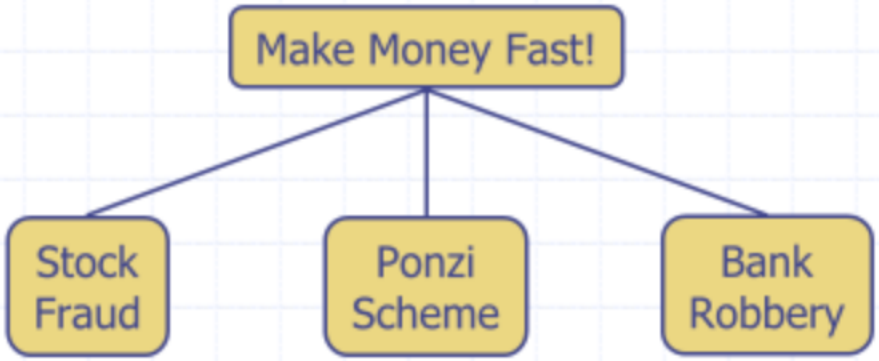
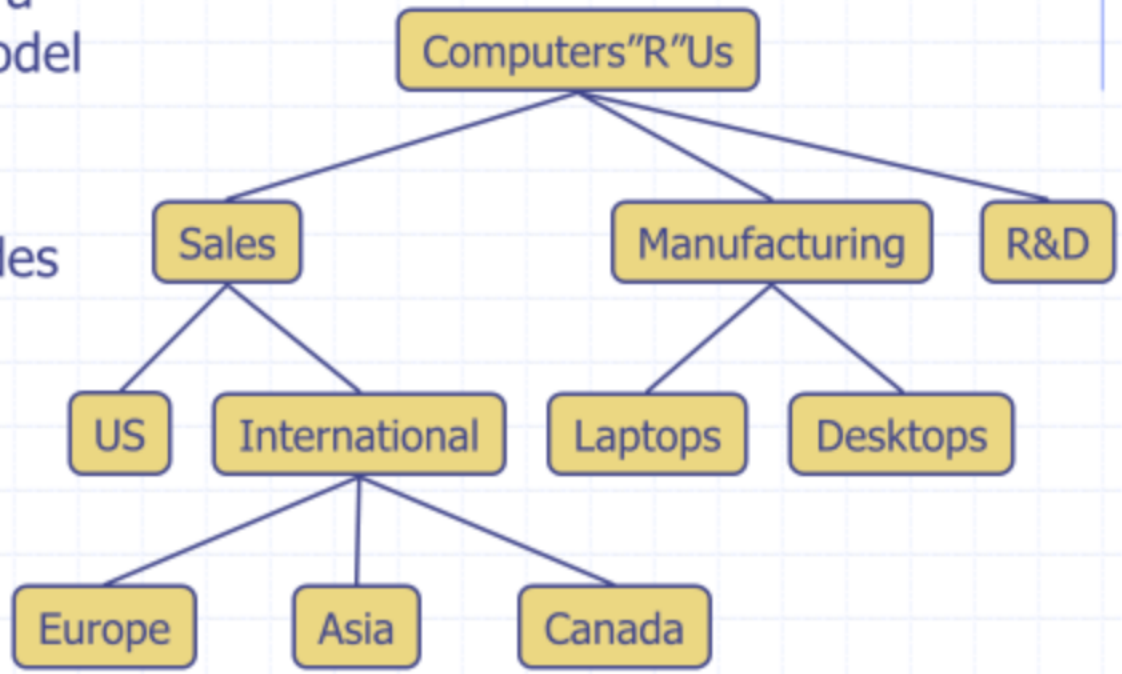
Trees



