Trees

Chapter 4

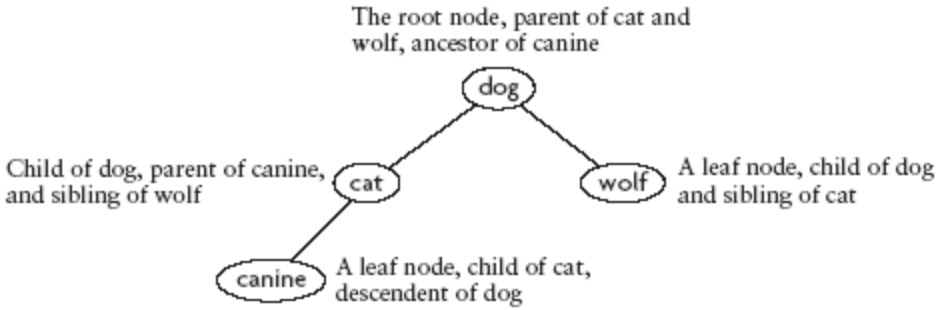
Tree Terminology

- A tree consists of a collection of elements or nodes, with each node linked to its successors
- The node at the top of a tree is called its root
- The links from a node to its successors are called branches
- The successors of a node are called its children
- The predecessor of a node is called its parent

Tree Terminology (continued)

- Each node in a tree has exactly one parent except for the root node, which has no parent
- Nodes that have the same parent are siblings
- A node that has no children is called a leaf node
- A generalization of the parent-child relationship is the ancestor-descendent relationship

FIGURE 8.2
A Tree of Words



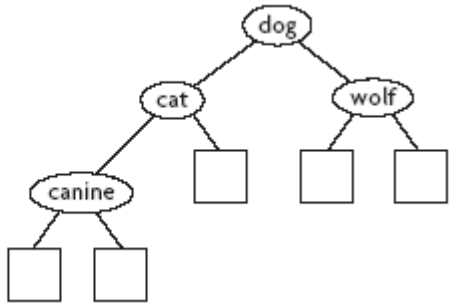
Tree Terminology (continued)

- A subtree of a node is a tree whose root is a child of that node
- The level of a node is a measure of its distance from the root

Binary Trees

- In a binary tree, each node has at most two subtrees
- A set of nodes T is a binary tree if either of the following is true
  - T is empty
  - Its root node has two subtrees, TL and TR, such that TL and TR are binary trees

