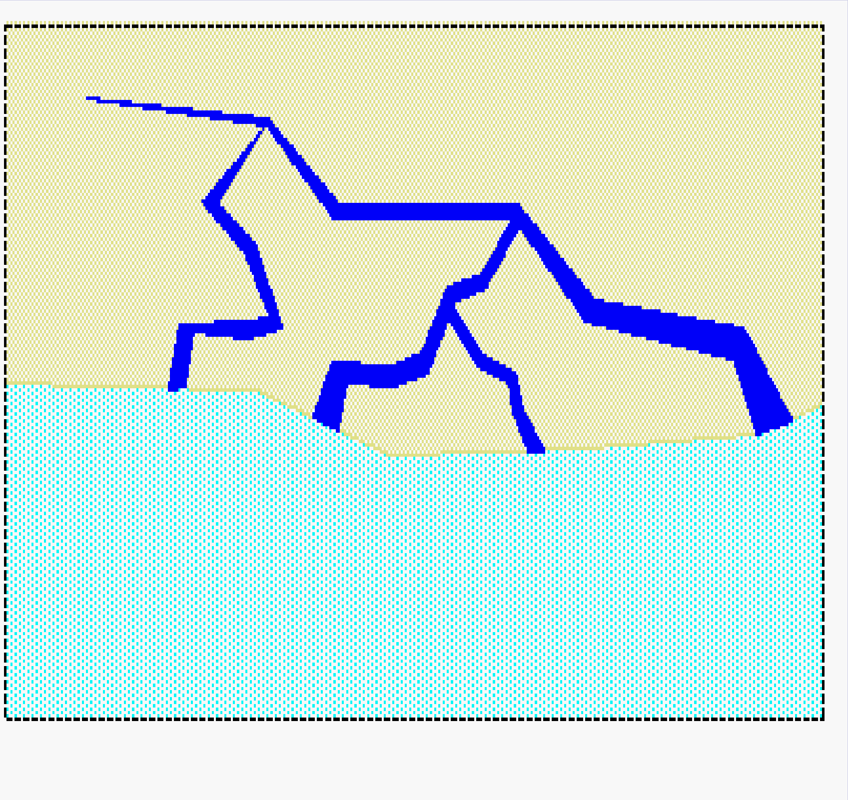
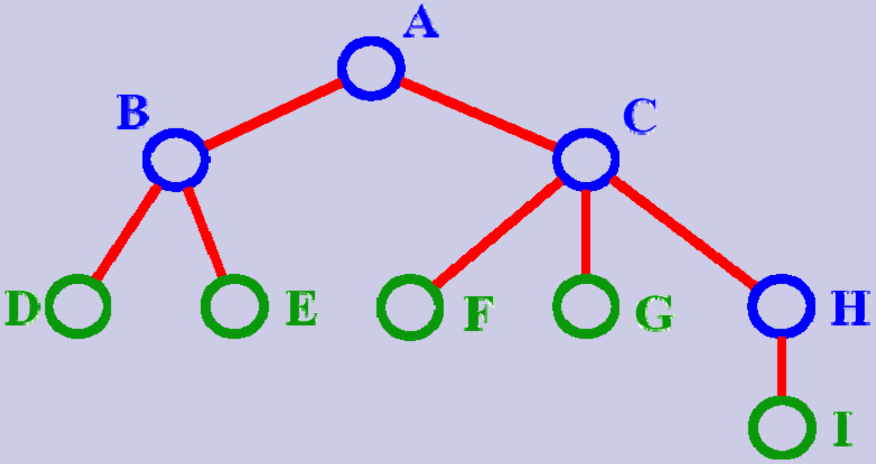
Trees

