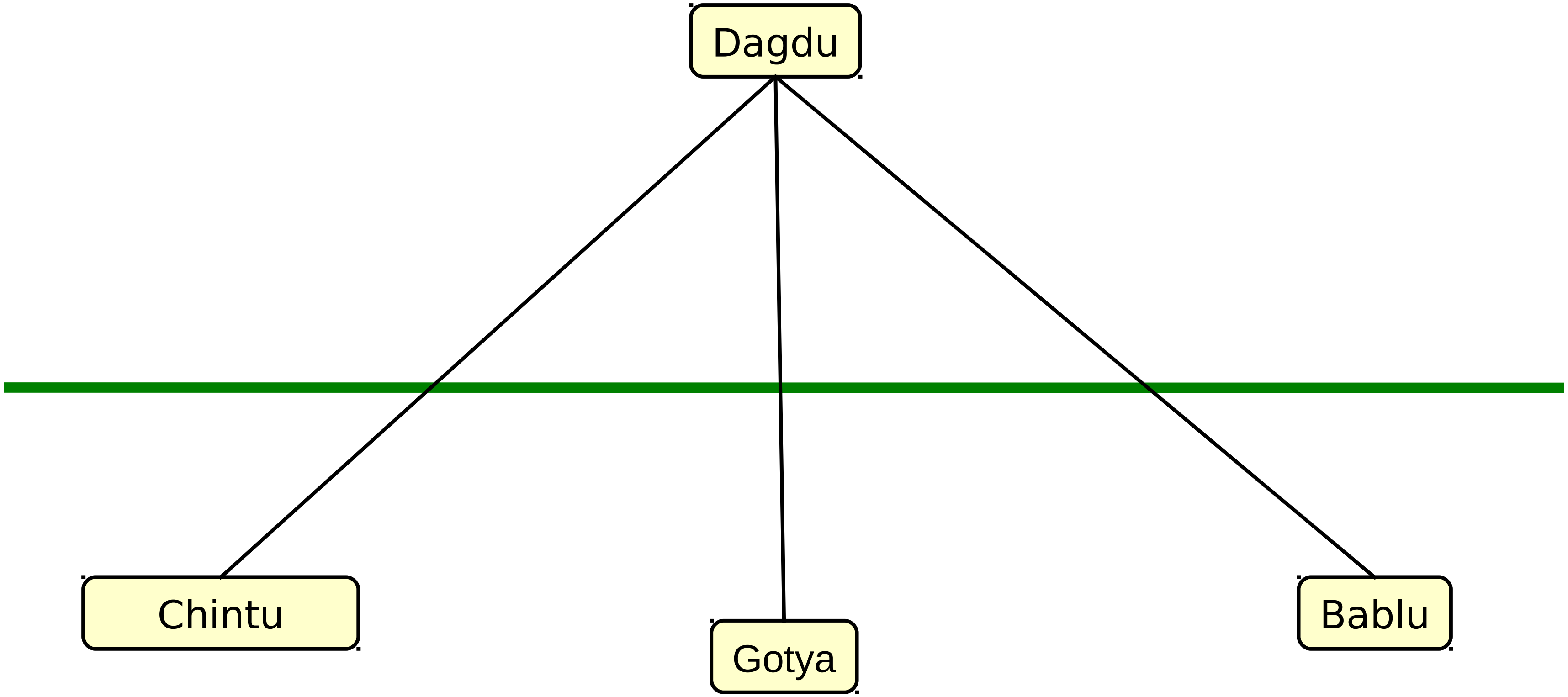
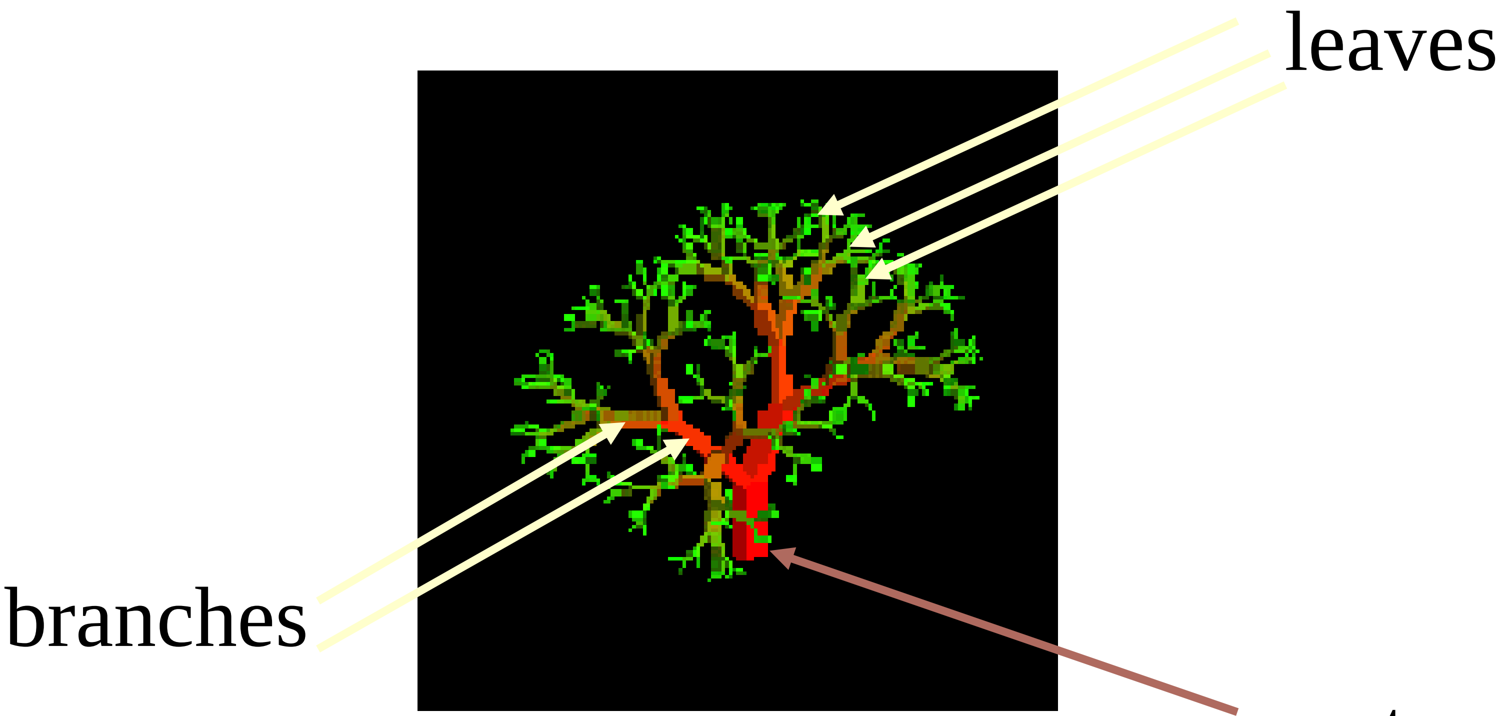
Trees and Binary Trees

