Binary Tree Traversal Methods

• Many binary tree operations are done by performing a traversal of the binary tree.

• Possible Binary Tree Operations:
  – Determine the height.
  – Determine the number of nodes.
  – Make a clone.
  – Evaluate the arithmetic expression represented by a binary tree.
  – …

Binary Tree Traversal Methods

• In a traversal of a binary tree, 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.

Preorder Traversal

```cpp
template <class T>
void PreOrder(TreeNode<T> *t)
{
    if (t != NULL)
    {
        Visit(t);
        PreOrder(t->leftChild);
        PreOrder(t->rightChild);
    }
}
```
Preorder Example (Visit = print)

Preorder Example (Visit = print)

Preorder Of Expression Tree

Merits Of Binary Tree Form

- Left and right operands are easy to visualize.
- Code optimization algorithms work with the binary tree form of an expression.
- Simple recursive evaluation of expression.

- $(a + b) * (c - d) / (e + f)$
- $/ * + a b - c d + e f$

Gives prefix form of expression!
Inorder Traversal

template <class T>
void InOrder(TreeNode<T> *t)
{
    if (t != NULL)
    {
        InOrder(t->leftChild);
        Visit(t);
        InOrder(t->rightChild);
    }
}

Inorder Example (Visit = print)

a
b
c
d e f
g h i
j

Inorder By Projection (Squishing)

g d h b e i a f j c
Inorder Of Expression Tree

Gives infix form of expression (without parentheses)!

Postorder Example (Visit = print)

Postorder Example (Visit = print)

Postorder Traversal

```c++
template <class T>
void PostOrder(TreeNode<T> *t)
{
    if (t != NULL)
    {
        PostOrder(t->leftChild);   PostOrder(t->rightChild);
        Visit(t);
    }
}
```
Postorder Of Expression Tree

```
+    
\  /  
+  *  
\  /  
\ b  c  
\  /  
\ a  d  
```

a b + c d - * e f + /

Gives postfix form of expression!

Traversal Applications

- Make a clone.
- Determine height.
- Determine number of nodes.

Level Order

Let t be the tree root.
while (t != NULL)
{
    visit t and put its children on a FIFO queue;
    if FIFO queue is empty, set t = NULL;
    otherwise, pop a node from the FIFO queue and call it t;
}

Level-Order Example (Visit = print)

```
a b c d e f g h i j
```
Binary Tree Construction

• Suppose that the elements in a binary tree are distinct.
• Can you construct the binary tree from which a given traversal sequence came?
• When a traversal sequence has more than one element, the binary tree is not uniquely defined.
• Therefore, the tree from which the sequence was obtained cannot be reconstructed uniquely.

Some Examples

preorder = ab
inorder = ab
postorder = ab
level order = ab

Binary Tree Construction

• Can you construct the binary tree, given two traversal sequences?
• Depends on which two sequences are given.

Preorder And Postorder

preorder = ab
postorder = ba

• Preorder and postorder do not uniquely define a binary tree.
• Nor do preorder and level order (same example).
• Nor do postorder and level order (same example).
Inorder And Preorder

- inorder = g d h b e i a f j c
- preorder = a b d g h e i c f j
- Scan the preorder left to right using the inorder to separate left and right subtrees.
- a is the root of the tree; gdhbei are in the left subtree; fjc are in the right subtree.

Inorder And Preorder

- preorder = a b d g h e i c f j
- b is the next root; gdh are in the left subtree; ei are in the right subtree.

Inorder And Preorder

- preorder = a b d g h e i c f j
- d is the next root; g is in the left subtree; h is in the right subtree.

Inorder And Preorder

- preorder = a b d g h e i c f j
- e is the next root; nothing is in the left subtree; i is in the right subtree.
Inorder And Preorder

- preorder = a b d g h e i c f j
- c is the next root; fj is in the left subtree; nothing is in the right subtree.

Inorder And Postorder

- Scan postorder from right to left using inorder to separate left and right subtrees.
- inorder = g d h b e i a f j c
- postorder = g h d i e b j f c a
- Tree root is a; gdhbei are in left subtree; fjc are in right subtree.

In Class Exercise

- Determine the tree
  - inorder = g d h b e i a f j c
  - postorder = g h d i e b j f c a

Inorder And Level Order

- Scan level order from left to right using inorder to separate left and right subtrees.
- inorder = g d h b e i a f j c
- level order = a b c d e f g h i j
- Tree root is a; gdhbei are in left subtree; fjc are in right subtree.
Homework

• Sec. 5.3 Exercise 10 @P 267
  – Remark: ADT 5.1 is defined @ P252