- trees
- binary trees
- data structures for trees



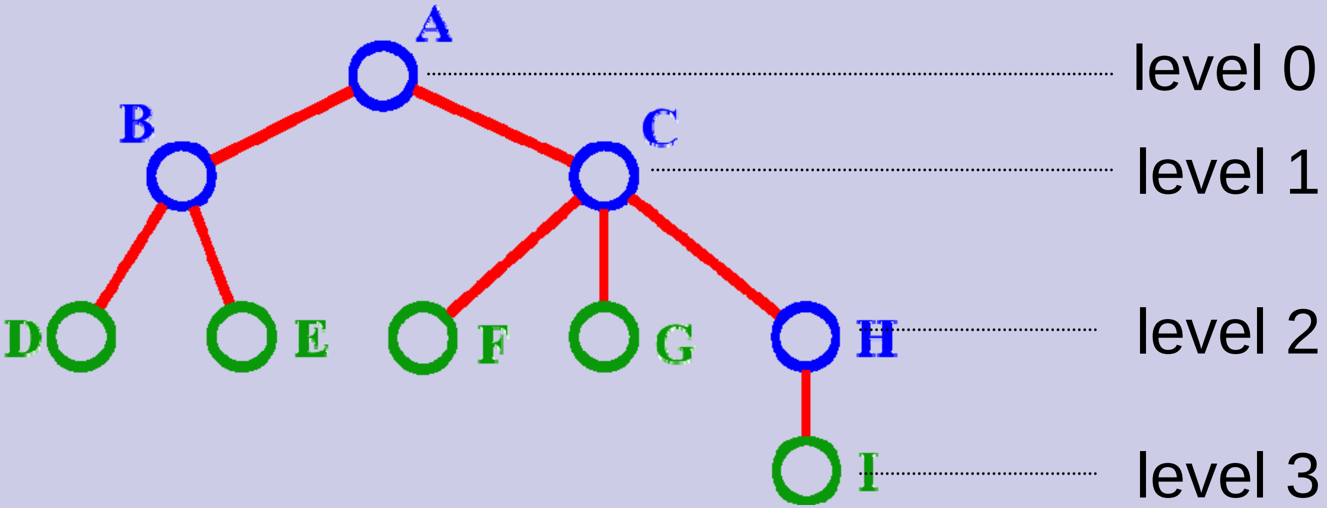
Trees: Definitions

- *A* is the *root* node.
- *B* is *parent* of D & E.
- *A* is *ancestor* of D & E.
- D and E are *descendants* of *A*.
- *C* is the *sibling* of B
- *D* and *E* are the *children* of B.
- *D, E, F, G, I* are *leaves*.

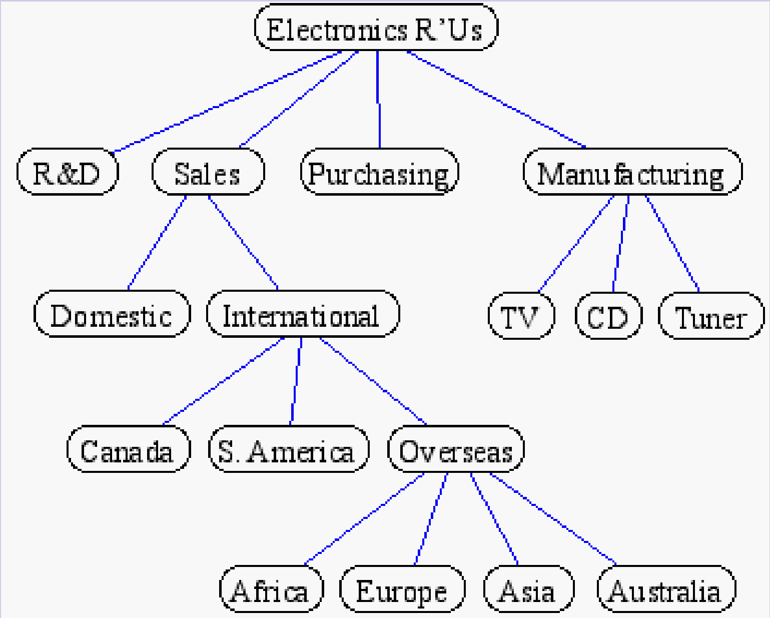


Trees: Definitions (2)

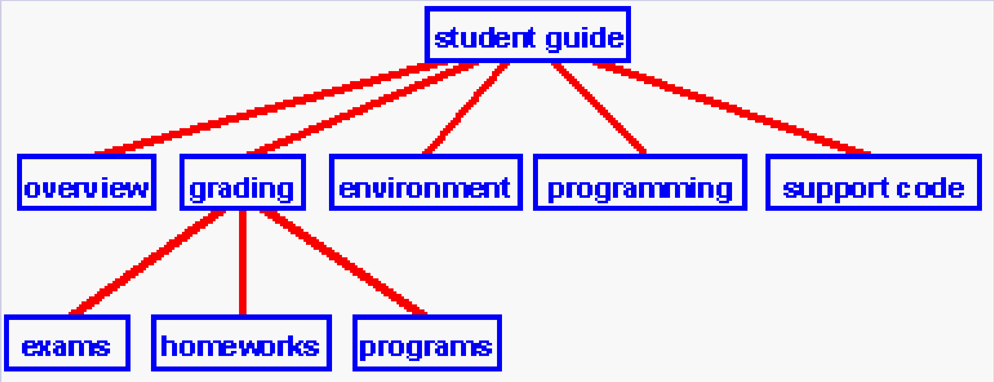
- *A, B, C, H* are *internal nodes*
- The *depth (level)* of *E* is *2*
- The *height* of the tree is *3*
- The *degree* of node *B* is *2*