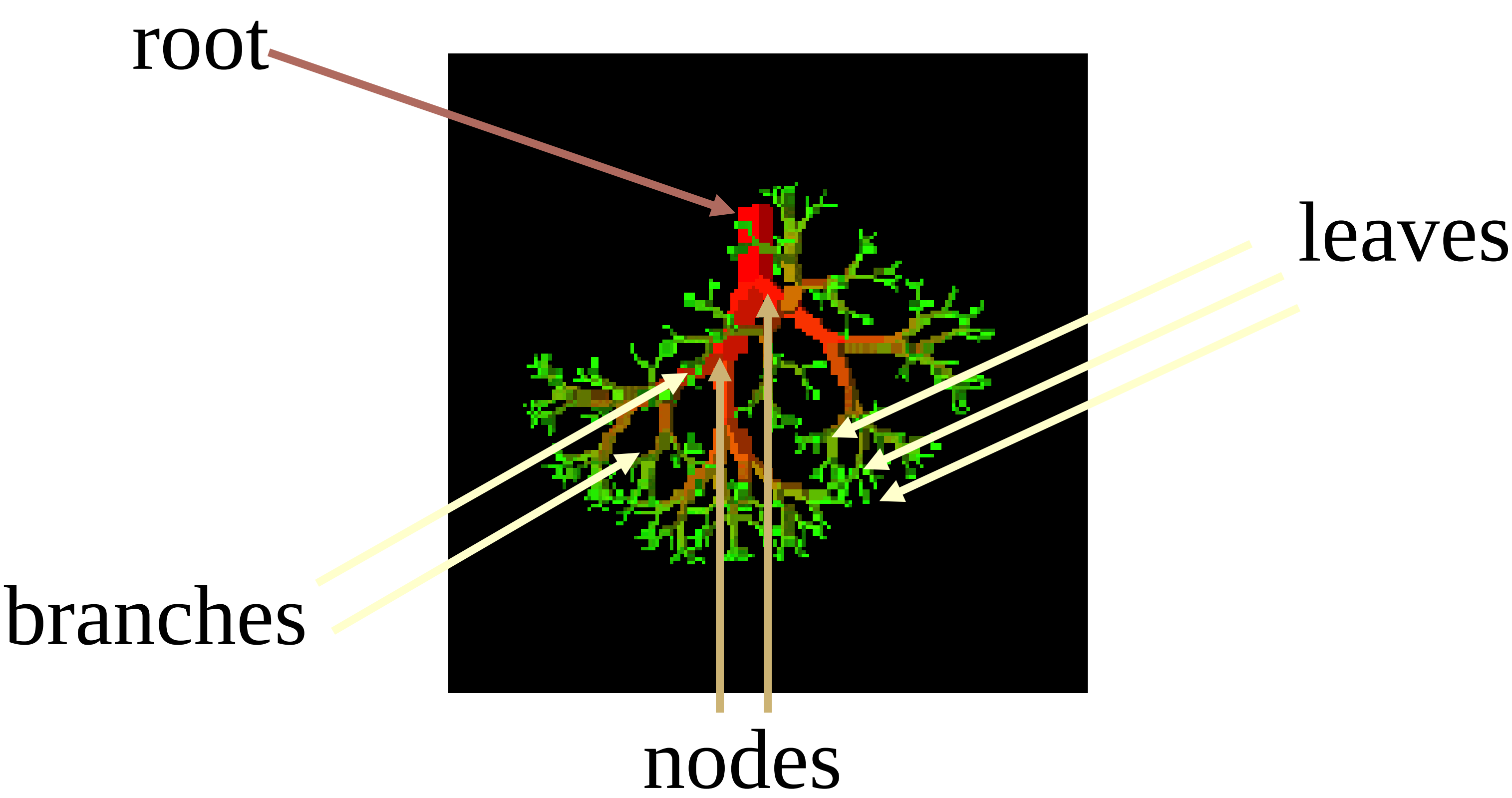


Theese notes are a compilation and edition from many sources. No claim of intellectual property or ownership of the lecture notes/ppt is being done here.

Nature's View of a Tree

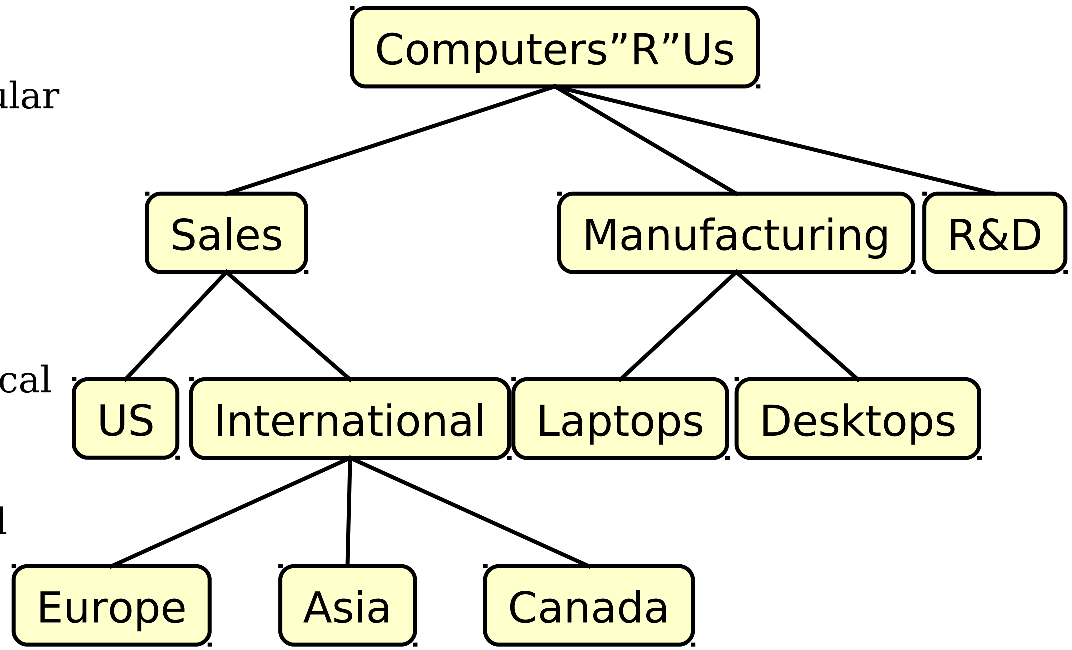


Computer Scientist's View



What is a Tree

- A tree is data structure, like a linked list, but, instead of each node pointing simply to the next node in a linear fashion, each node may point to multiple nodes.
- Non linear data structure (no particular sequence)
- A tree is a finite nonempty set of elements.
- It is an abstract model of a hierarchical structure.
- Consists of nodes with a parent-child relation.
- Applications:
  - Organization charts
  - File systems
  - Programming environments

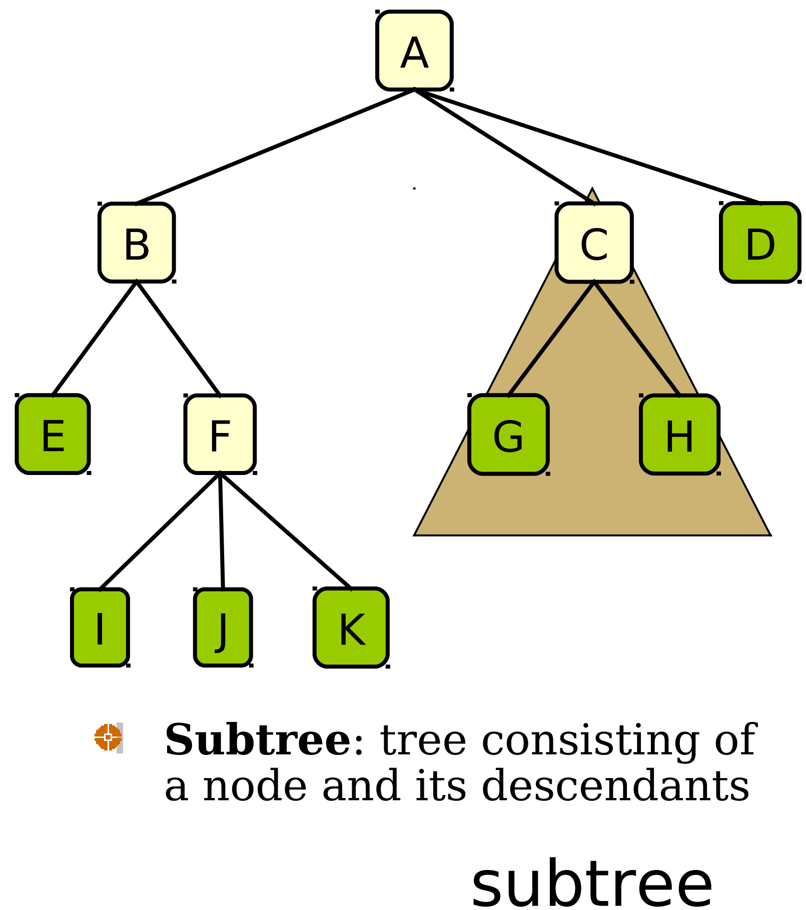


Terminology (Definitions)

- **Node:** Elements of tree are called as nodes
- **Root**: node without parent (A)
- **Edge**: refers to the link from parent to child
- **Siblings**: nodes share the same parent
- **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. Node p is ancestor of node q if there exists a path from root to q and p appears on this path.
- **Descendant** of a node: child, grandchild, grand-grandchild, etc.

  Node q is descendant of p in the above case.
- **Depth** of a node: number of ancestors from the **node to root** node
- **Height** of a tree: maximum depth of any node (3)
- **Degree** of a node: the number of its children
- **Degree** of a tree: the maximum number of a node in the tree.

