

# Tree and Binary Tree

## Basic Tree Concepts

- A family tree shows the descendants of a common ancestor.



Courtesy of iStockphoto.

- In computer science, a **tree** is a hierarchical data structure composed of nodes.
- Each **node** has a sequence of child nodes.
- The **root** is the node with no parent.
- A **leaf** is a node with no children.

## Basic Tree Concepts

- British royal family tree

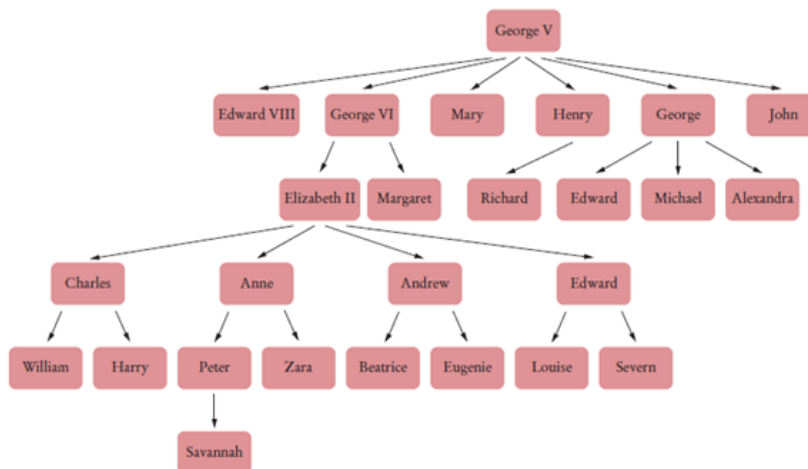
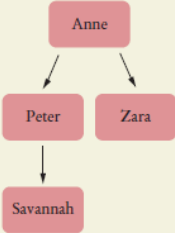
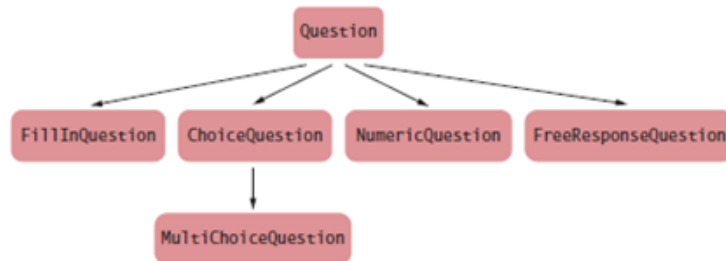
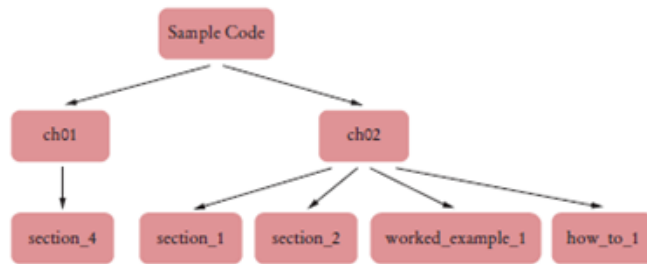


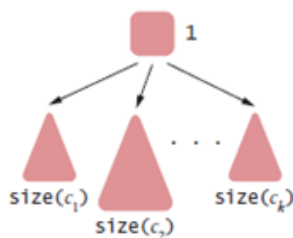
Table 1 Tree Terminology		
Term	Definition	Example (using Figure 1)
Node	The building block of a tree: A tree is composed of linked nodes.	This tree has 26 nodes: George V, Edward VIII, ..., Savannah.
Child	Each node has, by definition, a sequence of links to other nodes called its child nodes.	The children of Elizabeth II are Charles, Anne, Andrew, and Edward.
Leaf	A node with no child nodes.	This tree has 16 leaves, including William, Harry, and Savannah.
Interior node	A node that is not a leaf.	George V or George VI, but not Mary.
Parent	If the node $c$ is a child of the node $p$ , then $p$ is a parent of $c$ .	Elizabeth II is the parent of Charles.
Sibling	If the node $p$ has children $c$ and $d$ , then these nodes are siblings.	Charles and Anne are siblings.
Root	The node with no parent. By definition, each tree has one root node.	George V.
Path	A sequence of nodes $c_1, c_2, \dots, c_k$ where $c_{i+1}$ is a child of $c_i$ .	Elizabeth II, Anne, Peter, Savannah is a path of length 4.
Descendant	$d$ is a descendant of $c$ if there is a path from $c$ to $d$ .	Peter is a descendant of Elizabeth II but not of Henry.
Ancestor	$c$ is an ancestor of $d$ if $d$ is a descendant of $c$ .	Elizabeth II is an ancestor of Peter, but Henry is not.
Subtree	The subtree rooted at node $n$ is the tree formed by taking $n$ as the root node and including all its descendants.	<p>The subtree with root Anne is</p>  <pre> graph TD     Anne[Anne] --&gt; Peter[Peter]     Anne --&gt; Zara[Zara]     Peter --&gt; Savannah[Savannah] </pre>
Height	The number of nodes in the longest path from the root to a leaf. (Some authors define the height to be the number of edges in the longest path, which is one less than the height used in this book.)	This tree has height 6. The longest path is George V, George VI, Elizabeth II, Anne, Peter, Savannah.