Trees



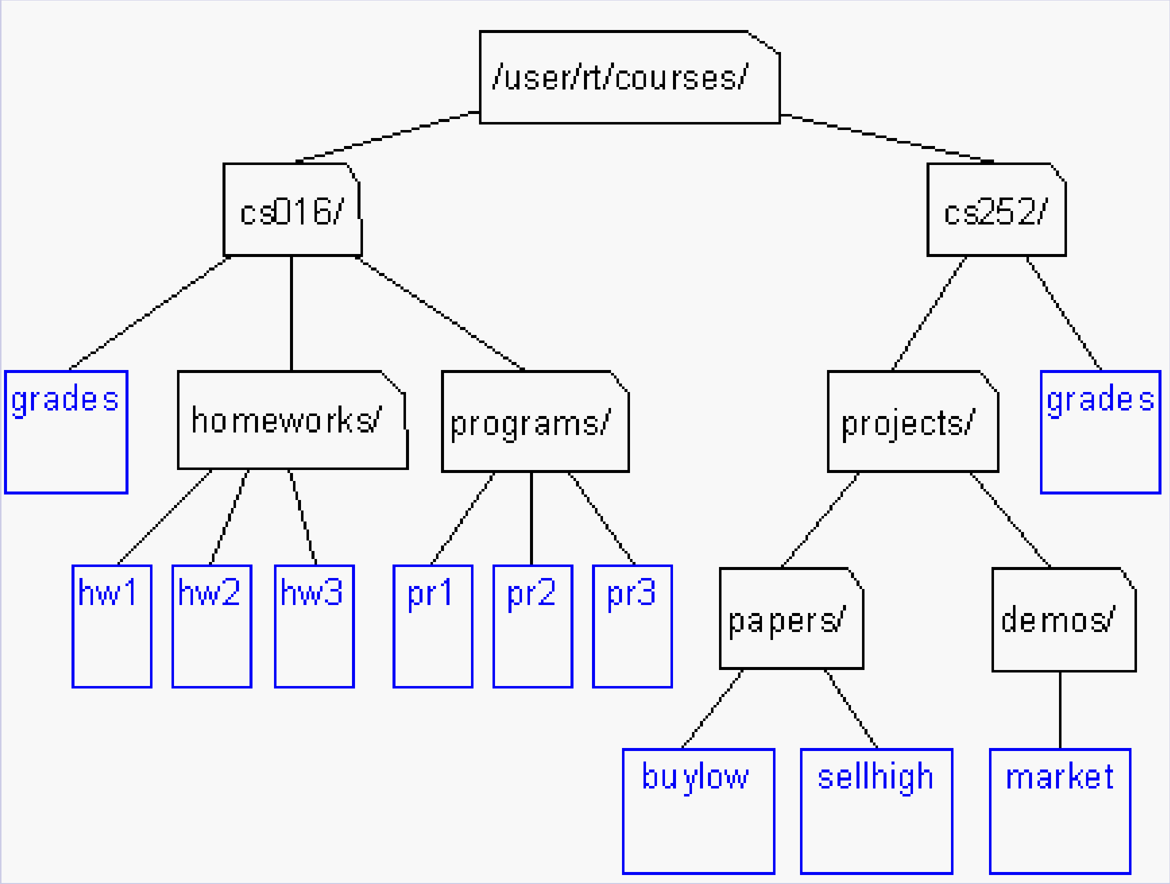
A tree represents a hierarchy, for e.g. the organization structure of a corporation