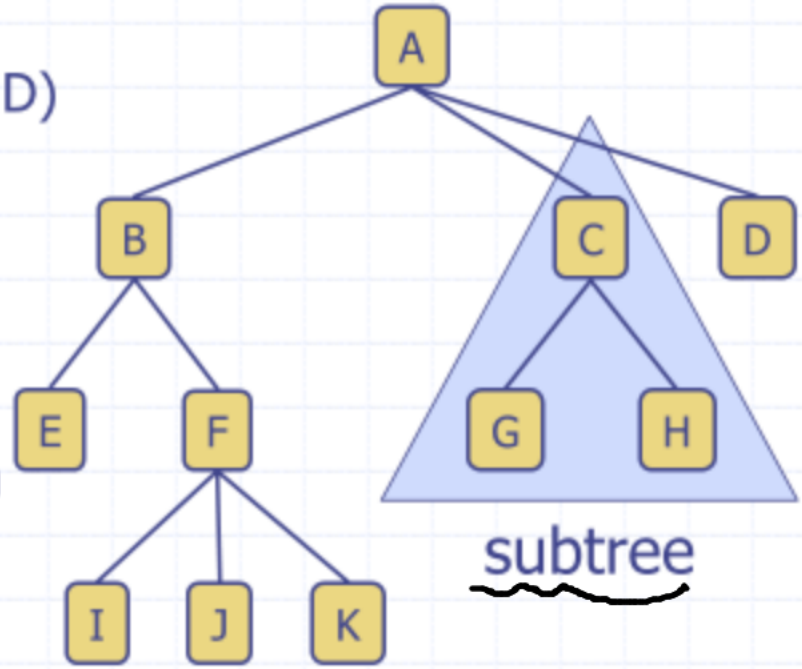
What is a Tree

- In computer science, a tree is an abstract model of a hierarchical structure
- A tree consists of nodes with a parent-child relation
- Applications:
  - Organization charts
  - File systems
  - Programming environments



Tree Terminology

- Root: node without parent (A)
- Internal node: node with at least one child (A, B, C, F)
- External node (a.k.a. leaf): node without children (E, I, J, K, G, H, D)
- Ancestors of a node: parent, grandparent, grand-grandparent, etc.
- Depth of a node: number of ancestors
- Height of a tree: maximum depth of any node (3)
- Descendant of a node: child, grandchild, grand-grandchild, etc.
- Subtree: tree consisting of a node and its descendants



Tree ADT (§ 6.1.2)

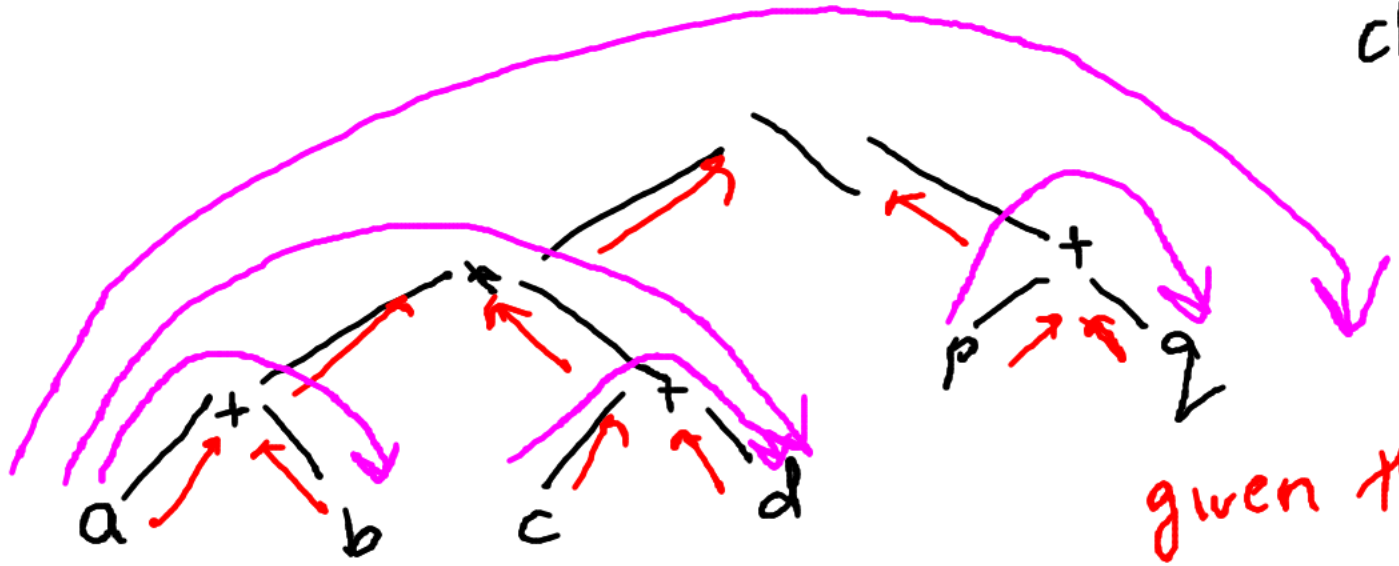
- We use positions to abstract nodes
- Generic methods:
  - integer size()
  - boolean isEmpty()
  - Iterator elements()
  - Iterator positions()
- Accessor methods:
  - position root()
  - position parent(p)
  - positionIterator children(p)
- Query methods:
  - boolean isInternal(p)
  - boolean isExternal(p)
  - boolean isRoot(p)
- Update method:
  - object replace (p, o)
- Additional update methods may be defined by data structures implementing the Tree ADT

