CHAPTER 10

Trees
Figure 10.1
A general tree
Figure 10.2
A subtree of the tree in Figure 10.1
Figure 10.3
a) An organization chart; b) a family tree

(a) President
   - VP Marketing
     - Director Media Relations
   - VP Manufacturing
     - Director Sales
   - VP Personnel

(b) Caroline
   - John
     - Joseph
   - Jacqueline
     - Rose
Figure 10.4
Binary trees that represent algebraic expressions

(a) \( a - b \)

(b) \( a - b / c \)

(c) \( (a - b) * c \)
Figure 10.5
A binary search tree of names
Figure 10.6
Binary trees with the same nodes but different heights

(a)  
(b)  
(c)
Figure 10.7

A full binary tree of height 3
Figure 10.8
A complete binary tree
Figure 10.9
Traversals of a binary tree: a) preorder; b) inorder; c) postorder

(a) Preorder: 60, 20, 10, 40, 30, 50, 70
(b) Inorder: 10, 20, 30, 40, 50, 60, 70
(c) Postorder: 10, 30, 50, 40, 20, 70, 60

(Numbers beside nodes indicate traversal order.)
Figure 10.10a

a) A binary tree of names

(a)

Jane

Bob

Tom

Alan

Ellen

Nancy
**Figure 10.10b**

b) its array-based implementations

<table>
<thead>
<tr>
<th>b) item</th>
<th>leftChild</th>
<th>rightChild</th>
<th>root</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Jane</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>Bob</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Tom</td>
<td>5</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>Alan</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>4</td>
<td>Ellen</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>5</td>
<td>Nancy</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>6</td>
<td>?</td>
<td>-1</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>?</td>
<td>-1</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>?</td>
<td>-1</td>
<td>9</td>
</tr>
</tbody>
</table>

Free list:

```
0
'''

Free

```
Figure 10.11
Level-by-level numbering of a complete binary tree
Figure 10.12
An array-based implementation of the complete binary tree in Figure 10-11

```
  0   Jane
  1   Bob
  2   Tom
  3   Alan
  4   Ellen
  5   Nancy
  6
  7
```
Figure 10.13
A reference-based implementation of a binary tree
Figure 10.14
Contents of the implicit stack as `treeNode` progresses through a given tree during a recursive inorder traversal

(The notation →60 means "a reference to the node containing 60.")

Stack:

<table>
<thead>
<tr>
<th>Step</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Visit</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| TreeNode at Step 1
| TreeNode at Steps 2, 9, and 10
| TreeNode at Steps 3, 5, 6, and 8
| TreeNode is null at Steps 4 and 7
Figure 10.15
Traversing a) the left and b) the right subtrees of 20

Left subtree of 20 has been traversed. Pop reference to 10 from stack, visit 20.

Right subtree of 20 has been traversed. Pop reference to 40 from stack.
Figure 10.16
Avoiding returns to nodes B and C
Figure 10.17
A binary search tree
Figure 10.18a
Binary search trees with the same data as in Figure 10-17
Figure 10.18b
Binary search trees with the same data as in Figure 10-17

(b)
Figure 10.18c
Binary search trees with the same data as in Figure 10-17
**Figure 10.19**
An array of names in sorted order

<table>
<thead>
<tr>
<th>Alan</th>
<th>Bob</th>
<th>Ellen</th>
<th>Jane</th>
<th>Nancy</th>
<th>Tom</th>
<th>Wendy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>
Figure 10.20
Empty subtree where `search` terminates
**Figure 10.21a and 10.21b**

a) Insertion into an empty tree; b) search terminates at a leaf
Figure 10.21c

c) insertion at a leaf
Figure 10.22

a) $N$ with only a left child—$N$ can be either the left or right child of $P$; b) after deleting node $N$
Figure 10.23

N with two children
Figure 10.24
Not any node will do
Figure 10.25
Search key $x$ can be replaced by $y$
Figure 10.26
Copying the item whose search key is the inorder successor of N’s search key

![Diagram showing a tree with nodes labeled Nancy, Jane, Bob, Ellen, Tom, Alan, and Wendy. The node labeled Nancy is highlighted, and the path from N to Nancy is indicated by an arrow.](image-url)
Figure 10.27
Recursive deletion of node $N$

Any change to TreeNode while deleting node $N$ (Bob) changes leftChild of Jane
Figure 10.28
A maximum-height binary tree with seven nodes
Figure 10.29
Binary trees of height 3
Figure 10.30
Counting the nodes in a full binary tree of height $h$

<table>
<thead>
<tr>
<th>Level</th>
<th>Number of nodes at this level</th>
<th>Number of nodes at this and previous levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$1 = 2^0$</td>
<td>$1 = 2^1 - 1$</td>
</tr>
<tr>
<td>2</td>
<td>$2 = 2^1$</td>
<td>$3 = 2^2 - 1$</td>
</tr>
<tr>
<td>3</td>
<td>$4 = 2^2$</td>
<td>$7 = 2^3 - 1$</td>
</tr>
<tr>
<td>4</td>
<td>$8 = 2^3$</td>
<td>$15 = 2^4 - 1$</td>
</tr>
<tr>
<td>$h$</td>
<td>$2^{h-1}$</td>
<td>$2^h - 1$</td>
</tr>
</tbody>
</table>
Figure 10.31
Filling in the last level of a tree
### Figure 10.32
The order of the retrieval, insertion, deletion, and traversal operations for the reference-based implementation of the ADT binary search tree

<table>
<thead>
<tr>
<th>Operation</th>
<th>Average case</th>
<th>Worst case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieval</td>
<td>$O(\log n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Insertion</td>
<td>$O(\log n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Deletion</td>
<td>$O(\log n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Traversal</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
</tbody>
</table>
Figure 10.33
a) A binary search tree $bst$; b) the sequence of insertions that result in this tree

(b) $bst$.insert(60); $bst$.insert(20); $bst$.insert(10); $bst$.insert(40); $bst$.insert(30); $bst$.insert(50); $bst$.insert(70);
Figure 10.34
A full tree saved in a file by using inorder traversal
Figure 10.35
A tree of minimum height that is not complete
Figure 10.36
A general tree
Figure 10.37
A reference-based implementation of the general tree in Figure 10.36
Figure 10.38
The binary tree that Figure 10-37 represents
Figure 10.39
An implementation of the $n$-ary tree in Figure 10.36
Figure 10.40
A tree for Self-Test Exercises 1, 3, 7, and 11 and for Exercises 6 and 11
Figure 10.41
An array for Self-Test Exercise 9

\[
\begin{array}{cccccccccc}
5 & 1 & 2 & 8 & 6 & 10 & 3 & 9 & 4 & 7 \\
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\end{array}
\]
Figure 10.42
A tree for Self-Test Exercise 10 and for Exercise 2a
Figure 10.43
A binary search tree for Exercise 3
Figure 10.44
A minimax tree for Exercise 17