- **Level**: Set of all nodes at a given depth. (root at level zero)

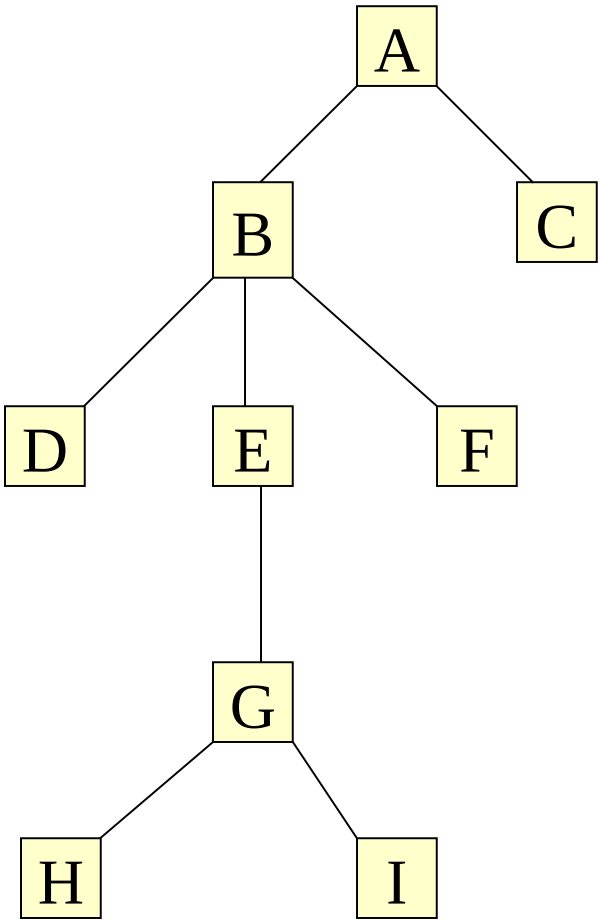


- **Subtree**: tree consisting of a node and its descendants

Terminology (Definitions)

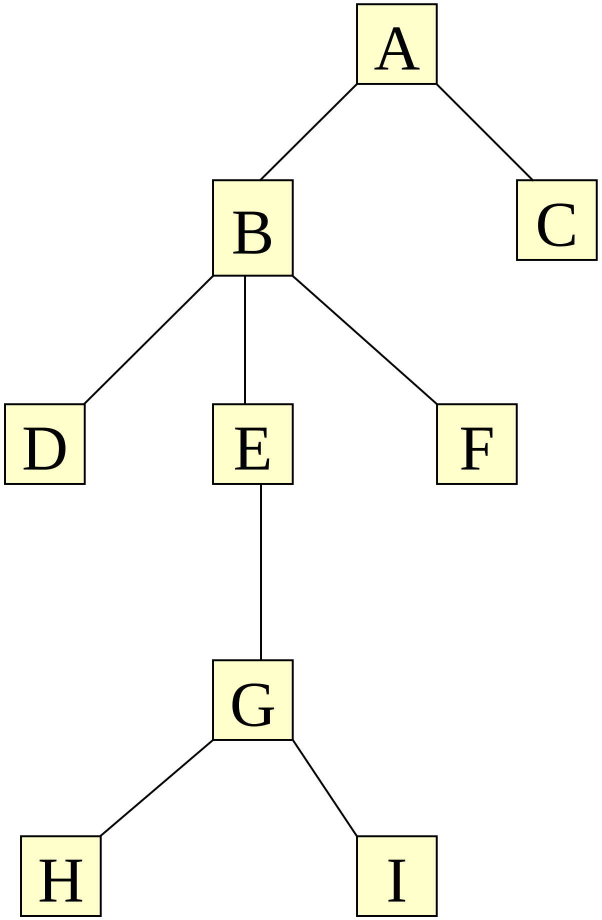
- **Size of a node ::** number of descendants it has including itself.
- **Length of path:** is the number of adjacent connections(links) which is one less than the number of adjacent nodes that it connects.
- **Singleton Tree:** root is the only node in the tree.
- **Path length of tree**: Sum of the lengths of all paths from its root. (weighted sum of (level * number of nodes)
- **Full tree:** A tree is said to be full if all of its internal nodes have the same degree and all of its leaves are at the same level.
- **# of nodes**:: Full tree of degree **d** and height **h** has ((d^h+1) – 1)/(d – 1) nodes.
- **Tree can be defined recursively as::** A tree is either empty or a single node (root) with a sequence of zero or more disjoint tree.

Tree Properties



Property	**Value**
Number of nodes	
Height	
Root Node	
Leaves	
Interior nodes	
Ancestors of H	
Descendants of B	
Siblings of E	
Right subtree of A	
Degree of this tree	

Tree Properties



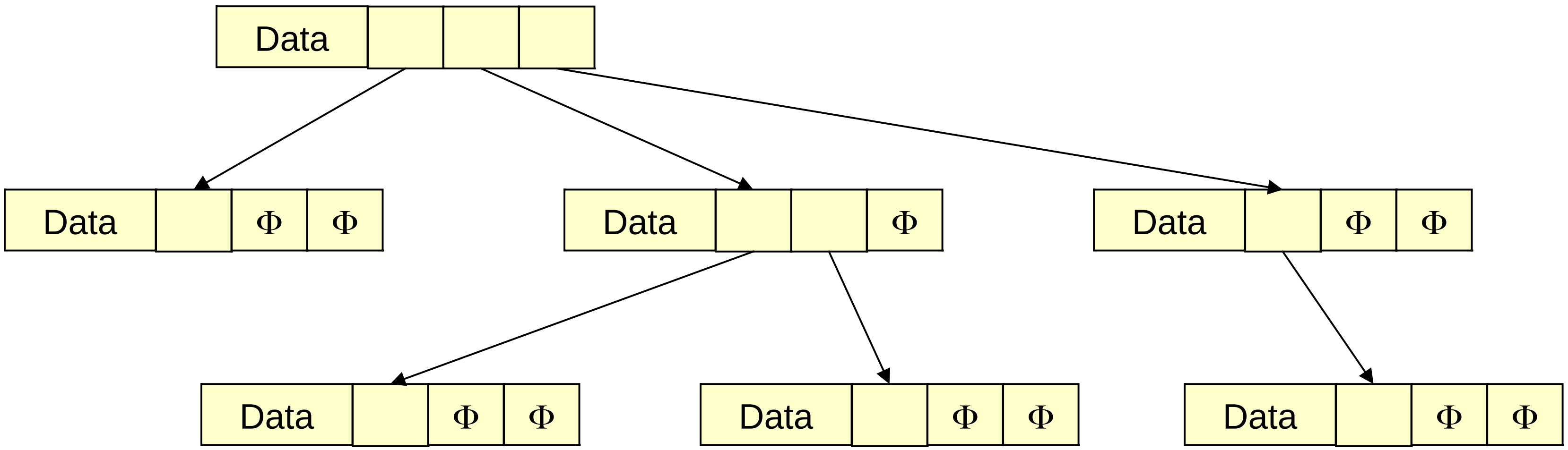
Property	**Value**
Number of nodes	9
Height	4
Root Node	A
Leaves	H, I, D, F, C
Interior nodes	B, E, G
Ancestors of H	G, E, B, A
Descendants of B	D, E, F,G,H,I
Siblings of E	G
Right subtree of A	C
Degree of this tree	3