# Trees in Computer Science



## Basic Tree Concepts

- A tree class holds a reference to a root node. Each node holds:
  - A data item
  - A list of references to the child nodes
- When computing tree properties, it is common to recursively visit smaller and smaller subtrees.
- Size of tree with root  $r$  whose children are  $c_1 \dots c_k$ 
  - $\text{size}(r) = 1 + \text{size}(c_1) + \dots + \text{size}(c_k)$



The size method in the Tree class:

```
public class Tree
{
    public int size()
    {
        if (root == null) { return 0; }
        else { return root.size(); }
    }
}
```

Recursive helper method in the Node class:

```
class Node
{
    public int size()
    {
        int sum = 0;
        for (Node child : children)
        {
            sum = sum + child.size();
        }
        return 1 + sum;
    }
}
```

## Binary Tree

In a binary tree, each node has a left and a right child node.

Example

- In a decision tree:
  - Each non-leaf node contains a question
  - The left subtree corresponds to a “yes” answer
  - The right subtree to a “no” answer
  - Every node has either two children or no children

## Binary Tree Examples - Decision Tree

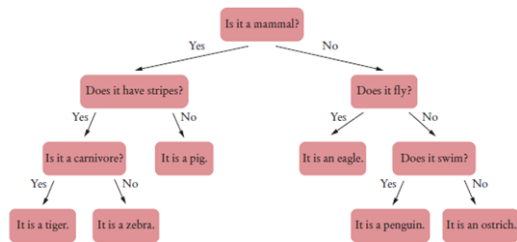


Figure 4 A Decision Tree for an Animal Guessing Game

In a balanced binary tree, each subtree has **approximately** the same number of nodes.

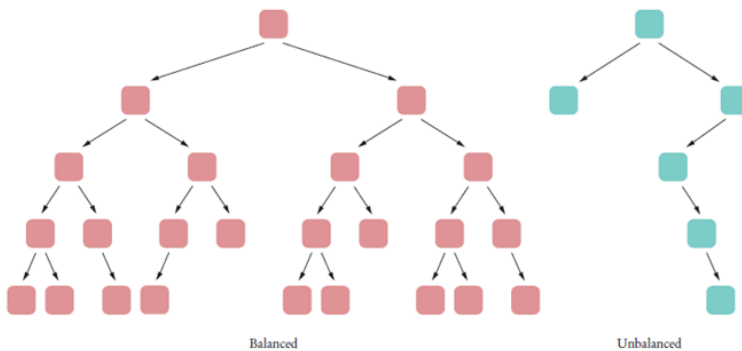


Figure 7 Balanced and Unbalanced Trees

- A binary tree of height  $h$  can have up to  $n = 2^h - 1$  nodes.
- A completely filled binary tree of height 4 has  $1 + 2 + 4 + 8 = 15 = 2^4 - 1$  nodes.

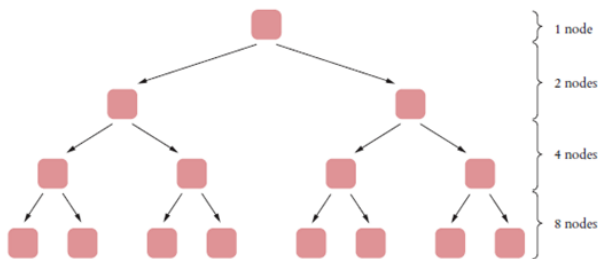


Figure 8 A Completely Filled Binary Tree of Height 4

## Balanced Trees

- For a completely filled binary tree:  $h = \log_2(n + 1)$  For a balanced tree:  $h \approx \log_2(n)$
- Example: the height of a balanced binary tree with 1,000 nodes Approximately 10 (because  $1000 \approx 1024 = 2^{10}$ ).
- Example: the height of a balanced binary tree with 1,000,000 nodes approximately 20 (because  $10^6 \approx 2^{20}$ )
- You can find any element in this tree in about 20 steps

## A Binary Tree Implementation

- Binary tree node has a reference:
  - to a right child
  - to a left child
  - either can be null
- **Leaf:** node in which both children are null.

## A Binary Tree Implementation

- **BinaryTree** class:

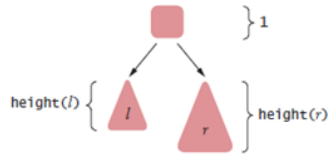
```
public class BinaryTree
{
    private Node root;

    public BinaryTree() { root = null; } // An empty tree

    public BinaryTree(Object rootData, BinaryTree left, BinaryTree right)
    {
        root = new Node();
        root.data = rootData;
        root.left = left.root;
        root.right = right.root;
    }

    class Node
    {
        public Object data;
        public Node left;
        public Node right;
    }
    . . .
}
```

- To find the height of a binary tree  $t$  with left and right children  $l$  and  $r$ 
  - Take the maximum height of the two children and add 1
  - $\text{height}(t) = 1 + \max(\text{height}(l), \text{height}(r))$



- Make a static recursive helper method `height` in the

`Tree` class:

```
public class BinaryTree
{
    private static int height(Node n)
    {
        if (n == null) { return 0; }
        else { return 1 + Math.max(height(n.left), height(n.right)); }
    }
    . . .
}
```

Provide a public `height` method in the `Tree` class:

```
public class BinaryTree
{
    public int height() { return height(root); }
}
```