CSC316 Trees
Chapter 6

Topics covered

- Trees
  - definitions and terminology
- Binary trees.
  - Definitions and property
- Tree representation.
- Tree Traversal.

Tree Definitions

- A finite set of one or more nodes s.t., one node is root, and the remaining nodes are partitioned into $k$ disjoint sets $T_1, T_2, \ldots, T_k$ where each of these sets is a tree.
- A hierarchical data structure in which each component (except one) has one immediate predecessor, but can have many immediate successors.

Terminology

- $A$ is the root node
- $B$ is the parent of $D$ and $E$
- $C$ is the sibling of $B$
- $D$ and $E$ are the children of $B$
- $D, E, F, G, and I$ are leaves
- $A, B, C, and I$ are internal nodes
- The depth (level) of $E$ is 2
- The height of the tree is 3
- The degree of node $B$ is 2

Tree Example

Property

- # edges = # nodes - 1
Binary Tree

- Ordered tree: the children of each node are ordered.
- Binary tree: ordered tree with all internal nodes of degree at most 2.

Note the difference with the definition in the book.

Example of Binary tree

(((3 x (1 + (4 + 6))) + (2 + 8)) x 5) + (4 x (7 + 2)))

The nesting level of a node decides the level of a node.

Binary tree traversal

- Printing all the nodes by visiting each node exactly once.
  - In-order (LVR)
  - Pre-order (VLR)
  - Post-order (LRV)

Inorder Traversal

- In an inorder traversal a node is visited after its left subtree and before its right subtree

Algorithm

\[
inOrder(v) \\
\begin{cases} 
\text{visit}(v) \\
\text{if isInternal}(v) \text{ then } \text{inOrder}(\text{leftChild}(v)) \\
\text{if isInternal}(v) \text{ then } \text{inOrder}(\text{rightChild}(v)) 
\end{cases}
\]

Preorder Traversal

- In a preorder traversal a node is visited before its left subtree and its right subtree

Algorithm

\[
\text{PreOrder}(v) \\
\begin{cases} 
\text{if hasLeftChild}(v) \text{ then } \text{PreOrder}(\text{leftChild}(v)) \\
\text{if hasRightChild}(v) \text{ then } \text{PreOrder}(\text{rightChild}(v)) \\
\text{visit}(v) 
\end{cases}
\]

Postorder Traversal

- In a postorder traversal a node is visited after its left subtree and its right subtree

Algorithm

\[
\text{PostOrder}(v) \\
\begin{cases} 
\text{if hasLeftChild}(v) \text{ then } \text{PostOrder}(\text{leftChild}(v)) \\
\text{if hasRightChild}(v) \text{ then } \text{PostOrder}(\text{rightChild}(v)) \\
\text{visit}(v) 
\end{cases}
\]
Properties of Binary Trees

• The maximum number of nodes on depth \( i \) of a binary tree is \( 2^i \).
• The maximum number of nodes in a binary tree of depth \( k \) is \( 2^{k+1} - 1 \).
• Proof by induction.

Properties of Binary Trees

• For any non-empty binary tree \( T \), if \( n_0 \) is the number of leaf nodes and \( n_2 \) the number of nodes of degree 2, then \( n_0 = n_2 + 1 \).
• Proof.

More terminology

• Full binary tree of height \( k \).
  – Binary tree of depth \( k \) having \( 2^{k+1} - 1 \) nodes.
  – Only nodes at depth \( k \) are leaves, and all the internal nodes have two children.
  – You can number nodes in the tree
    • root 1,
    • nodes at any level are numbered from left to right
    • parent of node \( i \) is numbered \( \text{floor}(i/2) \).
    • Left child of \( i \) is numbered \( 2i \).
    • Right child of \( i \) is number of \( 2i + 1 \).

More terminology

• A complete binary tree of height \( k \) with \( n \) nodes.
  – Its nodes correspond to the nodes numbered one to \( n \) in the full binary tree of depth \( k \).

More property

• The height of a full tree with node \( n \) is \( \log_2 (n+1) - 1 \)
• The height of a complete tree with \( n \) nodes is
  \( \log_2 (n+1) - 1 \leq h < \log_2 (n+1) \)

Tree ADT

• We use positions to abstract nodes
• Generic methods:
  – integer size()
  – boolean isEmpty()
• Accessor methods:
  – node root()
  – node parent(p)
• Query methods:
  – boolean isInternal(p)
  – boolean isExternal(p)
  – boolean isRoot(p)
• Update methods:
  – swapElements(p, q)
  – object replaceElement(p, o)
• Additional update methods may be defined by data structures implementing the Tree ADT
Tree data structure representation

- Linked list (binary tree)

Tree data structure representation

- Array (binary tree)
  - array index corresponds to node number in the full tree.

Code Examples

• Page 267
  - For node definition

• Page 268
  - For Linked Binary Tree definition

```java
public boolean isInternal(Node v) {
    return (v.getLeft()!=NULL) && (v.getRight()!=NULL);
}
```