FIGURE 8.3
A Tree of Words with Null Subtrees Indicated

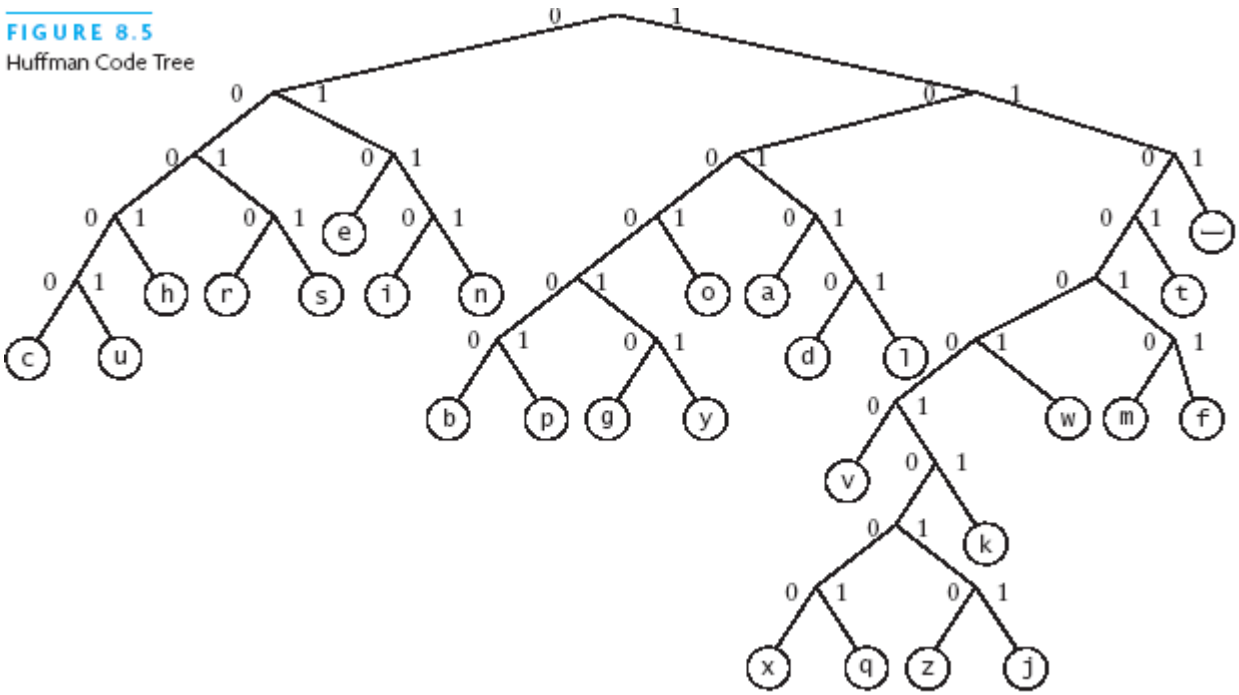


Some Types of Binary Trees

- Expression tree
  - Each node contains an operator or an operand
- Huffman tree
  - My book it’s page 389, We will cover that stuff later, this is just an introduction.
- Binary search trees
  - All elements in the left subtree precede those in the right subtree

Huffman Tree

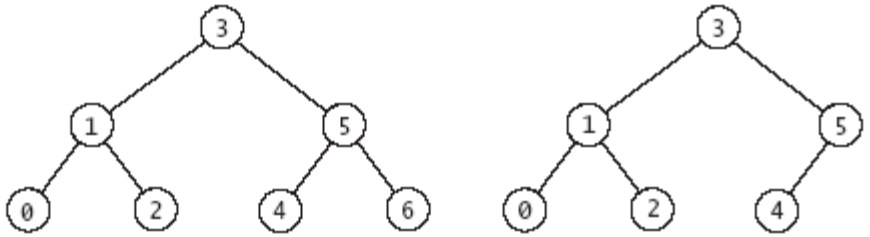
FIGURE 8.5
Huffman Code Tree



Fullness and Completeness

- Trees grow from the top down
- Each new value is inserted in a new leaf node
- A binary tree is full if every node has two children except for the leaves

FIGURE 8.6
Full Binary Tree (Left) and Complete Binary Tree (Right) of Height 3



Tree Traversals

- Often we want to determine the nodes of a tree and their relationship
  - Can do this by walking through the tree in a prescribed order and visiting the nodes as they are encountered
    - This process is called tree traversal
- Three kinds of tree traversal
  - Inorder
  - Preorder
  - Postorder

Tree Traversals (continued)

- Preorder: Visit root node, traverse TL, traverse TR
- Inorder: Traverse TL, visit root node, traverse TR
- Postorder: Traverse TL, Traverse TR, visit root node

Algorithm for Preorder Traversal

```
1.  if the tree is empty
2.      Return.
    else
3.      Visit the root.
4.      Preorder traverse the
        left subtree.
5.      Preorder traverse the
        right subtree.
```

Algorithm for Inorder Traversal

```
1.  if the tree is empty
2.      Return.
    else
3.      Inorder traverse the
        left subtree.
4.      Visit the root.
5.      Inorder traverse the
        right subtree.
```

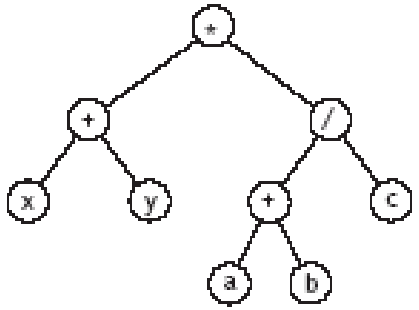
Algorithm for Postorder Traversal

```
1.  if the tree is empty
2.      Return.
    else
3.      Postorder traverse the
        left subtree.
4.      Postorder traverse the
        right subtree.
5.      Visit the root.
```

