Binary Tree Traversals

Lecture 32
Section 19.2

Robb T. Koether

Hampden-Sydney College

Wed, Apr 12, 2017

Outline

Binary Tree Traversals

Definition (Binary Tree Traversal)

To traverse a binary tree is to visit systematically every vertex in the tree exactly once.

- There are only two natural ways to traverse a list: head to tail and tail to head.
- Because a tree is nonlinear, there are many ways to traverse it.

Outline

Pre-Order Traversals

Definition (Pre-Order Traversal)

A pre-order traversal of a binary tree visits the root node first, then performs a pre-order traversal of its left subtree first, and then a pre-order traversal of its right subtree. A pre-order traversal of an empty tree does nothing.

Pre-Order Traversals

Pre-Order Traversal

```
void preorderTraversal(BinaryTreeNode<T>* node,
    void visit(BinaryTreeNode<T>*)) const
{
    if (node != NULL)
    {
        visit(node);
        preorderTraversal(node->m_left, visit);
        preorderTraversal(node->m_right, visit);
    }
}
```

Outline

Post-Order Traversals

Definition (Post-Order Traversal)

A post-order traversal of a binary tree performs a post-order traversal of its left subtree first, then a post-order traversal of its right subtree, and then it visits its root node. A post-order traversal of an empty tree does nothing.

Post-Order Traversals

Post-Order Traversal

```
void postorderTraversal(BinaryTreeNode<T>* node,
    void visit(BinaryTreeNode<T>*)) const
{
    if (node != NULL)
    {
        postorderTraversal(node->m_left, visit);
        postorderTraversal(node->m_right, visit);
        visit(node);
    }
}
```

Outline

In-Order Traversals

Definition (In-Order Traversal)

A in-order traversal of a binary tree performs a in-order traversal of its left subtree first, then visits the root node, and then performs a in-order traversal of its right subtree. A in-order traversal of an empty tree does nothing.

In-Order Traversals

In-Order Traversal

```
void inorderTraversal(BinaryTreeNode<T>* node,
    void visit(BinaryTreeNode<T>*)) const
{
    if (node != NULL)
    {
        inorderTraversal(node->m_left, visit);
        visit(node);
        inorderTraversal(node->m_right, visit);
    }
}
```

Outline

Expression Trees

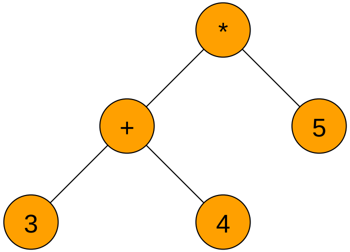
Definition (Expression Tree)

A binary expression tree is a binary tree that is used to represent an expressions with binary operators. Each interior node represents an operator. At each interior node, the left subtree represents the left operand and the right subtree represents the right operand.

- As a consequence, each terminal node represents an operand.
- The order of operation is indicated by the structure of the tree.

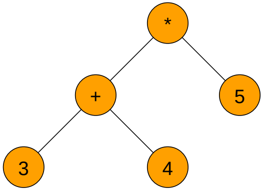
Expression Trees

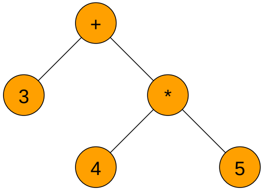
- For example, $(3 + 4) * 5$ may be represented as



Expression Trees

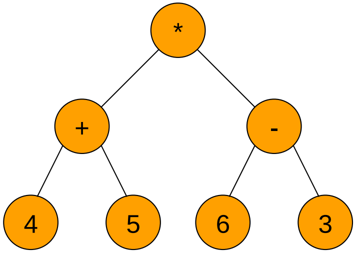
- If there is more than one operator, the order of operation is indicated by the structure of the tree.





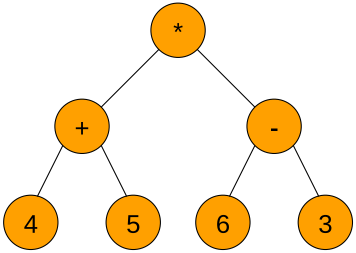
Traversals and Expression Trees

- Perform a pre-order traversal of the expression tree and print the nodes.



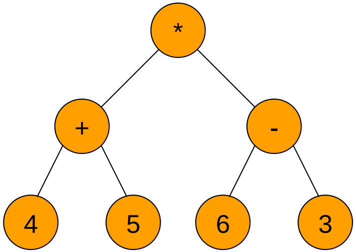
Traversals and Expression Trees

- Perform an in-order traversal of the expression tree and print the nodes.



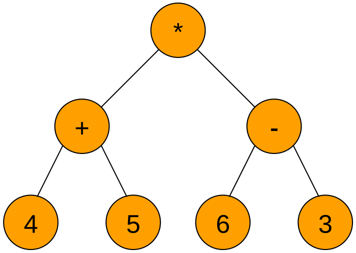
Traversals and Expression Trees

- Perform a post-order traversal of the expression tree and print the nodes.



Traversals and Expression Trees

- A post-order traversal is used to evaluate the expression.



Outline

Level-Order Traversals

Definition (Level-Order Traversal)

A level-order traversal of a binary tree visits the nodes level by level, from left to right, beginning with the root node

Level-Order Traversals

- To perform a level-order traversal, we need a queue.
- We begin by enqueueing the root node.
- Then do the following until the queue is empty.
    - Dequeue a node to get the next node to be visited.
    - Visit the node.
    - Enqueue its right child (if there is one) followed by its left child(if there is one).

Outline

Breadth-First vs. Depth-First Searches

- A depth-first search will pursue each branch of the tree to a leaf before backtracking and trying another branch.
- A breadth-first search pursues all branches to the same level before pursuing any branch to the next level.

Breadth-First vs. Depth-First Searches

- A depth-first search will pursue each branch of the tree to a leaf before backtracking and trying another branch.
- A breadth-first search pursues all branches to the same level before pursuing any branch to the next level.
- Which traversal(s) will perform a depth-first search?

Breadth-First vs. Depth-First Searches

- A depth-first search will pursue each branch of the tree to a leaf before backtracking and trying another branch.
- A breadth-first search pursues all branches to the same level before pursuing any branch to the next level.
- Which traversal(s) will perform a depth-first search?
- Which traversal(s) will perform a breadth-first search?

Outline

Assignment

Assignment

- Read Section 19.2.