Binary Trees and Tree Traversals

T. M. Murali

Apr 12, 2004
Binary Trees

- A binary tree is a tree where each internal node has at most two children (exactly two in a proper binary tree).
- Usually, the children of a node are ordered: left and right child.
- Application: cancer classification

![Binary Tree Diagram]

[Diagram showing a binary tree with nodes labeled with conditions and outcomes for cancer classification.]
Properties of Proper Binary Trees

- Notation:
  - $n$, number of nodes
  - $l$, number of leaves
  - $i$, number of internal nodes
  - $h$, height

\[
\begin{align*}
n &= l + i \\
l &= e + 1 \\
i &\leq 2^h \\
h &\geq \log_2 e \\
 &\geq \log_2(n + 1) - 1
\end{align*}
\]
Properties of Proper Binary Trees

- Notation:
  - \( n \), number of nodes
  - \( l \), number of leaves
  - \( i \), number of internal nodes
  - \( h \), height

- \( n = l + i \)
Properties of Proper Binary Trees

- Notation:
  - $n$, number of nodes
  - $l$, number of leaves
  - $i$, number of internal nodes
  - $h$, height

- $n = l + i$

- $l = e + 1$
Properties of Proper Binary Trees

- **Notation:**
  - $n$, number of nodes
  - $l$, number of leaves
  - $i$, number of internal nodes
  - $h$, height

- $n = l + i$

- $l = e + 1$

- $e \leq 2^h$
Properties of Proper Binary Trees

- Notation:
  - $n$, number of nodes
  - $l$, number of leaves
  - $i$, number of internal nodes
  - $h$, height

- $n = l + i$
- $l = e + 1$
- $e \leq 2^h$
- $h \geq \log_2 e \geq \log_2(n + 1) - 1$
Binary Tree ADT

- BinaryTree ADT extends the Tree ADT and adds the following methods:
  - position left(p)
  - position right(p)
  - position hasLeft(p)
  - position hasRight(p)
- An implementation of the BinaryTree ADT may provide other update operations.
Linked Structure Implementation of Binary Trees

- Look at BTNode and LinkedBinaryTree classes in net.datastructures package.
Tree Traversals

- A traversal visits the nodes of a tree in a systematic manner (and performs some action at each node visited)
  - Pre-order: visit node before visiting children
  - In-order: visit node after visiting left node but before visiting right node.
  - Post-order: visit node after visiting children.
- Different traversals perform action at a node at different stages.
Applications of Tree Traversal