Tree ADT

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

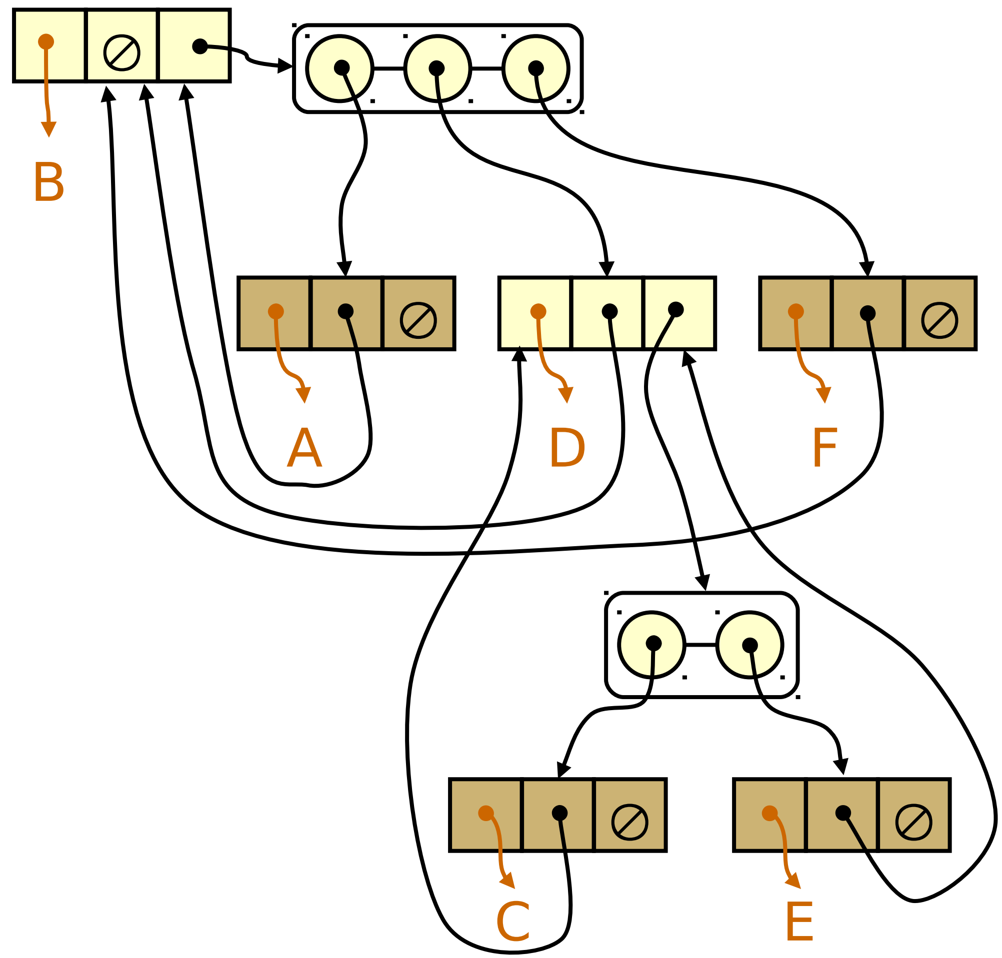
Trees Nodes - Representation

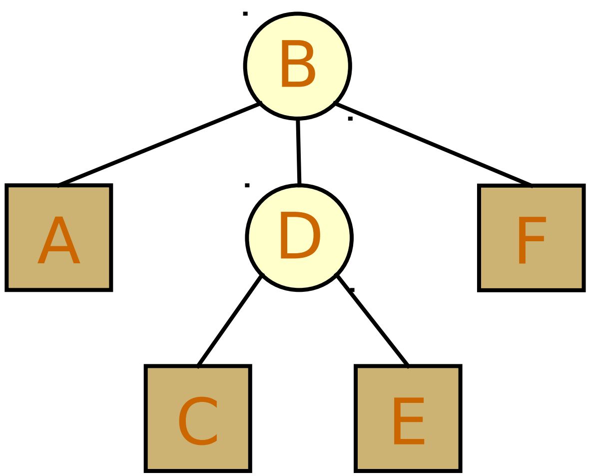
- Every tree node:
  - object – useful information
  - children – pointers to its children



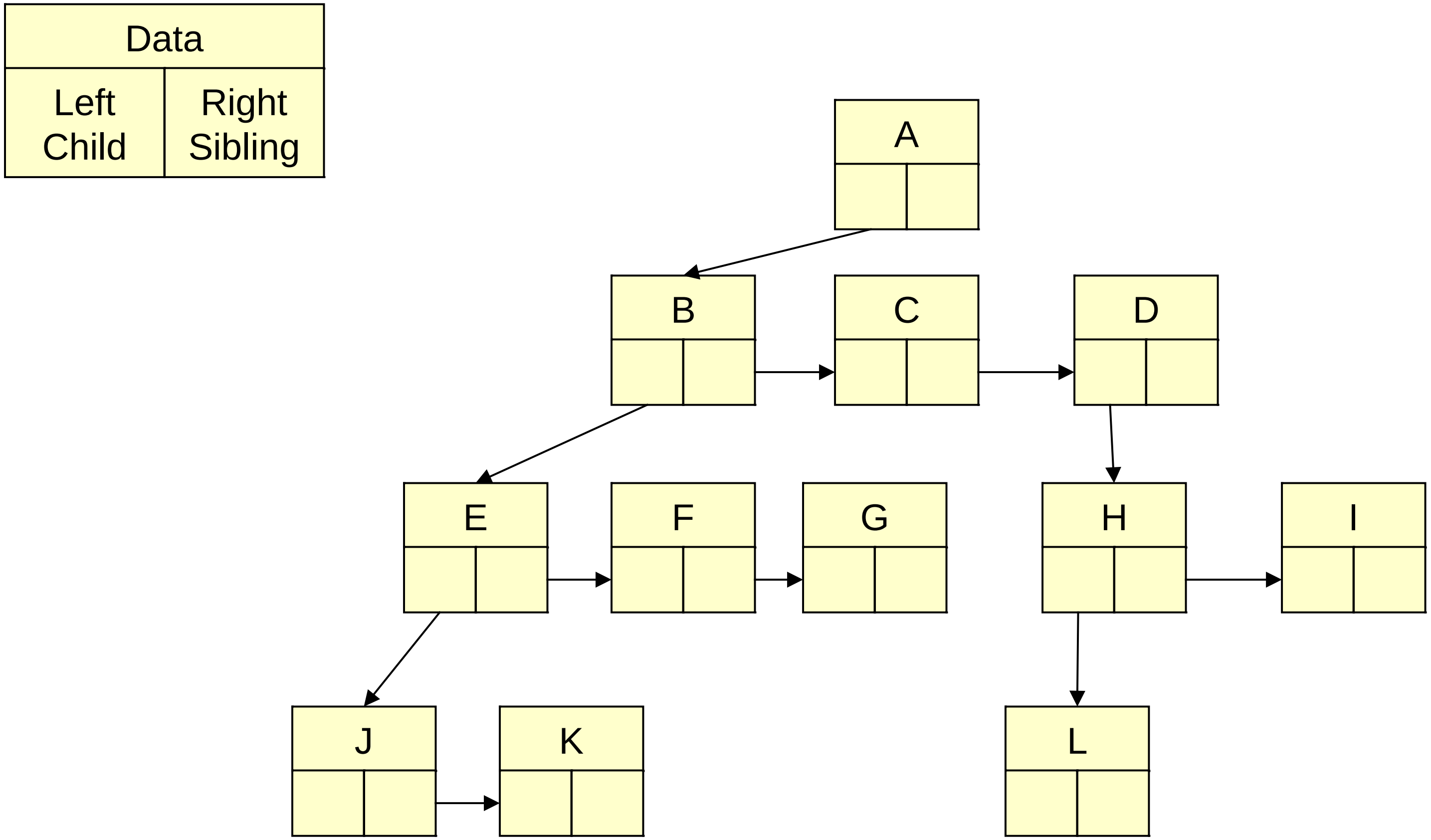
A Tree Representation

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





Left Child, Right Sibling Representation



Tree Traversal

- Two main methods:
  - Preorder
  - Postorder
- Recursive definition

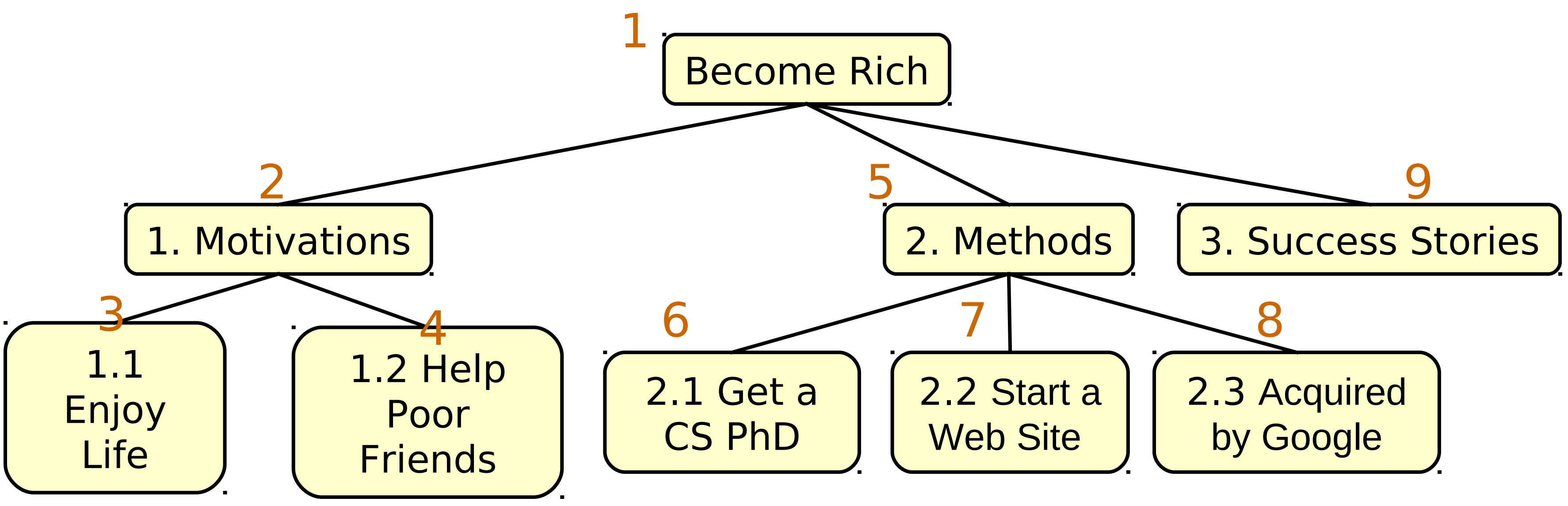
- Preorder:
  - visit the root
  - traverse in preorder the children (subtrees)

