Terminology

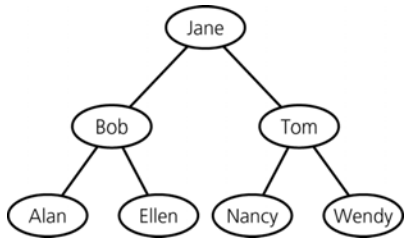
- Definition of a general tree
  - A general tree T is a set of one or more nodes such that T is partitioned into disjoint subsets:
    - A single node r, the root
    - Sets that are general trees, called subtrees of r
- Definition of a binary tree
  - A binary tree is a set T of nodes such that either
    - T is empty, or
    - T is partitioned into three disjoint subsets:
      - A single node r, the root
      - Two possibly empty sets that are binary trees, called left and right subtrees of r

Terminology

- A binary search tree
  - A binary tree that has the following properties for each node n
    - n's value is greater than all values in its left subtree T_L
    - n's value is less than all values in its right subtree T_R
    - Both T_L and T_R are binary search trees

Figure 11-5

A binary search tree of names



Terminology

- The height of trees
  - Level of a node n in a tree T
    - If n is the root of T, it is at level 1
    - If n is not the root of T, its level is 1 greater than the level of its parent
  - Height of a tree T defined in terms of the levels of its nodes
    - If T is empty, its height is 0
    - If T is not empty, its height is equal to the maximum level of its nodes

Terminology

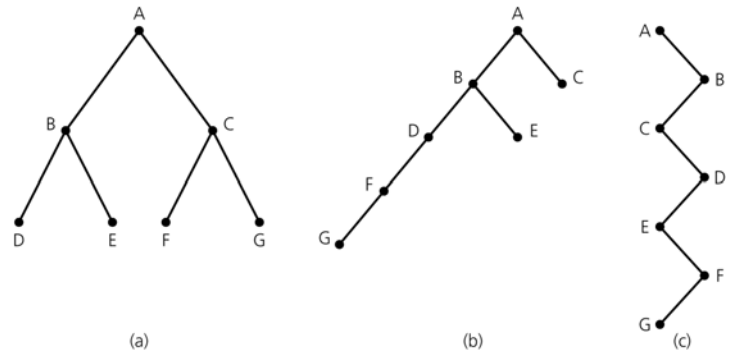


Figure 11-6

Binary trees with the same nodes but different heights

Terminology

- Full, complete, and balanced binary trees
  - Recursive definition of a full binary tree
    - If T is empty, T is a full binary tree of height 0
    - If T is not empty and has height h > 0, T is a full binary tree if its root's subtrees are both full binary trees of height h – 1

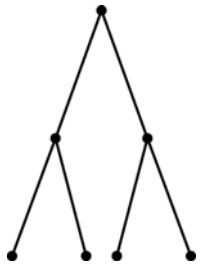


Figure 11-7

A full binary tree of height 3

Terminology

- Complete binary trees
  - A binary tree T of height h is complete if
    - All nodes at level h – 2 and above have two children each, and
    - When a node at level h – 1 has children, all nodes to its left at the same level have two children each, and
    - When a node at level h – 1 has one child, it is a left child

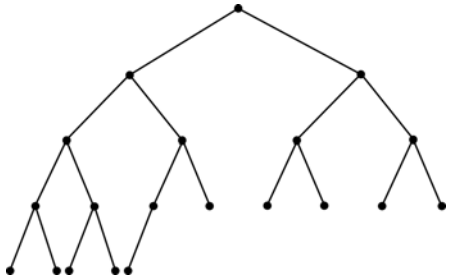


Figure 11-8

A complete binary tree

Terminology

- Balanced binary trees
  - A binary tree is balanced if the height of any node's right subtree differs from the height of the node's left subtree by no more than 1
- Full binary trees are complete
- Complete binary trees are balanced

Terminology

- Summary of tree terminology
  - General tree
    - A set of one or more nodes, partitioned into a root node and subsets that are general subtrees of the root
  - Parent of node n
    - The node directly above node n in the tree
  - Child of node n
    - A node directly below node n in the tree
  - Root
    - The only node in the tree with no parent

Terminology

- Summary of tree terminology (Continued)
  - Leaf
    - A node with no children
  - Siblings
    - Nodes with a common parent
  - Ancestor of node n
    - A node on the path from the root to n
  - Descendant of node n
    - A node on a path from n to a leaf
  - Subtree of node n
    - A tree that consists of a child (if any) of n and the child's descendants

Terminology

- Summary of tree terminology (Continued)
  - Height
    - The number of nodes on the longest path from the root to a leaf
  - Binary tree
    - A set of nodes that is either empty or partitioned into a root node and one or two subsets that are binary subtrees of the root
    - Each node has at most two children, the left child and the right child
  - Left (right) child of node n
    - A node directly below and to the left (right) of node n in a binary tree

Terminology

- Summary of tree terminology (Continued)
  - Left (right) subtree of node n
    - In a binary tree, the left (right) child (if any) of node n plus its descendants
  - Binary search tree
    - A binary tree where the value in any node n is greater than the value in every node in n's left subtree, but less than the value of every node in n's right subtree
  - Empty binary tree
    - A binary tree with no nodes

Terminology

- Summary of tree terminology (Continued)
  - Full binary tree
    - A binary tree of height h with no missing nodes
    - All leaves are at level h and all other nodes each have two children
  - Complete binary tree
    - A binary tree of height h that is full to level h – 1 and has level h filled in from left to right
  - Balanced binary tree
    - A binary tree in which the left and right subtrees of any node have heights that differ by at most 1

