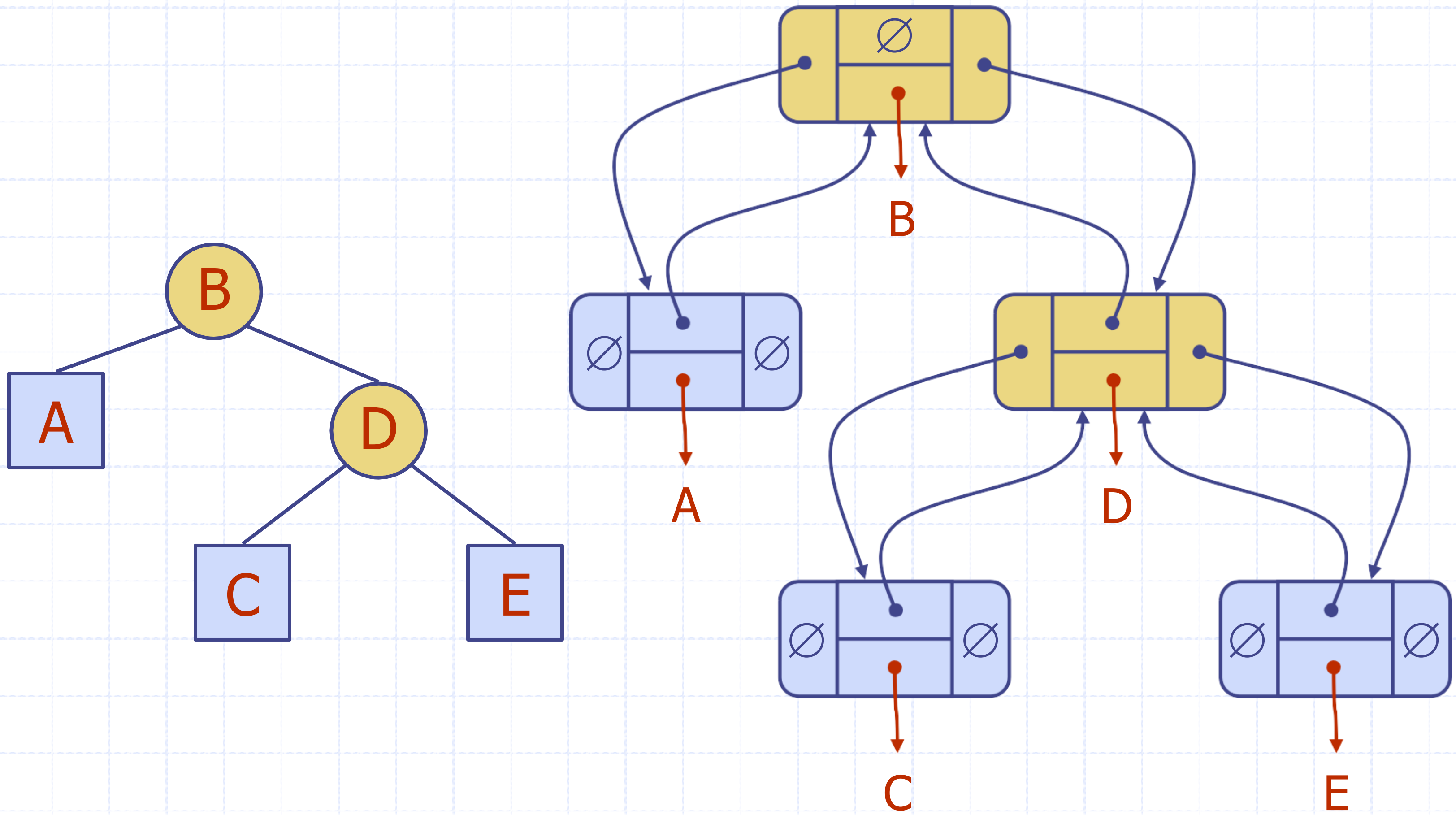
Linked Structure for Trees



A node is represented by a node storing

- Element or data
- Parent
- List of children

Linked Structure for Binary Trees



A node is represented by an object storing

- Element or data
- Parent
- Left child
- Right child