Traversals of Binary Search Trees and Expression Trees

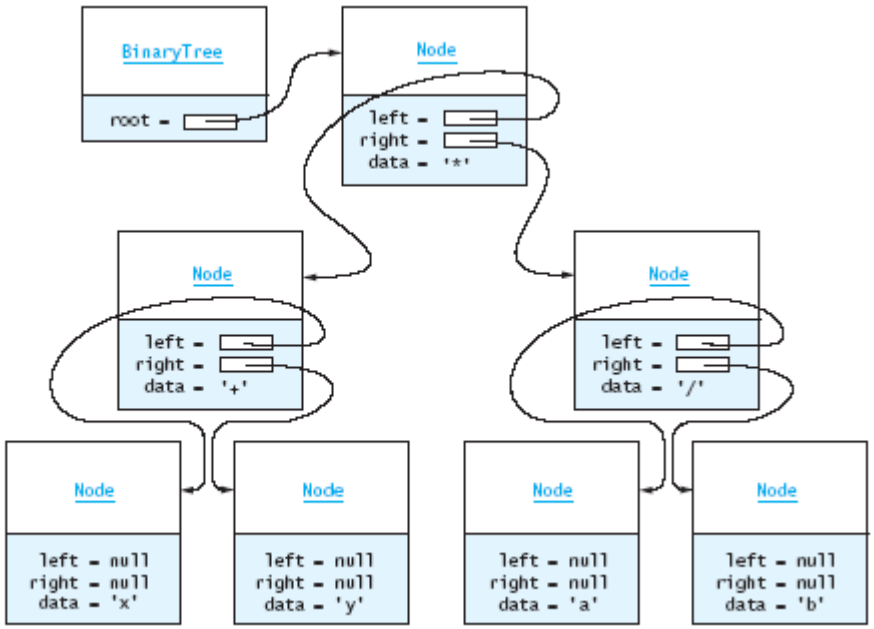
- An inorder traversal of a binary search tree results in the nodes being visited in sequence by increasing data value
- An inorder traversal of an expression tree inserts parenthesis where they belong (infix form)
- A postorder traversal of an expression tree results in postfix form

(x + y) * ((a + b) / c)



The BinaryTree<E> Class

FIGURE 8.11
Linked Representation of Expression Tree ((x + y) * (a / b))



