**Binary Tree Properties & Representation**

**Minimum Number Of Nodes**
- Minimum number of nodes in a binary tree whose height is $h$.
- At least one node at each of first $h$ levels.

Minimum number of nodes is $h$

**Maximum Number Of Nodes**
- All possible nodes at first $h$ levels are present.

Maximum number of nodes
$$= 1 + 2 + 4 + 8 + \ldots + 2^{h-1}$$
$$= 2^h - 1$$

**Number Of Nodes & Height**
- Let $n$ be the number of nodes in a binary tree whose height is $h$.
- $h \leq n \leq 2^h - 1$
- $\log_2(n+1) \leq h \leq n$

**Full Binary Tree**
- A full binary tree of a given height $h$ has $2^h - 1$ nodes.

Height 4 full binary tree.

**Numbering Nodes In A Full Binary Tree**
- Number the nodes 1 through $2^h - 1$.
- Number by levels from top to bottom.
- Within a level number from left to right.
Node Number Properties

• Parent of node \( i \) is node \( i / 2 \), unless \( i = 1 \).
• Node 1 is the root and has no parent.

Node Number Properties

• Left child of node \( i \) is node \( 2i \), unless \( 2i > n \), where \( n \) is the number of nodes.
• If \( 2i > n \), node \( i \) has no left child.

Node Number Properties

• Right child of node \( i \) is node \( 2i+1 \), unless \( 2i+1 > n \), where \( n \) is the number of nodes.
• If \( 2i+1 > n \), node \( i \) has no right child.

Complete Binary Tree With n Nodes

• Start with a full binary tree that has at least \( n \) nodes.
• Number the nodes as described earlier.
• The binary tree defined by the nodes numbered 1 through \( n \) is the unique \( n \) node complete binary tree.

Example

• Complete binary tree with 10 nodes.

Binary Tree Representation

• Array representation.
• Linked representation.
Array Representation

- Number the nodes using the numbering scheme for a full binary tree. The node that is numbered \( i \) is stored in \( \text{tree}[i] \).

```
<table>
<thead>
<tr>
<th>i</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>f</td>
<td>g</td>
<td>h</td>
<td>i</td>
<td>j</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Right-Skewed Binary Tree

- An \( n \) node binary tree needs an array whose length is between \( n+1 \) and \( 2^n \).

```
| i | 0 | 5 | 10 |
|---|---|----|
| a | b | c |
| 1 | 3 | 7 |
| d |
```

Linked Representation

- Each binary tree node is represented as an object whose data type is \text{BinaryTreeNode}.
- The space required by an \( n \) node binary tree is \( n \times (\text{space required by one node}) \).

```java
package dataStructures;
public class BinaryTreeNode {
    Object element;
    BinaryTreeNode leftChild; // left subtree
    BinaryTreeNode rightChild; // right subtree
    // constructors and any other methods
    // come here
}
```

Linked Representation Example

Some Binary Tree Operations

- Determine the height.
- Determine the number of nodes.
- Make a clone.
- Determine if two binary trees are clones.
- Display the binary tree.
- Evaluate the arithmetic expression represented by a binary tree.
- Obtain the infix form of an expression.
- Obtain the prefix form of an expression.
- Obtain the postfix form of an expression.
### Binary Tree Traversal

- Many binary tree operations are done by performing a *traversal* of the binary tree.
- In a traversal, each element of the binary tree is *visited* exactly once.
- During the *visit* of an element, all action (make a clone, display, evaluate the operator, etc.) with respect to this element is taken.

### Binary Tree Traversal Methods

- Preorder
- Inorder
- Postorder
- Level order