The ADT Binary Tree: Basic Operations of the ADT Binary Tree

- The operations available for a particular ADT binary tree depend on the type of binary tree being implemented
- Basic operations of the ADT binary tree
  - `createBinaryTree()`
  - `createBinaryTree(rootItem)`
  - `makeEmpty()`
  - `isEmpty()`
  - `getRootItem() throws TreeException`

General Operations of the ADT Binary Tree

- General operations of the ADT binary tree
  - `createBinaryTree (rootItem, leftTree, rightTree)`
  - `setRootItem(newItem)`
  - `attachLeft(newItem) throws TreeException`
  - `attachRight(newItem) throws TreeException`
  - `attachLeftSubtree(leftTree) throws TreeException`
  - `attachRightSubtree(rightTree) throws TreeException`
  - `detachLeftSubtree() throws TreeException`
  - `detachRightSubtree() throws TreeException`

Traversals of a Binary Tree

- A traversal algorithm for a binary tree visits each node in the tree
- Recursive traversal algorithms
  - Preorder traversal
  - Inorder traversal
  - Postorder traversal
- Traversal is O(n)

Traversal of a Binary Tree

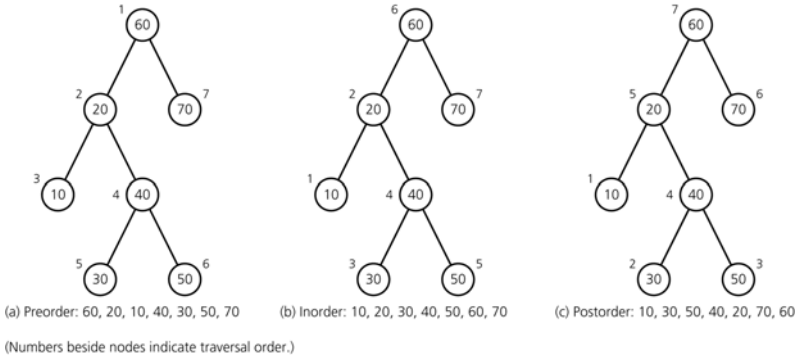


Figure 11-10

Traversals of a binary tree: a) preorder; b) inorder; c) postorder

Possible Representations of a Binary Tree

- An array-based representation
  - A Java class is used to define a node in the tree
  - A binary tree is represented by using an array of tree nodes
  - Each tree node contains a data portion and two indexes (one for each of the node's children)
  - Requires the creation of a free list which keeps track of available nodes

Possible Representations of a Binary Tree

(a)

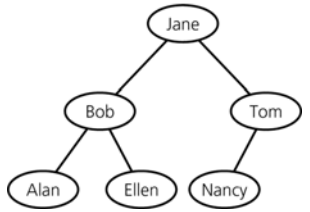


Figure 11-11a

a) A binary tree of names

(b) tree

	item	leftChild	rightChild	root
0	Jane	1	2	0
1	Bob	3	4	
2	Tom	5	–1	free
3	Alan	–1	–1	6
4	Ellen	–1	–1	
5	Nancy	–1	–1	
6	?	–1	7	
7	?	–1	8	
8	?	–1	9	
.	.	.	.	
.	.	.	.	
.	.	.	.	

Free list

Figure 11-11b

b) its array-based implementations

Possible Representations of a Binary Tree

- An array-based representation of a complete tree
  - If the binary tree is complete and remains complete
    - A memory-efficient array-based implementation can be used
- If a node is stored as element n of the array then its children are at n*2 + 1 and n*2 + 2

Possible Representations of a Binary Tree

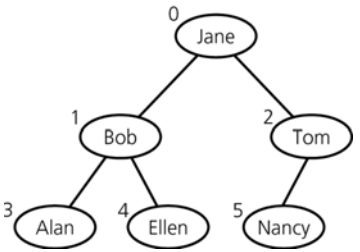


Figure 11-12

Level-by-level numbering of a complete binary tree

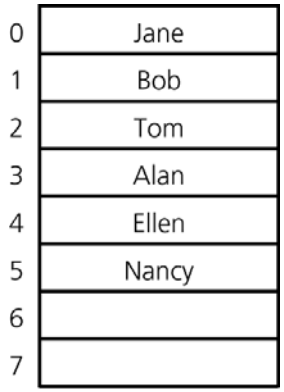


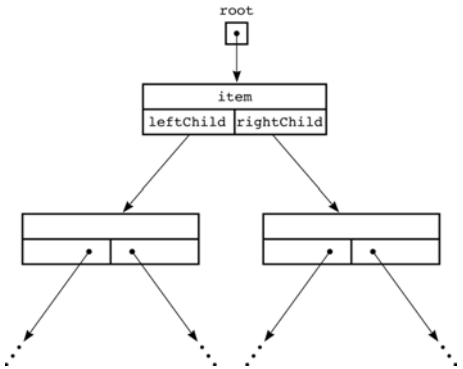
Figure 11-13

An array-based implementation of the complete binary tree in Figure 10-12

Possible Representations of a Binary Tree

- A reference-based representation
  - Java references can be used to link the nodes in the tree

Figure 11-14
A reference-based implementation of a binary tree



A Reference-Based Implementation of the ADT Binary Tree

- Classes that provide a reference-based implementation for the ADT binary tree
  - `TreeNode`
    - Represents a node in a binary tree
  - `TreeException`
    - An exception class
  - `BinaryTreeBasis`
    - An abstract class of basic tree operation
  - `BinaryTree`
    - Provides the general operations of a binary tree
    - Extends `BinaryTreeBasis`