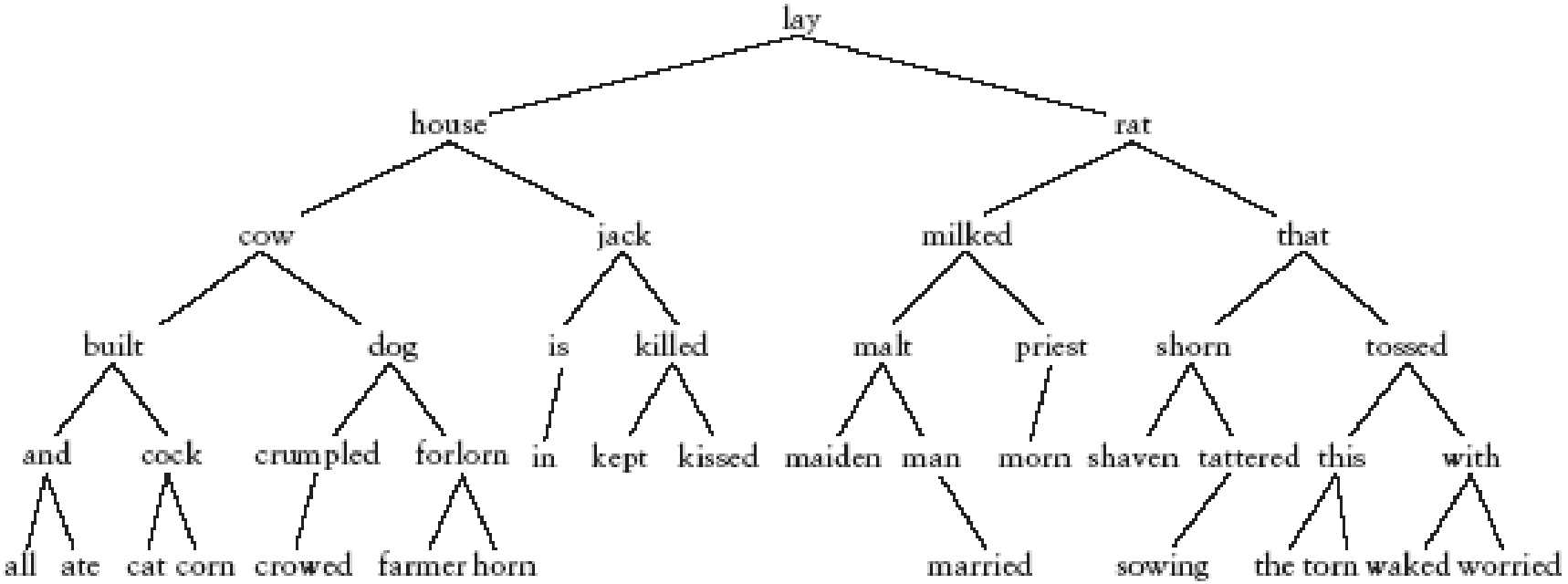
Overview of a Binary Search Tree

- Binary search tree definition
  - A set of nodes T is a binary search tree if either of the following is true
    - T is empty
    - Its root has two subtrees such that each is a binary search tree and the value in the root is greater than all values of the left subtree but less than all values in the right subtree

Overview of a Binary Search Tree (continued)

FIGURE 8.13

Binary Search Tree Containing All of the Words from "The House That Jack Built"