Or table of contents of a book

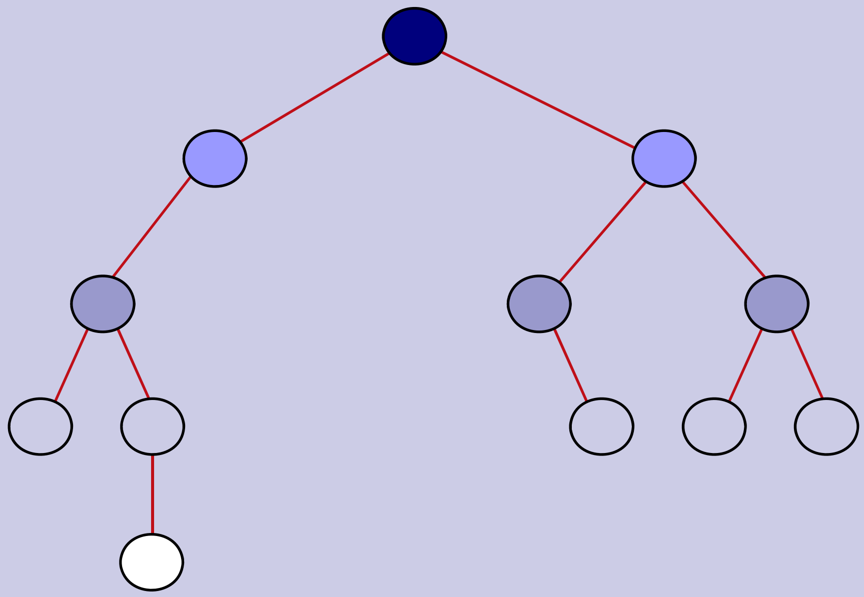
Another Example

Unix or DOS/Windows file system



Binary Tree

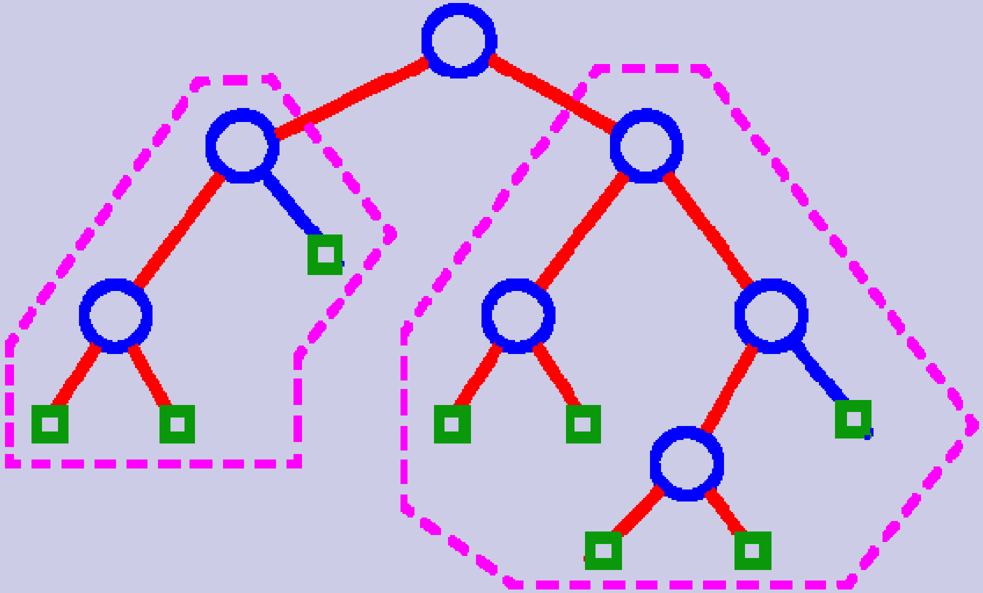
- An **ordered tree** is one in which the children of each node are ordered.
- **Binary tree:** ordered tree with all nodes having at most 2 children.



