Chapter 7: Trees

Outline

1. Chapter 7: Trees
   - Tree Terminology
   - Example: Expression Trees
   - Binary Tree Representations
   - In-Class Work
Chapter 7: Trees

Uses of Trees

Taxonomies

- Animal
  - Vertebrate
    - Fish
    - Mammal
    - Reptile
  - Invertebrate
    - Insect
Hierarchies

Board Of Directors

CEO

Administrative Staff...

COO

Division Executives

Managers...
USES OF TREES

EFFICIENT CONTAINERS OF SEQUENTIAL DATA

Each node represents a single data item of a collection organized, for efficient access, into a tree:

- Binary Search Trees
- Heaps
- Priority Queues
- B-Trees, Quad Trees, etc.
Definitions

Definitions I

- A tree consists of nodes connected by edges.
- A node can have zero or more child nodes. A child is connected to its parent node by a single edge.
- Children with the same parent are called siblings.
- A tree with no nodes or edges is an empty tree.
- A nonempty tree will have one special node with no parent called the root node.
- A node with no children is called a leaf.
Definitions

All nodes are connected to the root by a path of edges.

The depth of a node is the length of its path to the root. The root has depth zero.

A level of a tree is a set of nodes which have the same depth.

The descendants of a node are nodes whose paths include that node.

The ancestors of a node are nodes on its path to the root.
A tree whose nodes have at most two children is a **binary tree**.
A full binary tree is one whose levels have every position filled.
A full binary tree is one whose levels have every position filled.

Question: how many nodes are in a full binary tree of height $h$?
A full binary tree is one whose levels have every position filled.

Question: how many nodes are in a full binary tree of height $h$? $2^h - 1$
Full Binary Trees

A full binary tree is one whose levels have every position filled.

Question: how many leaves are in the full binary tree of height $h$?
A full binary tree is one whose levels have every position filled.

Question: how many leaves are in the full binary tree of height $h$?

$2^{h-1}$
A full binary tree is one whose levels have every position filled.

Question: If there are $n$ nodes in a full binary tree, how many leaves are there?
A full binary tree is one whose levels have every position filled.

Question: If there are $n$ nodes in a full binary tree, how many leaves are there? $\left\lceil \frac{n}{2} \right\rceil$
A **complete** binary tree has every level filled except the bottom, which is filled from left to right.
Binary Tree Traversal

Binary Trees are Recursive Data Structures

A binary tree can be defined as either

- An empty binary tree (base case), or
- A node having two binary trees as attributes, a left subtree and a right subtree (recursive case).

Using this recursive definition, recursive algorithms can be used to process the nodes of a binary tree.
**Binary Tree Traversal**

**Preorder**

```python
def traverse(tree):
    if tree is not empty:
        process data at tree’s root
        traverse(tree’s left subtree)
        traverse(tree’s right subtree)
```
**Binary Tree Traversal**

**Inorder**

```python
def traverse(tree):
    if tree is not empty:
        traverse(tree’s left subtree)
        process data at tree’s root
        traverse(tree’s right subtree)
```
**Binary Tree Traversal**

**Postorder**

```python
def traverse(tree):
    if tree is not empty:
        traverse(tree’s left subtree)
        traverse(tree’s right subtree)
        process data at tree’s root
```
Binary Tree Representation Of An Expression

Abstract Syntax Tree

```
      +
     /|
    / +
   /  4
  /   5
 /    6
```

Example: Expression Trees

In-Class Work
Binary Tree Representation of an Expression

Different Traversals Give the Different Notations

- Printing in preorder gives Prefix Notation.
- Printing inorder gives Infix Notation.
- Printing in Postorder gives Postfix Notation.
Binary Tree Representation of an Expression

Different traversals give the different notations:

- Printing in preorder gives Prefix Notation. \( +*+234*56 \)
- Printing inorder gives Infix Notation. \( (2+3)*4+5*6 \)
- Printing in Postorder gives Postfix Notation. \( 23+4*56*+ \)
Chapter 7: Trees

Example: Expression Trees

Binary Tree Representation of an Expression

**Evaluation**

```
def evaluateTree(tree):
    if tree’s root is an operand:
        return root data
    else:  # root contains an operator
        leftValue = evaluateTree(tree’s left subtree)
        rightValue = evaluateTree(tree’s right subtree)
        result = operator(leftValue, rightValue)
        return result
```
A TreeNode Class in Python

- An empty tree is represented by None.
- A nonempty tree is defined using a TreeNode as root.
- The TreeNode class is defined recursively.
Recursive Definition Of A TreeNode Class

```python
class TreeNode:
    def __init__(self, data = None, left=None, right=None):
        self.item = data
        self.left = left # TreeNode or None
        self.right = right # TreeNode or None
```

Python Examples Using The TreeNode Class

```
TreeNode

left item right

None
```

linked representation of a binary tree
Linked Representation Of A Binary Tree

Example 1: Abstract View

1
2
3
Example 1: Abstract View

```
1
2
3
```

Example 1: Python Implementation

```python
left = TreeNode(1)
right = TreeNode(3)
root = TreeNode(2, left, right)
```
Example 1: Python Memory Model

```
  left  item  right
    ●      2      ●

  left  item  right
    None    1    None

  left  item  right
    None    3    None
```
**Example 2: Abstract View**

```
   2
  / \
7   6
 / \  / \
5   8 4
/   / \ \
3   8   4
```
Example 2: Abstract View

```
    2
   / \
  7   6
 /     /
5   8   4
```

Example 2: Python Implementation

```python
root = TreeNode(2,
    TreeNode(7,
        None,
        TreeNode(5, TreeNode(3))),
    TreeNode(6, TreeNode(8), TreeNode(4)))
```
Linked Representation of a Binary Tree

Example 2: Python Memory Model

```
left  item  right
None  7    

left  item  right
None  5    None

left  item  right
None  8    None

left  item  right
None  4    None

left  item  right
None  6    

left  item  right
2
```
Array Representation of Binary Trees

Abstract View - Array Representation

```
<table>
<thead>
<tr>
<th>2</th>
<th>7</th>
<th>6</th>
<th>None</th>
<th>5</th>
<th>8</th>
<th>4</th>
<th>None</th>
<th>None</th>
<th>3</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>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>
```
Array Representation of Binary Trees

The node at position \( i \) has:

```python
def left_child(i):
    return 2 * i + 1

def right_child(i):
    return 2 * i + 2

def parent(i):
    return (i - 1) // 2
```
See the handout.