- Postorder
  - traverse in postorder the children (subtrees)
  - visit the root

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

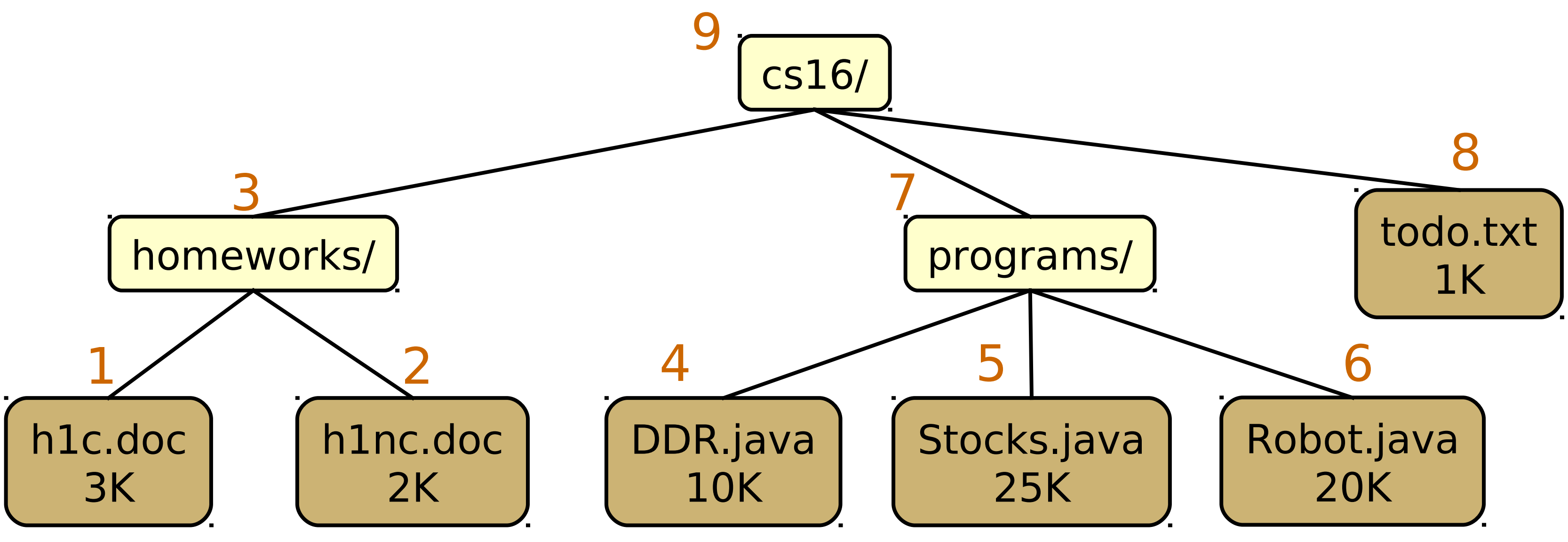
Algorithm ***preOrder***(*v*)
    visit(*v*)
    for each child ***w*** of ***v***
        preorder (***w***)



Postorder Traversal

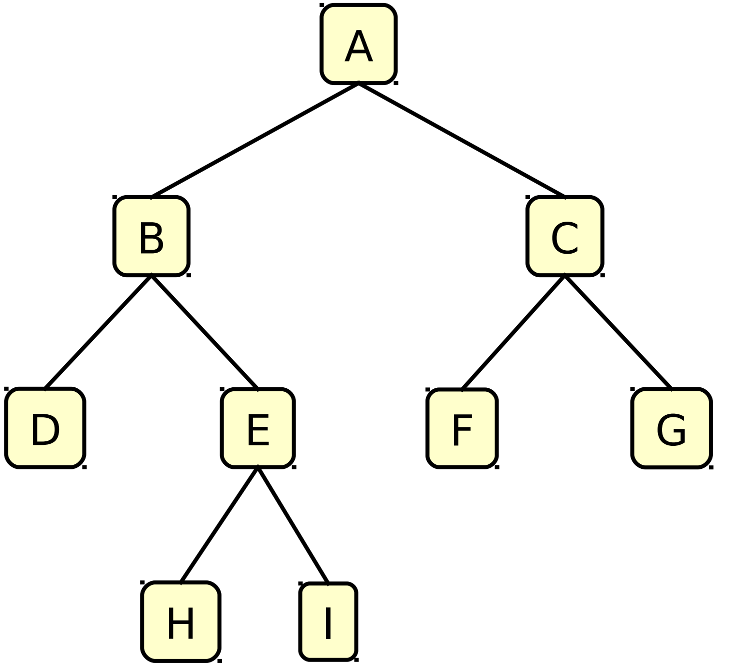
- In a postorder traversal, a node is visited after its descendants
- Application: compute space used by files in a directory and its subdirectories

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



Binary Tree

- A binary tree is a tree with the following properties:
  - Each internal node has at most two children (degree of two)
  - 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



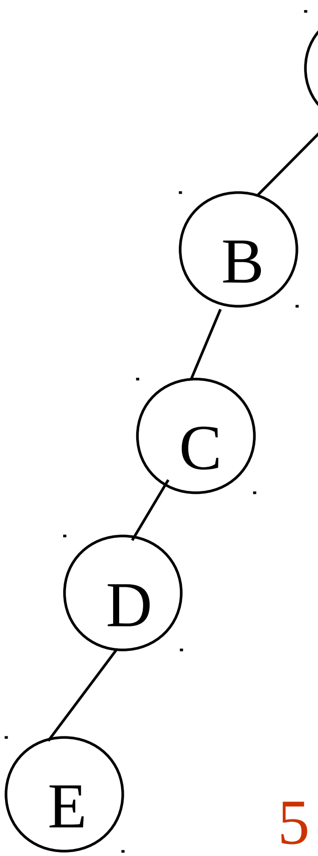
BinaryTree ADT

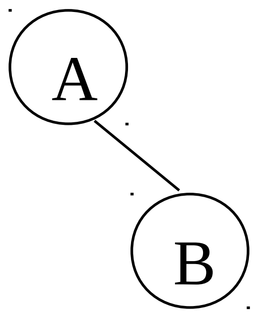
- The BinaryTree ADT extends the Tree ADT, i.e., it inherits all the methods of the Tree ADT
- Additional methods:
  - position leftChild(p)
  - position rightChild(p)
  - position sibling(p)
- Update methods may be defined by data structures implementing the BinaryTree ADT

Examples of the Binary Tree

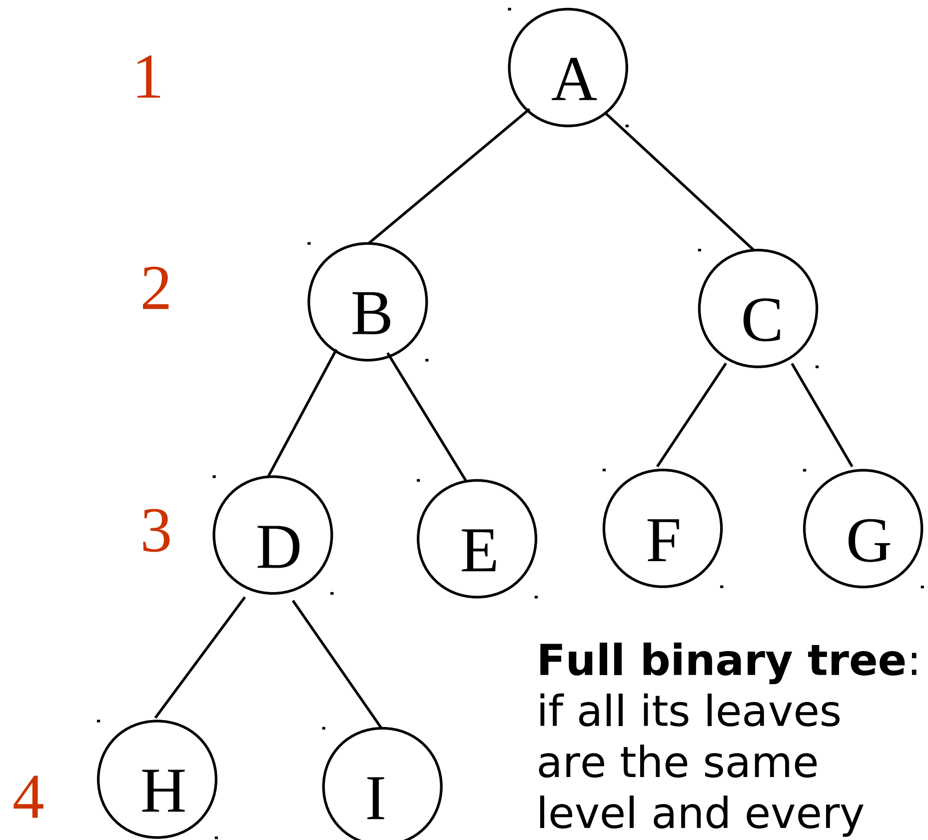
Every node has only one child

Skewed Binary Tree





Complete Binary Tree



Complete binary tree: is either a full binary tree or can be made made a full binary tree by just adding leaves only in an uninterrupted segment of nodes in the right of the bottom level.

Full binary tree: if all its leaves are the same level and every interior node has two children.

Examples of the Binary Tree

Consider full binary trees of size 2,3,4 and 5.
Draw the possibilities of FBTs.

- If h represents height of a full binary tree then
  It has L=2^h leaves and (2^h)-1 internal nodes.