Recursive definition of binary tree

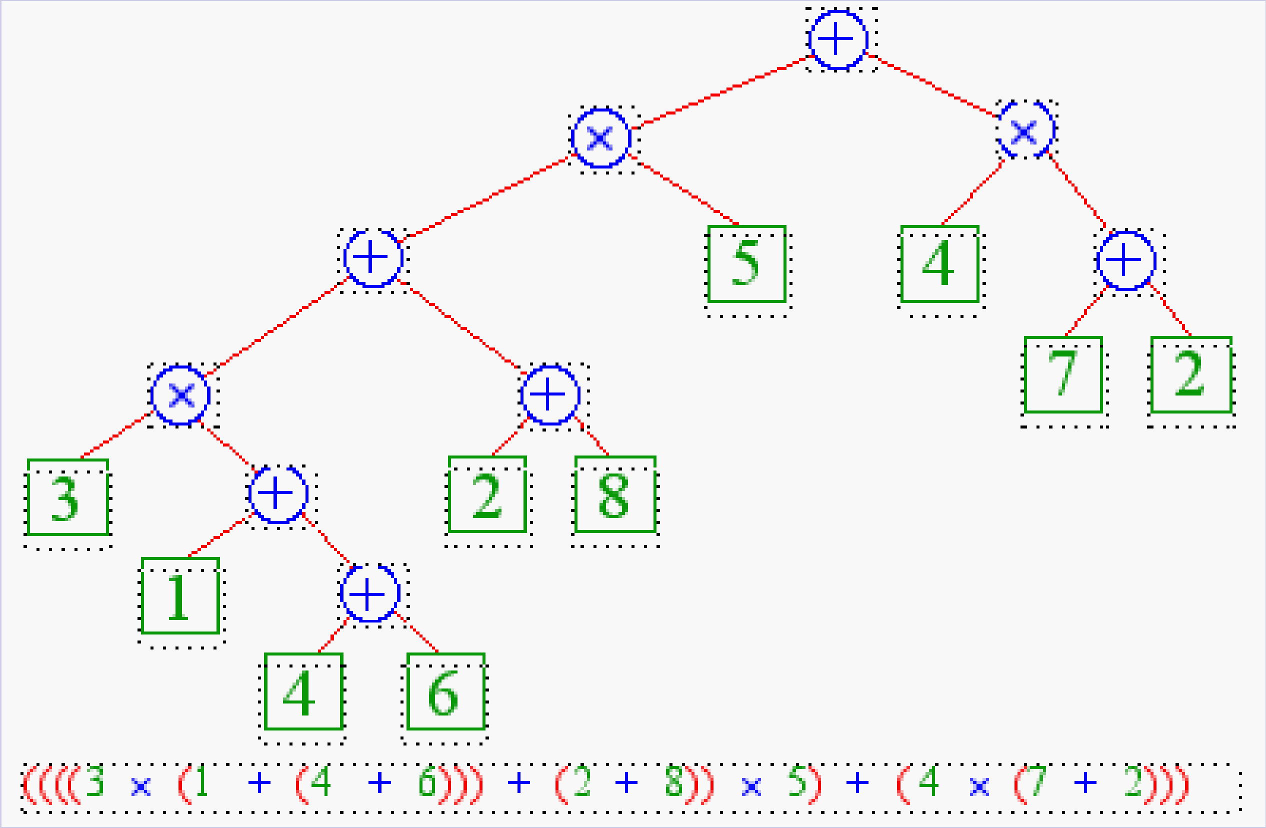
A binary tree is either a

- leaf or
- An internal node (the root) and one/two binary trees (left subtree and/or right subtree).



