Trees
Chapter 23
Motivation

- Tree is a fundamental data-structure
- Operating Systems: Directory Structure
- Artificial Intelligence: Game Playing Programs
- Programming Languages: Inheritance
- Data Structure: Heap (for Priority Queue)
- Algorithms: Binary Search Tree
- Compilers: Parse tree, expression tree
- Machine Learning: Decision Trees
- Database Management: B Tree & B+ Tree indexes
  - Ex. Is B+Tree really a tree?
## Contents

- Motivation
- Tree Concepts
  - Hierarchical Organizations
  - Tree Terminology

<table>
<thead>
<tr>
<th></th>
<th>Binary Tree (B. T.)</th>
<th>General Tree (G. T.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traversal</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java Interface</td>
<td>B. T. API</td>
<td>G. T. API</td>
</tr>
<tr>
<td>Examples</td>
<td>Heap, Binary Search Trees Decision Trees, Expression Trees</td>
<td>Game Tree, Parse Tree</td>
</tr>
</tbody>
</table>
Objectives

• Describe binary trees and general trees
  ▪ using standard terminology

• Traverse tree in one of four ways:
  ▪ preorder
  ▪ postorder
  ▪ inorder
  ▪ level order

• Give examples:
  ▪ Binary Trees: expression trees, decision trees, binary search trees, and heaps
  ▪ General Trees: parse trees, game trees
Tree Concepts

• A way to organize data
  ▪ Consider a family tree
• Hierarchical organization
  ▪ Data items have ancestors, descendants
  ▪ Data items appear at various levels
• Contrast with previous linearly organized structures

Figure 23-1 Carole’s children and grandchildren
Figure 23-2 Jared’s parents and grandparents
Figure 23-4 Computer files organized into folders

Figure 23-5 A tree equivalent to the tree in Figure 23-4
Tree Concepts: Root, Parent, Children

- *Root* of an ADT tree is at tree’s top
  - Only node with no parent
  - All other nodes have one parent each
- Each node can have *children*
  - A node with children is a *parent*
  - A node without children is a *leaf*
- **Ex.** Identify (a.) leafs, (b.) siblings of node K, (c.) children of node B.

![Diagram of a tree with labeled nodes and levels](image.png)

Figure 23-5 A tree equivalent to the tree in Figure 23-4

Copyright ©2012 by Pearson Education, Inc. All rights reserved
Tree Concepts: Subtree, Height, Path

- Node and its descendants form a *subtree*
- Subtree of a node
  - Tree rooted at a child of that node
  - Subtree of the tree’s root
- Height of a tree
  - Number of levels in the tree
- **Ex.** Identify (a.) children of node B, (b.) descendants of node B, (c.) ancestors of node N, (d.) all parent nodes, (e.) height of subtrees rooted at node B and node C.

Figure 23-5 A tree equivalent to the tree in Figure 23-4
Tree vs. Non-Tree

• Root and its descendants form a tree
• Path between a tree’s root and any other node is unique.

Ex. Which of the following are not trees? Why?

(a.) B+ Tree
(b.) Java Collections
(c.) Search Tree

(d.) Figure 23-3 A portion of a university’s administrative structure
Tree Concepts: Binary Tree

- **Classify Trees by number of children**
  - General tree: Node can any number of children
  - $N$-ary tree: Node has max $n$ children
  - Binary tree: node has max 2 children
    - Full Binary Tree, Complete Binary Tree, Other Binary Trees

---

**Figure 23-6 Three binary trees**

- (a) Full tree
- (b) Complete tree
- (c) Tree that is not full and not complete

Left children: B, D, F
Right children: C, E, G
The height of a binary tree with \( n \) nodes that is either complete or full is \( \log_2 (n + 1) \) rounded up.

Q. 6. How many are in a full binary tree of height 6?
Traversals of a BinaryTree

- Must visit/process each data item exactly once
- Nodes can be visited in different orders
- For a binary tree
  - Visit root, Visit all nodes in root’s left subtree, Visit all nodes in right subtree
- Could visit root before, between, or after subtrees
  - Level-order: F, B, G, A, D, I, C, E, H
  - Inorder: A, B, C, D, E, F, G, H, I
  - Postorder: A, C, E, D, B, H, I, G, F
- Alternate Categorization:
  - breadth-first traversal, e.g., level-order
  - depth-first traversal, e.g., pre-order, inorder, postorder
- Q. 7. List output of different traversal on Fig. 23-13.
  - Level-order
  - Preorder
  - Inorder
  - Postorder

Figure 23-13 A binary tree
Figure 23-8 **preorder** traversal: Visit root *before* visiting root’s subtrees

Figure 23-9: **inorder**: Visit root *between* visiting root’s subtrees

Figure 23-10 **postorder**: Visit root *after* visiting root’s subtrees

Figure 23-11 **level-order**: Begins at root, visits nodes *by level*
Traversals of a General Tree

- For a general tree (not a binary):
  - In-order traversal not well defined
  - Can do level-order, pre-order, post-order
- **Exercise:** Number the nodes of a general tree in Figure 23-12 for level-order traversal.