Q: what would be the axioms for the tree ADT?

(c) Preorder on this tree gives (with visit = print)

\ (* (+ (a b)) (+ (c d))) (+ (p q)) : Process node 4 then its children

Draw binary trees to represent (i) the arithmatic expression: ((a+ b)*(c+ d))/(p+ q)



(a) Evaluation of this expression given the tree requires processing of children before processing of the node : Post order traversal

(b) Printing of this expression given the tree

Recurse : (left subtree) current node (right subtree)

Inorder traversal

Preorder Traversal

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

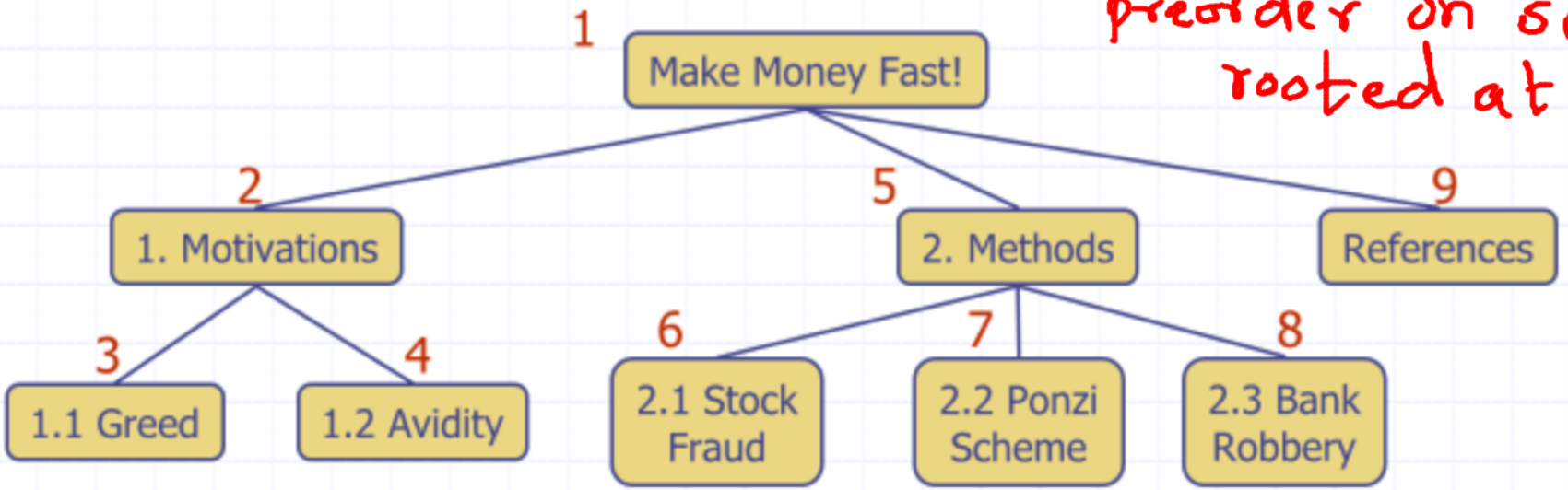
Algorithm *preOrder(v)*
visit(v)
for each child *w* of *v*
preorder (w)