- # of Nodes in FBT = (2^(h+1))-1

- If n represents number of nodes in the FBT, then
h=lg(n+1)-1

- In any BT, (h+1) <= n <= ((2^(h+1))-1 and
  l Lg n l <= h <= n-1

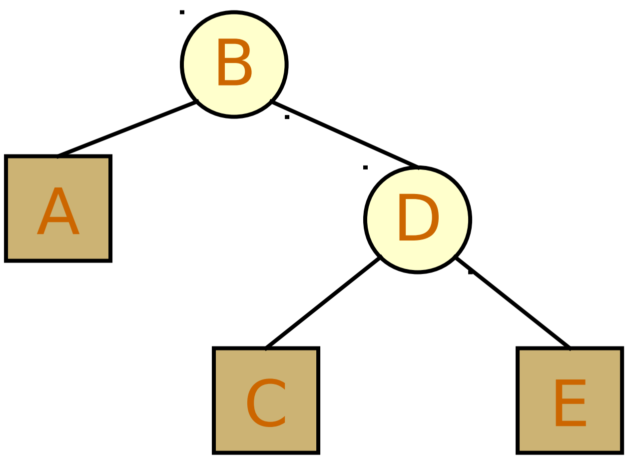
Differences Between A Tree and A Binary Tree

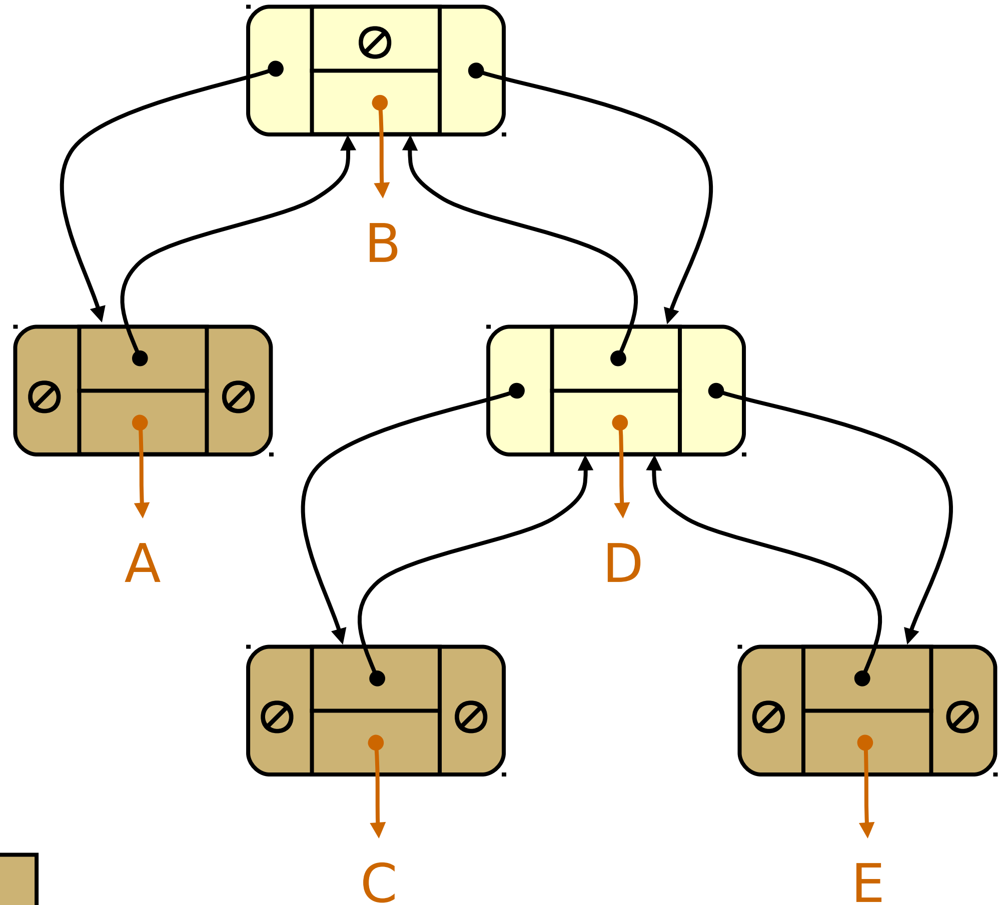
- The subtrees of a binary tree are ordered; those of a tree are not ordered.

- Are different when viewed as binary trees.
- Are the same when viewed as trees.

Data Structure for Binary Trees

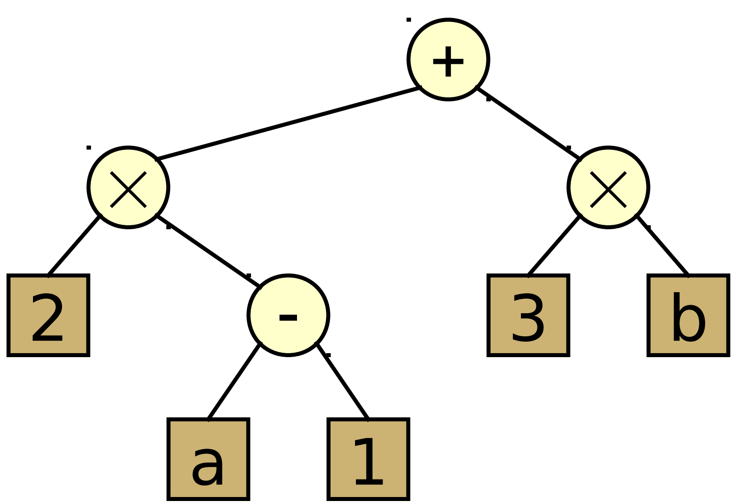
- A node is represented by an object storing
  - Element
  - Parent node
  - Left child node
  - Right child node