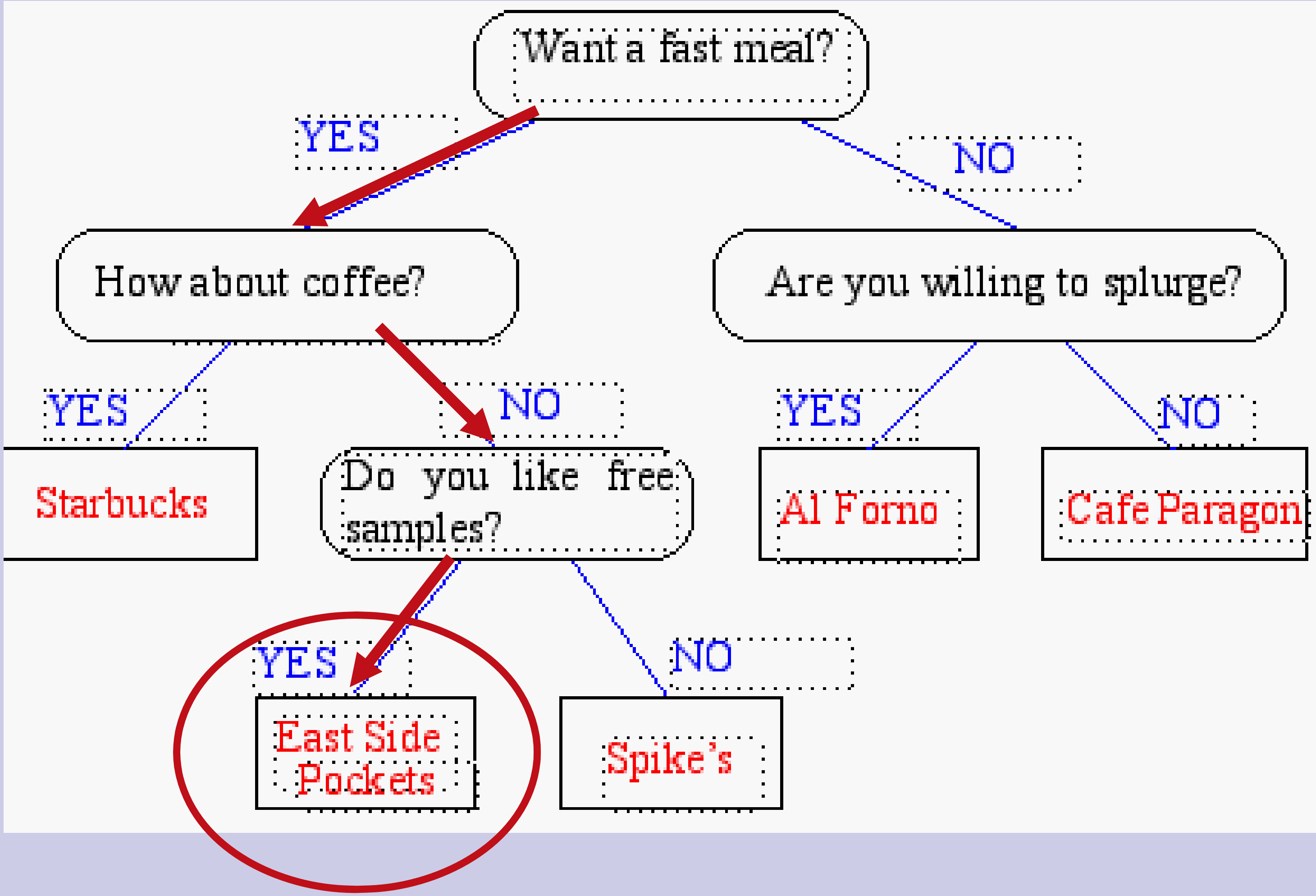
Examples of Binary Trees

arithmetic expressions

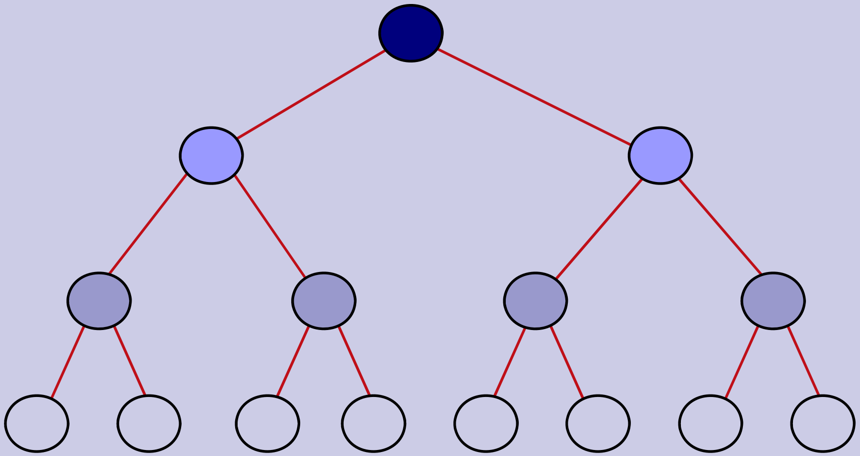


Examples of Binary Trees

decision trees



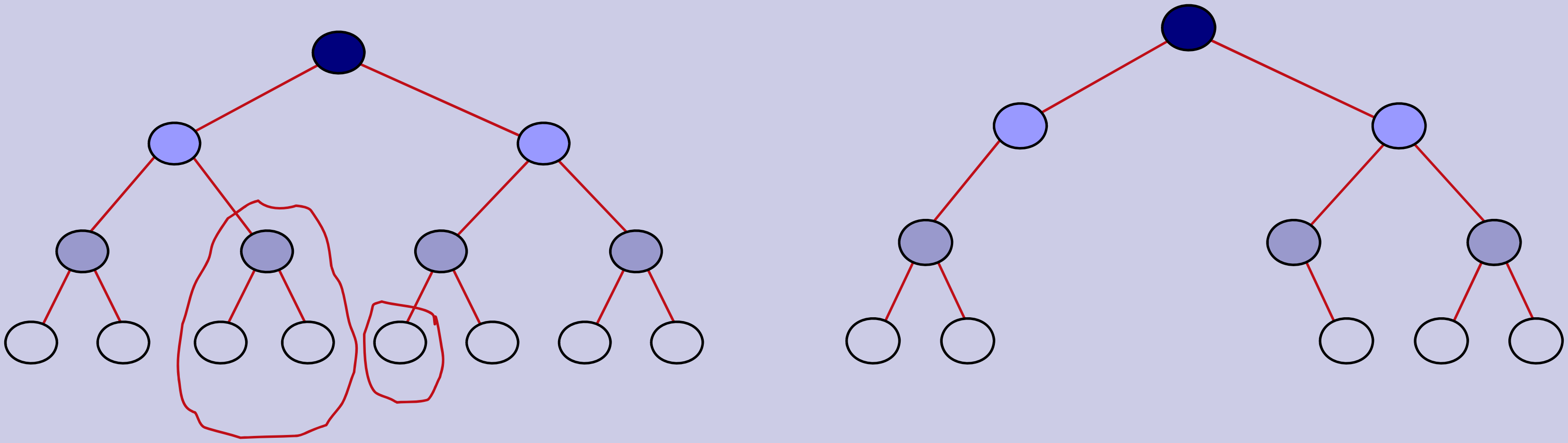
Complete Binary tree



- level i has 2^i nodes
- In a tree of height h
  - leaves are at level h
  - No. of leaves is 2^h
  - No. of internal nodes = $1+2+2^2+\ldots+2^{h-1} = 2^h-1$
  - No of internal nodes = no of leaves -1
  - Total no. of nodes is $2^{h+1}-1 = n$
- In a tree of n nodes
  - No of leaves is $(n+1)/2$
  - Height = $\log_2$ (no of leaves)

Binary Tree

- A Binary tree can be obtained from an appropriate complete binary tree by pruning