eg: print or evaluate

visit ⇒ Process

visit ≠ access

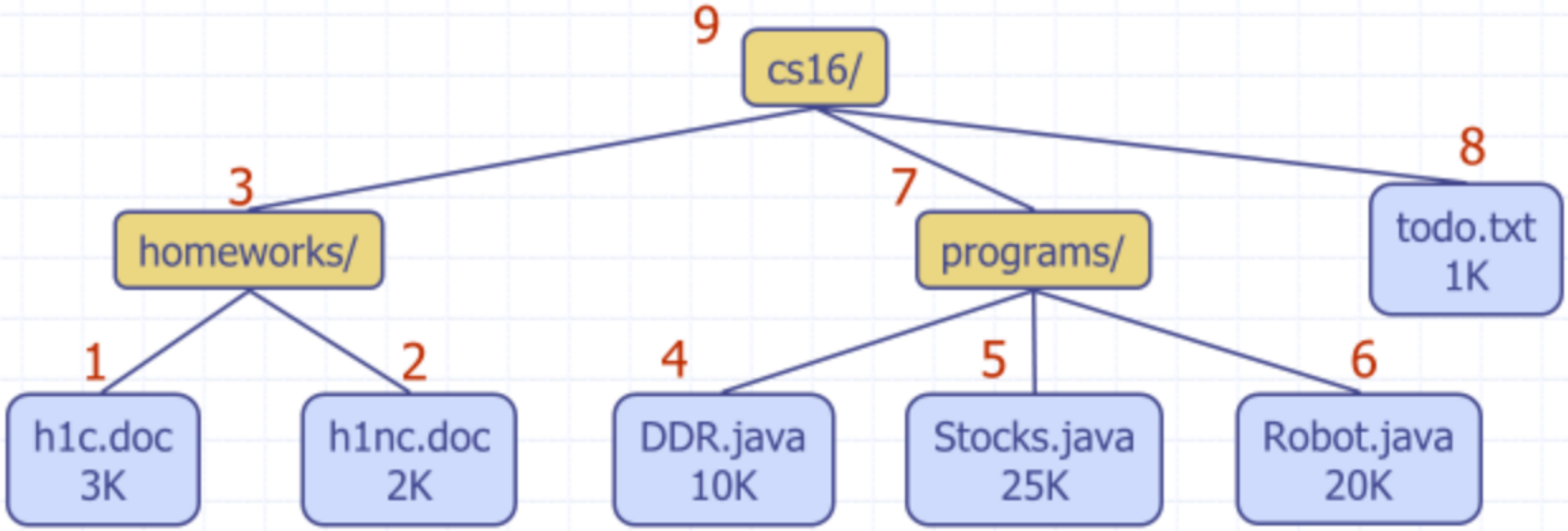
preorder on subtree rooted at w



Postorder Traversal

- In a postorder traversal, a node is visited after its descendants
- Application: compute space used by files in a directory and its subdirectories