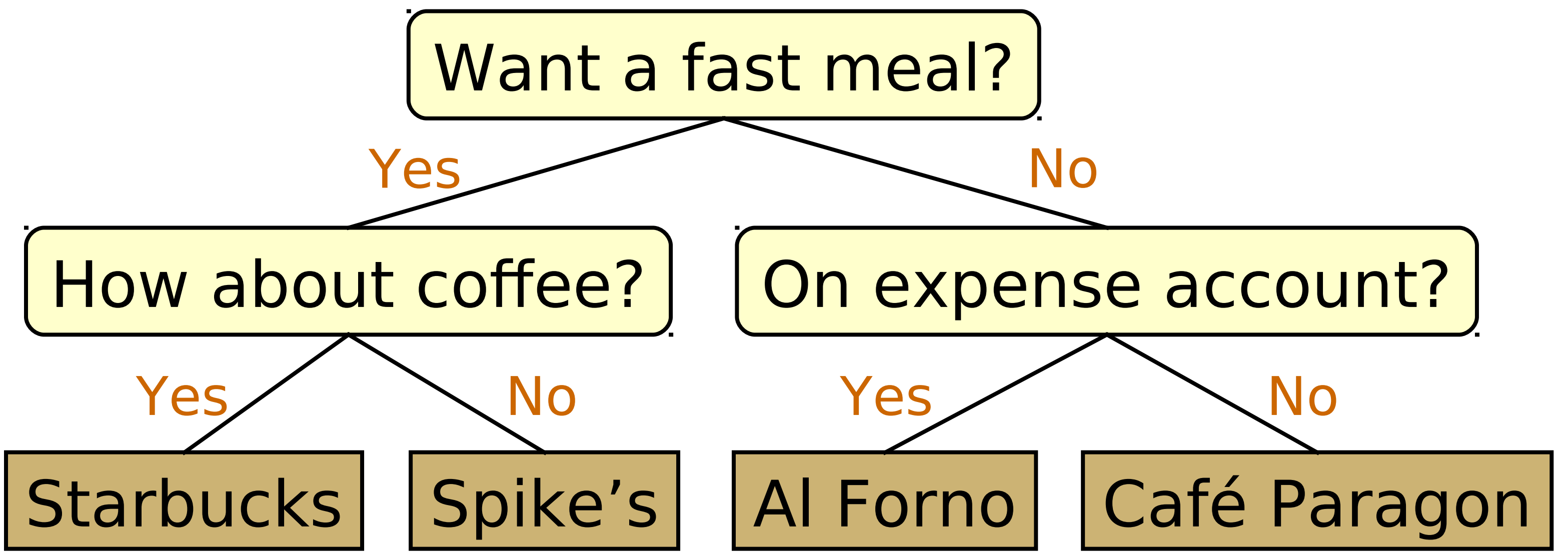
Arithmetic Expression Tree

- 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



Maximum Number of Nodes in Binary Tree

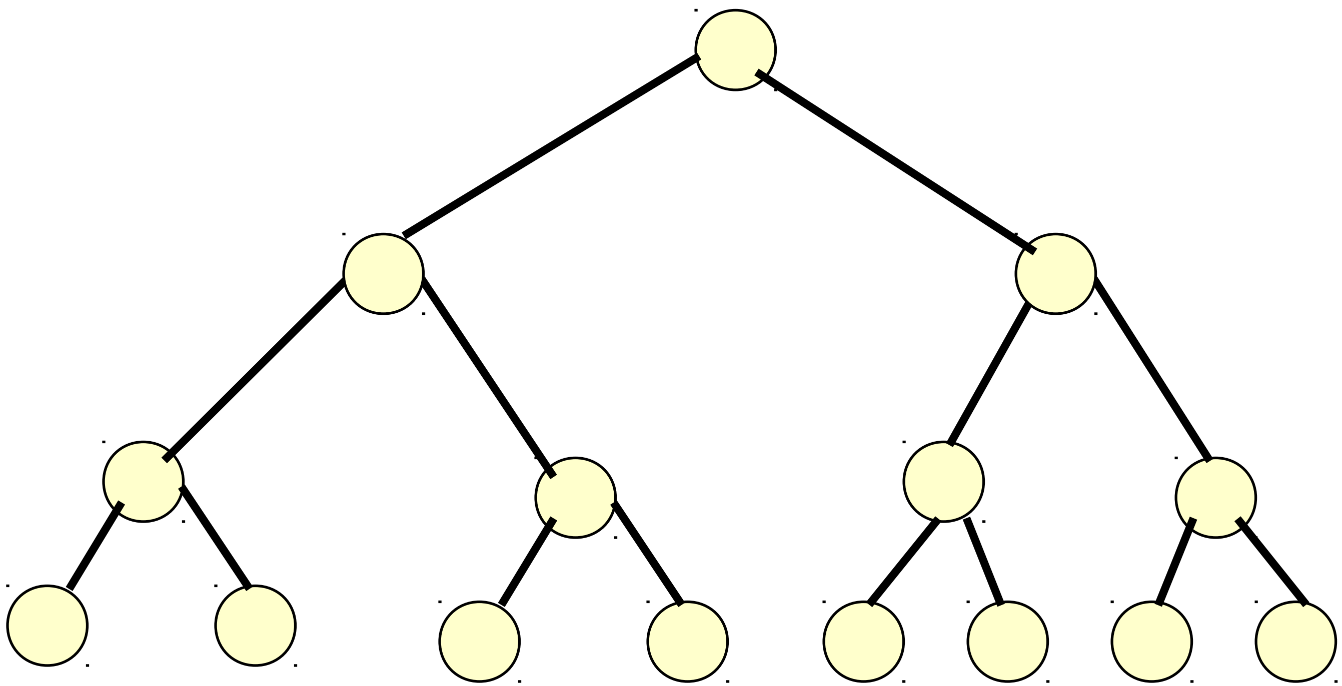
- The maximum number of nodes on depth i of a binary tree is 2^i, $i>=0$.
- The maximum nubmer of nodes in a binary tree of height k is $2^{k+1}-1$, $k>=0$.

Prove by induction.

$$\sum_{i=0}^{k} 2^i = 2^{k+1} - 1$$

Full Binary Tree

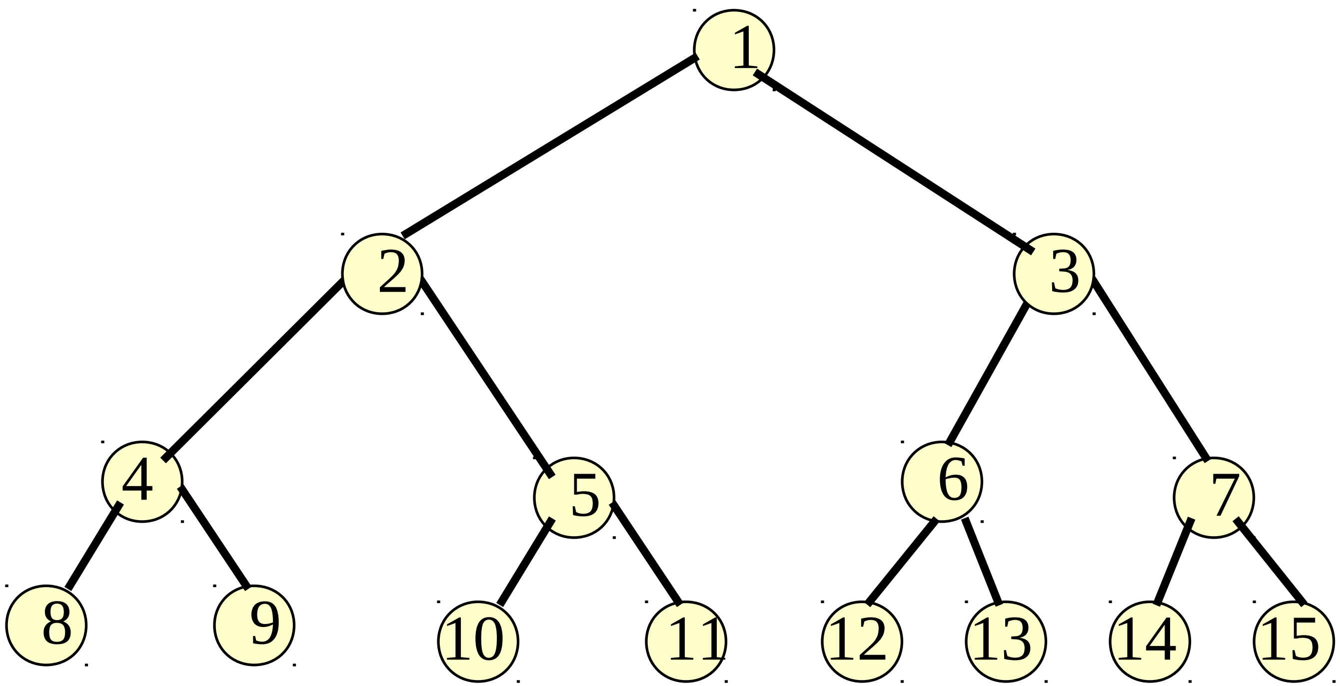
- A full binary tree of a given height k has $2^{k+1}–1$ nodes.



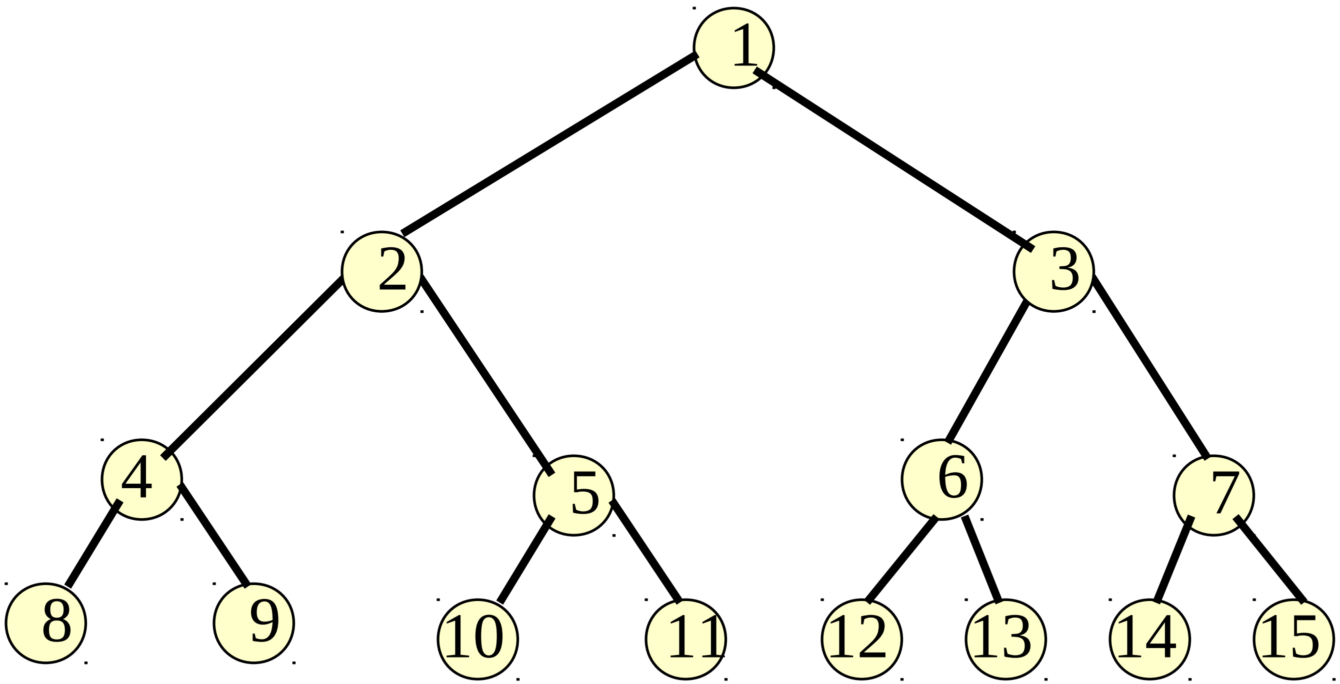
Height 3 full binary tree.

Labeling Nodes In A Full Binary Tree

- Label the nodes 1 through $2^{k+1} - 1$.
- Label by levels from top to bottom.
- Within a level, label from left to right.