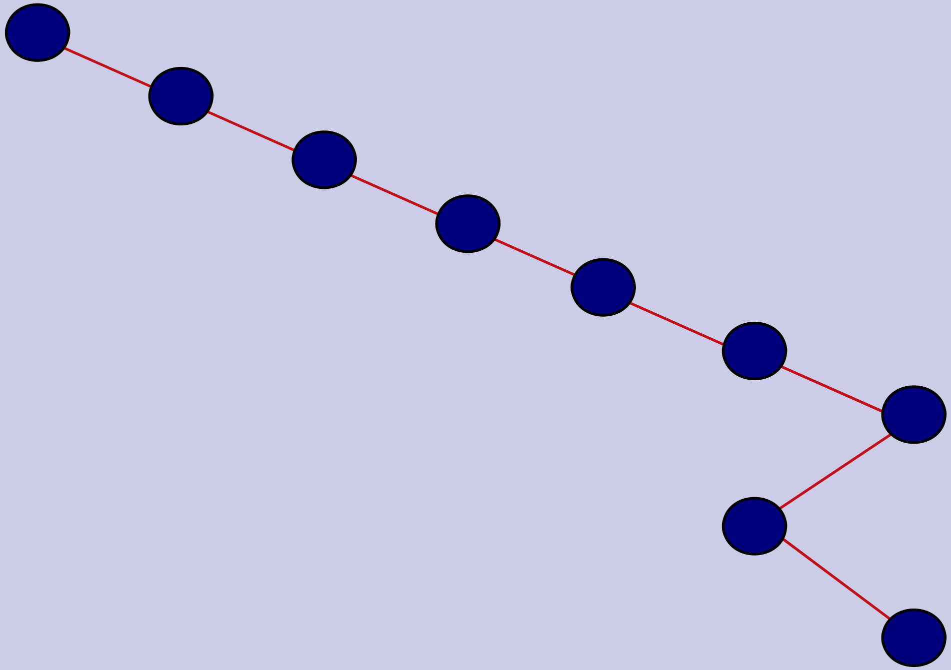


Minimum height of a binary tree

- A binary tree of height h has
  - At most 2^i nodes at level i
  - At most $1+2+2^2+\ldots+2^h = 2^{h+1}-1$ nodes
- If the tree has n nodes then
  - $n <= 2^{h+1}-1$
  - Hence $h >= \log_2 (n+1)/2$

Maximum height of a binary tree

- A binary tree on n nodes has height at most n-1
- This is obtained when every node (except the leaf) has exactly one child



No of leaves in a binary tree

- no of leaves <= 1+ no of internal nodes.
- Proof: by induction on no of internal nodes
  - Tree with 1 node has a leaf but no internal node.
  - Assume stmt is true for tree with k-1 internal nodes.
  - A tree with k internal nodes has k_1 internal nodes in left subtree and $(k-k_1-1)$ internal nodes in right subtree.
  - No of leaves $\le (k_1+1)+(k-k_1) = k+1$

leaves in a binary tree (2)

For a binary tree on n nodes

- No of leaves + no of internal nodes = n
- No of leaves <= no of internal nodes + 1
- Hence, no of leaves <= (n+1)/2
- Minimum no of leaves is 1

ADTs for Trees

- generic container methods: size(), isEmpty(), elements()
- positional container methods: positions(), swapElements(p,q), replaceElement(p,e)
- query methods: isRoot(p), isInternal(p), isExternal(p)
- accessor methods: root(), parent(p), children(p)
- update methods: application specific

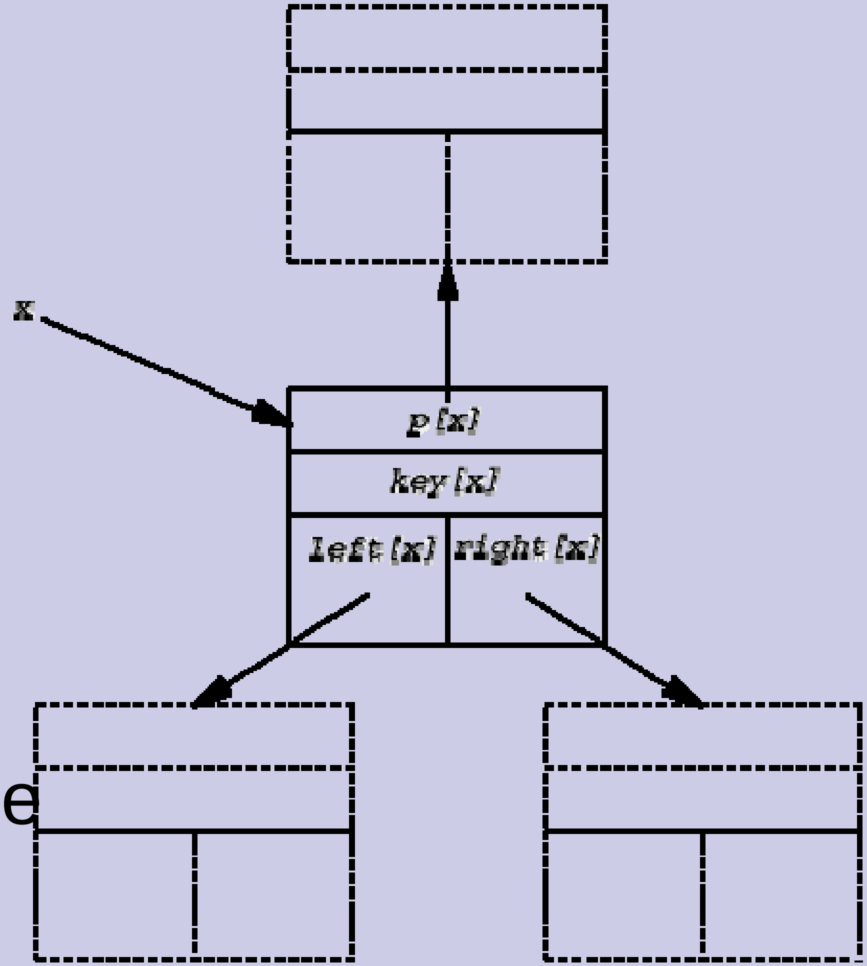
ADTs for Binary Trees

- accessor methods: leftChild(p), rightChild(p), sibling(p)
- update methods:
  - expandExternal(p), removeAboveExternal(p)
  - other application specific methods