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



Eg: "du" utility in linux computes disk space utilised by a dir by evaluating children befor the node

```
ganesh@ganesh:~/Installation$ du -h dhvani-tts-master/
116Kdhvani-tts-master/database/v
2.6Mdhvani-tts-master/database/cv
132Kdhvani-tts-master/database/c
496Kdhvani-tts-master/database/halfs
2.7Mdhvani-tts-master/database/vc
6.1Mdhvani-tts-master/database
324Kdhvani-tts-master/m4
44Kdhvani-tts-master/tdemos
12Kdhvani-tts-master/python
36Kdhvani-tts-master/man
28Kdhvani-tts-master/doc
900Kdhvani-tts-master/autom4te.cache
24Kdhvani-tts-master/src/soundtouch4c/.libs
12Kdhvani-tts-master/src/soundtouch4c/src
8.0Kdhvani-tts-master/src/soundtouch4c/.deps
108Kdhvani-tts-master/src/soundtouch4c
820Kdhvani-tts-master/src/.libs
148Kdhvani-tts-master/src/.deps
2.0Mdhvani-tts-master/src
12Mdhvani-tts-master/
```

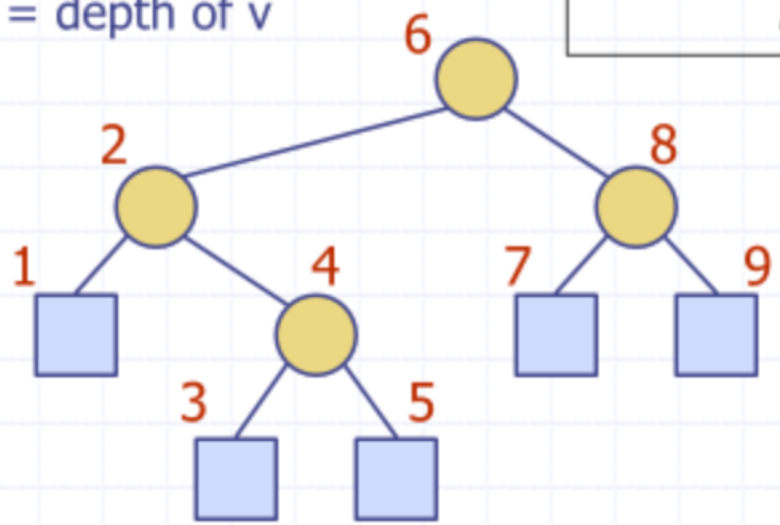
Try
du dir-name

Note: The underlined values dont add up exactly to 12 M since they have been rounded off for printing

Inorder Traversal

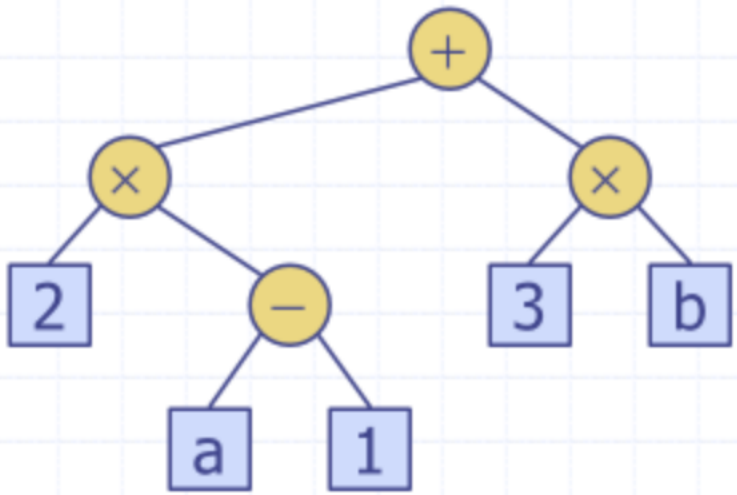
- In an inorder traversal a node is visited after its left subtree and before its right subtree
- Application: draw a binary tree
  - x(v) = inorder rank of v
  - y(v) = depth of v

Algorithm *inOrder(v)*
  if *hasLeft (v)*
    inOrder (left (v))
  visit(v)
  if *hasRight (v)*
    inOrder (right (v))