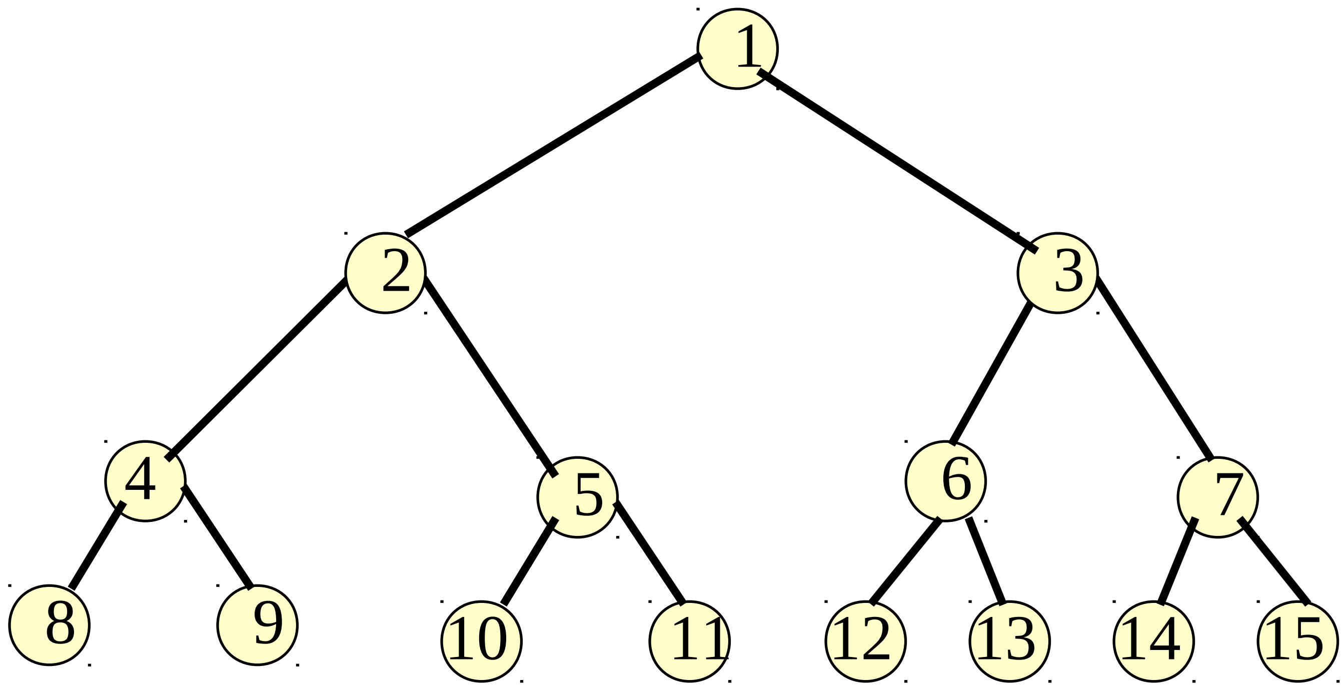


Node Number Properties



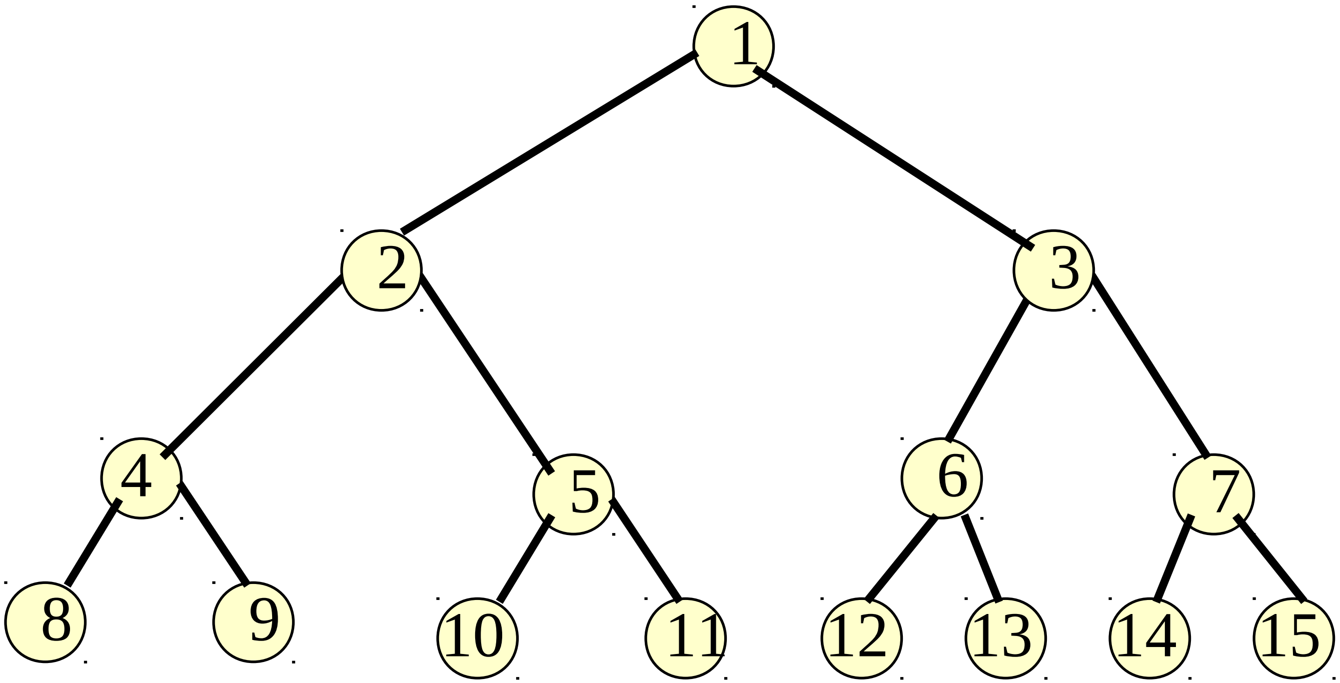
- Parent of node i is node i / 2, unless i = 1.
- Node 1 is the root and has no parent.

Node Number Properties



- Left child of node i is node 2i, unless 2i > n, where n is the number of nodes.
- If 2i > n, node i has no left child.

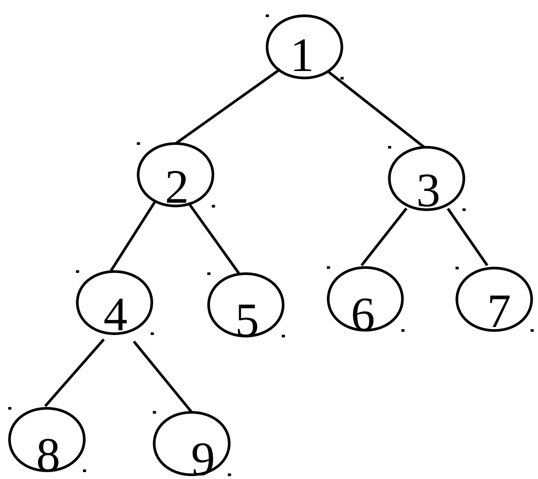
Node Number Properties



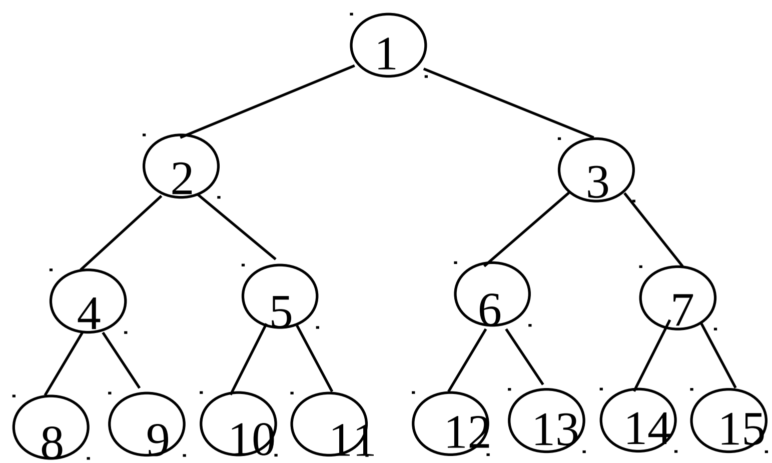
- Right child of node i is node 2i+1, unless 2i+1 > n, where n is the number of nodes.
- If 2i+1 > n, node i has no right child.

Complete Binary Trees

- A labeled binary tree containing the labels 1 to n with root 1, branches leading to nodes labeled 2 and 3, branches from these leading to 4, 5 and 6, 7, respectively, and so on.
- A binary tree with *n* nodes and level *k* is complete *iff* its nodes correspond to the nodes numbered from 1 to *n* in the full binary tree of level *k*.



Complete binary tree



Full binary tree of depth 3

Binary Tree Traversals

- Let l, R, and r stand for moving left, visiting the node, and moving right.
- There are six possible combinations of traversal
  - lRr, lrR, Rlr, Rrl, rRl, rlR
- Adopt convention that we traverse left before right, only 3 traversals remain
  - lRr, lrR, Rlr
  - inorder, postorder, preorder