![Figure 23-12](image_url)

**FIGURE 23-12** Visitation order of traversals of a general tree: (a) preorder; (b) postorder
Java Interfaces for Trees & Binary Trees

// Listings 23-1, 23-2, 23-3
package TreePackage;
public interface TreeInterface < T > {
    public T getRootData ();
    public int getHeight ();
    public int getNumberOfNodes ();
    public boolean isEmpty ();
    public void clear ();
}
import java.util.Iterator;
public interface TreeIteratorInterface < T > {
    public Iterator < T > getPreorderIterator ();
    public Iterator < T > getPostorderIterator ();
    public Iterator < T > getInorderIterator ();
    public Iterator < T > getLevelOrderIterator ();
}
public interface BinaryTreeInterface < T > extends TreeInterface < T >, TreeIteratorInterface < T > {
    public void setTree (T rootData);
    public void setTree (T rootData,
                           BinaryTreeInterface < T > leftTree,
                           BinaryTreeInterface < T > rightTree);
}
Exercise: Which traversal strategy is used by minimax?
- Preorder
- Inorder
- Postorder
- Level-wise
Use Case 1: Heaps

- Complete binary tree: nodes contain **Comparable** objects
- Organization
  - Each node contains object no smaller (or no larger) than objects in descendants
  - Maxheap, object in node greater than or equal to descendant objects
  - Minheap, object in node less than or equal to descendant objects
- **Ex.** Discuss tree operations to implement `getMax()` and `removeMax()` in maxheap.
- **Leading Question:** Which Abstract Data Type may be implemented by Heap?

![Maxheap and Minheap Diagram](image)

**FIGURE 23-20** (a) A maxheap and (b) a minheap that contain the same values
U. C. 1: Use Heaps to implement Priority Queues

//Listing 23-6, 23-7
public interface MaxHeapInterface<T extends Comparable<? super T>> {
    public void add(T newEntry);
    public T removeMax();
    public T getMax();
    public boolean isEmpty();
    public int getSize();
    public void clear();
}

// Assume class MaxHeap implements MaxHeapInterface
public class PriorityQueue<T extends Comparable<? super T>> implements PriorityQueueInterface<T> {
    private MaxHeapInterface<T> pq;
    public PriorityQueue() { pq = new MaxHeap<T>();}
    public void add(T newEntry) { pq.add(newEntry); }
    //Implement remove, peek, isEmpty, getSize, and clear here.
}

Ex. Discuss heap operations to implement remove() and peek() for Priority Queues.
Use Case 2: Binary Search Trees

• Nodes contain **Comparable** objects
• For each node in a search tree:
  - Node’s data greater than all data in node’s left subtree
  - Node’s data less than all data in node’s right subtree
Use Case 3: Expression Trees

- Use binary tree to represent expressions
  - Two operands
  - One binary operator
  - The operator is the root
- Can be used to evaluate an expression
  - Post order traversal
  - Each operand, then the operator
- Q. 9, 10, 11 (pp. 583)

FIGURE 23-14 Expression trees for four algebraic expressions
Use Case for General Trees

• Parse trees
  ▪ Use grammar rules for algebraic expression
  ▪ Apply to elements of a string
  ▪ Expression is root
  ▪ Variables, operators are the leaves

• Must be general
  ▪ To accommodate any expression

• Q. 18 (pp. 593)
Use Case 4: Decision Trees

- Used for expert systems
  - Helps users solve problems
  - Parent node asks question
  - Child nodes provide conclusion or further question
- Decision trees are generally n-ary
  - Expert system application often binary
- Note interface for decision tree, [Listing 23-4](#)

![Decision Tree Diagram](image)

Figure 23-15 A portion of a binary decision tree
Use Case 2: Decision Trees & Guessing Game

• Consider a guessing game
  ▪ Program asks yes/no questions
  ▪ Adds to its own decision tree as game progresses

• Example
• View class GuessingGame, Listing 23-5

Figure 23-16 An initial decision tree for a guessing game

Figure 23-17 The decision tree for a guessing game after acquiring another fact
End

Chapter 23