Arithmetic Expression Tree

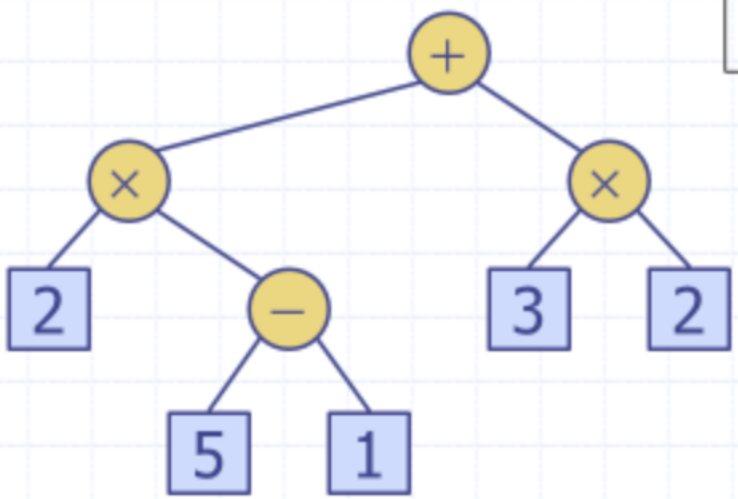
- Binary tree associated with an arithmetic expression
  - internal nodes: operators
  - external nodes: operands
- Example: arithmetic expression tree for the expression $(2 \times (a - 1) + (3 \times b))$



Evaluate Arithmetic Expressions

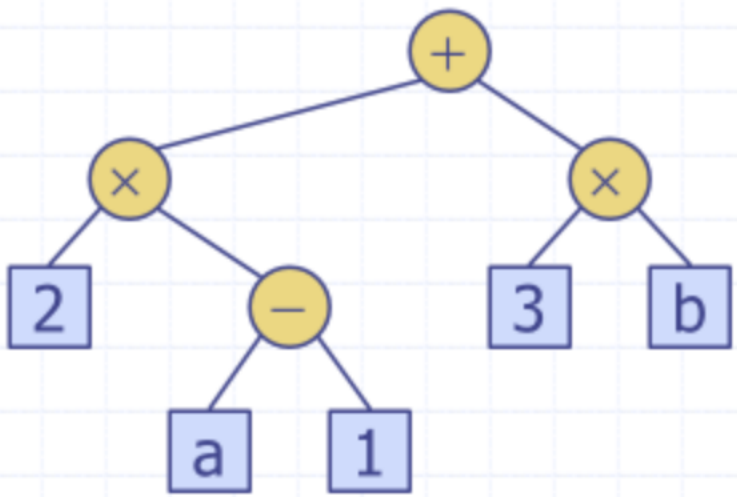
- Specialization of a postorder traversal
  - recursive method returning the value of a subtree
  - when visiting an internal node, combine the values of the subtrees

```
Algorithm evalExpr(v)
    if isExternal (v)
        return v.element ()
    else
        x ← evalExpr(leftChild (v))
        y ← evalExpr(rightChild (v))
        ◊ ← operator stored at v
        return x ◊ y
```



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

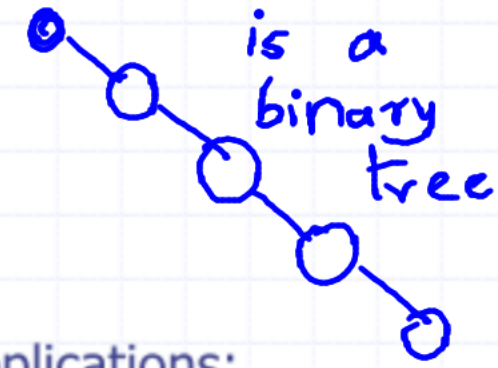


Algorithm *printExpression*(*v*)