The Node Structure

Each node in the tree contains

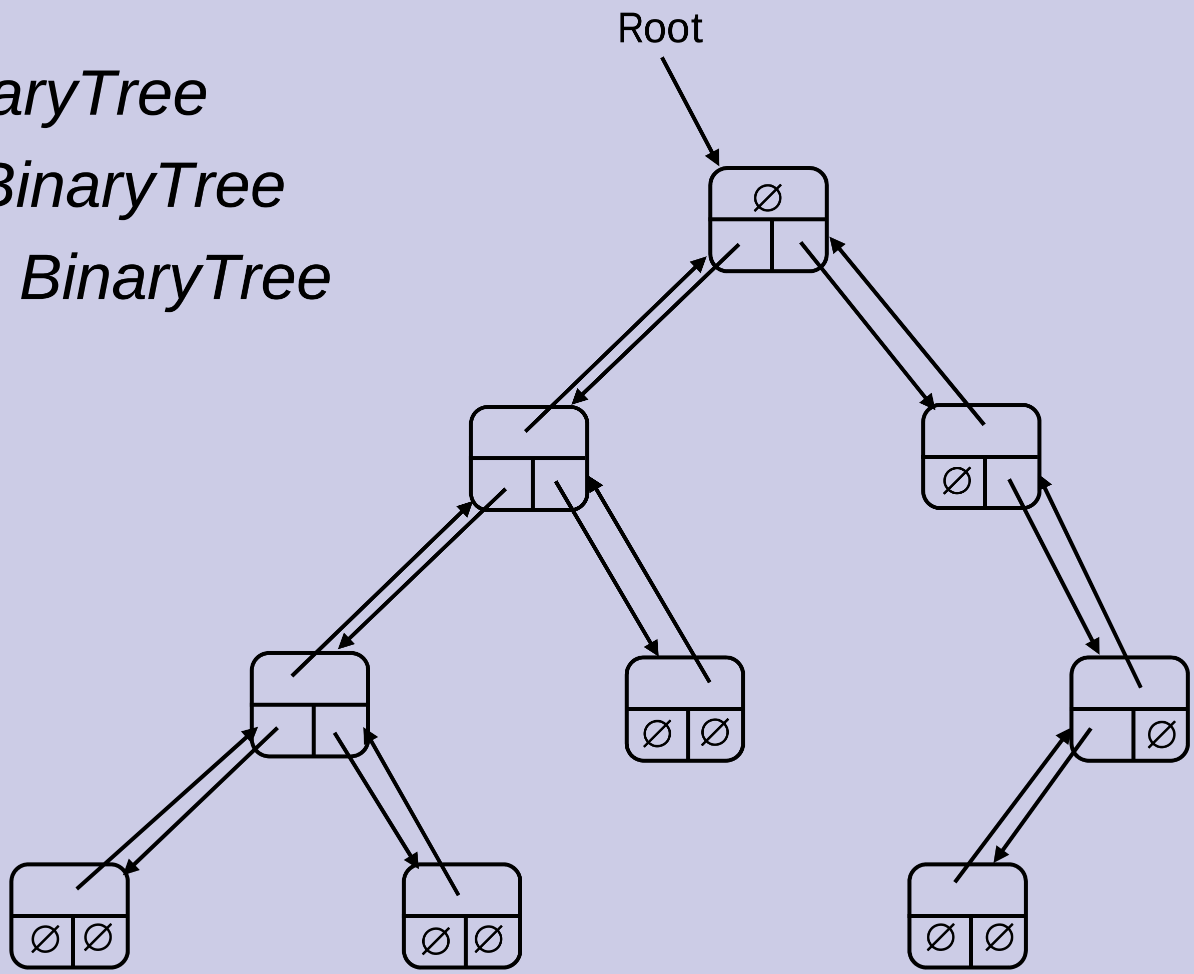
- *key*[*x*] – key
- *left*[*x*] – pointer to left child
- *right*[*x*] – pt. to right child
- *p*[*x*] – pt. to parent node



Representing Rooted Trees

BinaryTree:

- **Parent:** *BinaryTree*
- **LeftChild:** *BinaryTree*
- **RightChild:** *BinaryTree*



Unbounded Branching

UnboundedTree:

- **Parent:** *UnboundedTree*
- **LeftChild:** *UnboundedTree*
- **RightSibling:** *UnboundedTree*