Searching a Binary Tree

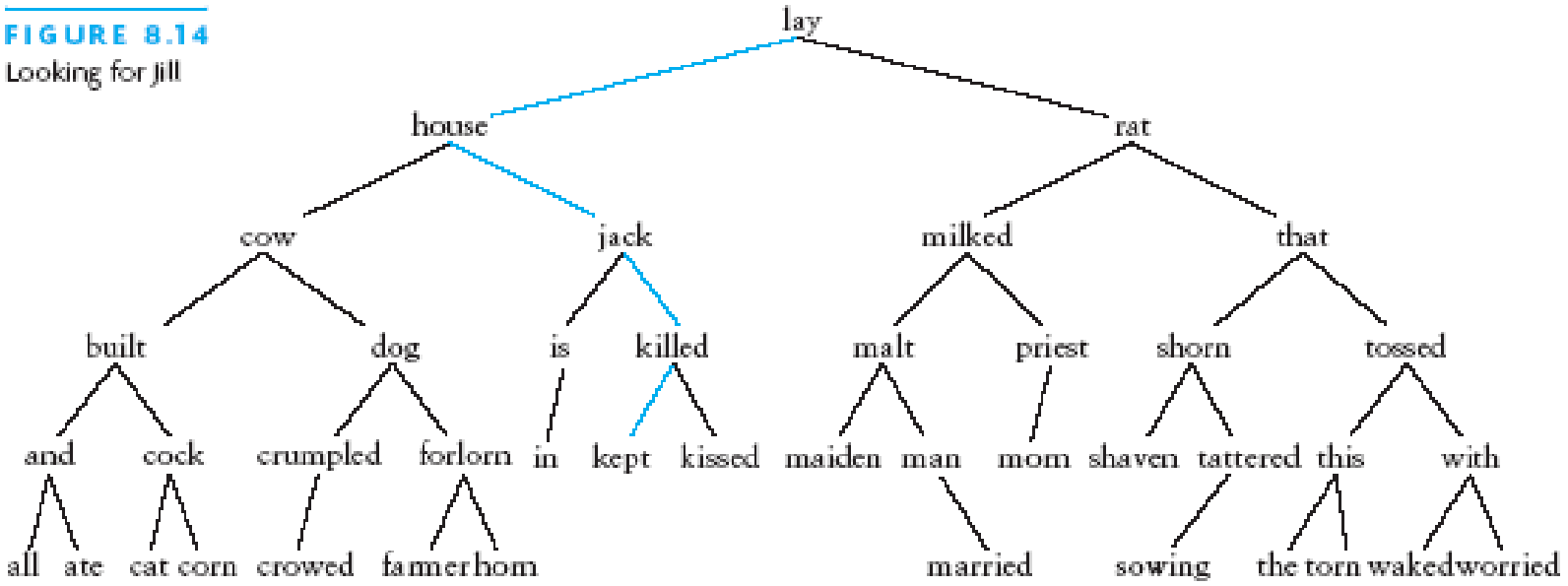


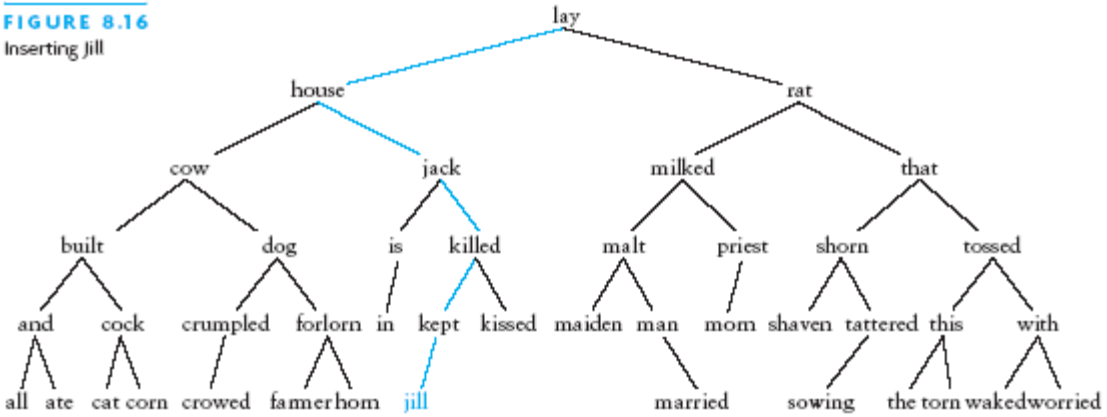
FIGURE 8.14
Looking for Jill

Insertion into a Binary Search Tree

Recursive Algorithm for Insertion in a Binary Search Tree

1. `if` the root is `null`
2. Replace empty tree with a new tree with the item at the root and return `true`.
3. `else if` the item is equal to `root.data`
4. The item is already in the tree; return `false`.
5. `else if` the item is less than `root.data`
6. Recursively insert the item in the left subtree.
7. `else`
8. Recursively insert the item in the right subtree.

FIGURE 8.16
Inserting Jill



Removing from a Binary Search Tree

Recursive Algorithm for Removal from a Binary Search Tree

```
1.  if the root is null
2.      The item is not in tree – return null.
3.  Compare the item to the data at the local root.
4.  if the item is less than the data at the local root
5.      Return the result of deleting from the left subtree.
6.  else if the item is greater than the local root
7.      Return the result of deleting from the right subtree.
8.  else // The item is in the local root
9.      Store the data in the local root in deletedReturn.
10.     if the local root has no children
11.         Set the parent of the local root to reference null.
12.     else if the local root has one child
13.         Set the parent of the local root to reference that child.
14.     else // Find the inorder predecessor
15.         if the left child has no right child it is the inorder predecessor
16.             Set the parent of the local root to reference the left child.
17.         else
18.             Find the rightmost node in the right subtree of the left
                child.
19.             Copy its data into the local root's data and remove it by
                setting its parent to reference its left child.
```

Removing from a Binary Search Tree (continued)