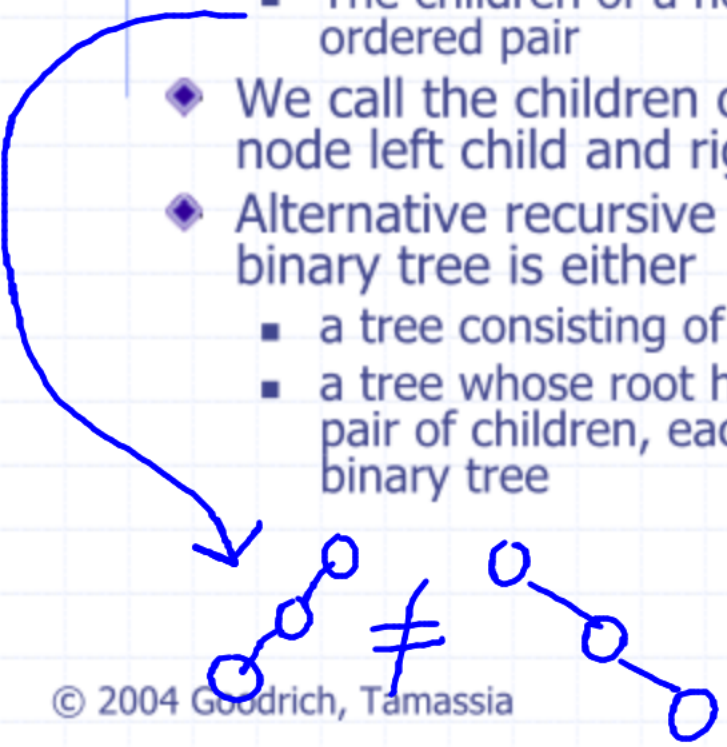
```
if hasLeft (v)
    print("(")
    inOrder (left(v))
print(v.element ())
if hasRight (v)
    inOrder (right(v))
    print (")")
```

((2 × (a − 1)) + (3 × b))

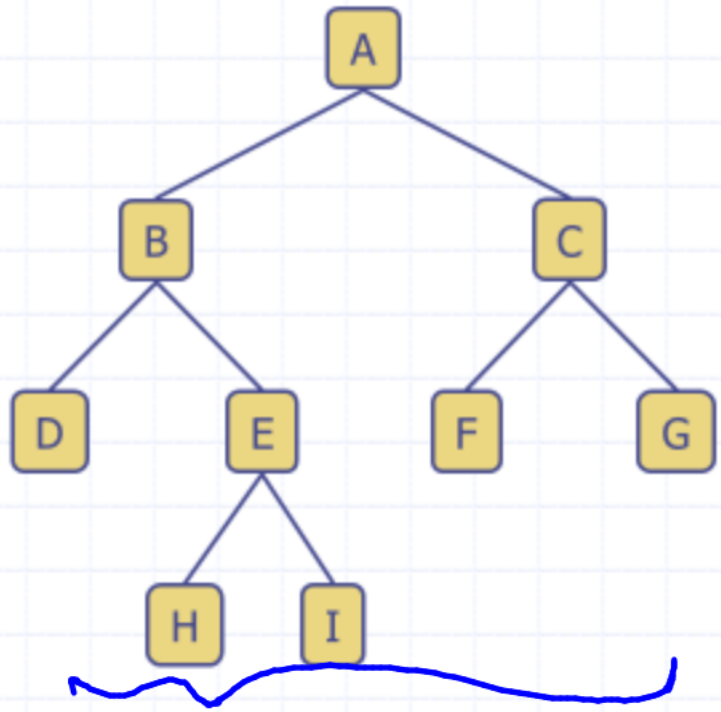
Binary Trees (§ 6.3)



- A binary tree is a tree with the following properties:
  - Each internal node has at most two 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
- Alternative recursive definition: a 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 binary tree



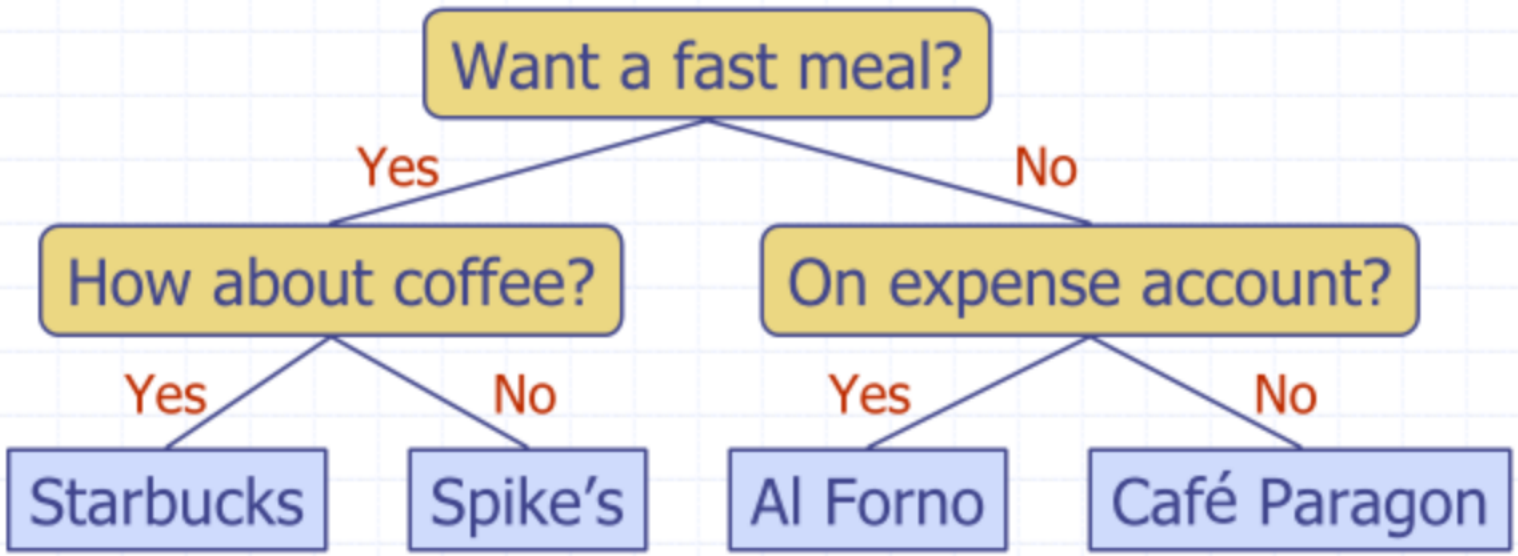
- Applications:
  - arithmetic expressions
  - decision processes
  - searching



A proper binary tree

Decision Tree

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



Properties of binary trees:

h = height

e = # of external nodes = # leaves

i = # of internal nodes

n = total # of nodes

(a) Show a relation between e & i
(Hint: You could use recursion)

(b) Show an upperbound & lowerbound for h in terms of n

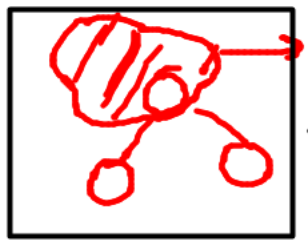
If your finding is only for "proper" binary trees, please state so.

Solutions / suggestions

(a) (i) $e = i + 1$ for proper tree : Proof by induction

Base: $i = 1$

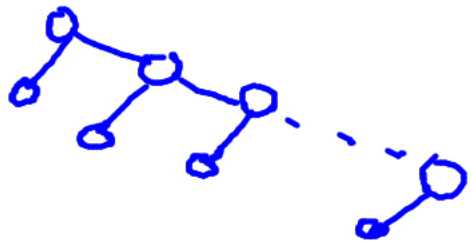
Assume for $i \le k$. Let $i = k+1$



has $i \& e$

→ has $i+1$ & $e+2-1$
$= e+1$

(b) (i) $h \le \left\lceil \frac{n}{2} \right\rceil$



for proper

(ii) $h \ge \log_2(n+1)$

(v) $h \ge \log_2(n+1) - 1$

(iii) $h \le \left\lceil \frac{n-1}{2} \right\rceil$

(iv) $h \ge \log_2 n + 1$