

For Friday

- No reading
- Recommended homework problems:
 - Checkpoint 11.1

Program 8

- Any questions?

Exam 3

- Wednesday
- Covers linked lists through recursion

Review questions?

Binary Tree Traversal

- In a binary tree **traversal**, we **visit** each node in the tree exactly once
- There are four different orders in which we often choose to visit the nodes
 - Preorder
 - Inorder
 - Postorder
 - Level order

Preorder Traversal

- ```
public static void preOrder(BinTreeNode tree)
{ // preOrder traversal of tree
 if (tree!=null) {
 Visit(tree); // visit the root
 preOrder(tree.left) // do left subtree
 preOrder(tree.right) // do right subtree
 }
}
```

# Inorder Traversal

- ```
public static void inOrder(BinTreeNode tree)
{    // inOrder traversal of tree
    if (tree != null) {
        inOrder(tree.left)        // do left subtree
        Visit(tree);              // visit the root
        inOrder(tree.right)       // do right subtree
    }
}
```

Postorder Traversal

- ```
public static void postOrder(BinTreeNode* tree)
{ // postOrder traversal of tree
 if (tree != null) {
 postOrder(tree.left) // do left subtree
 postOrder(tree.right) // do right subtree
 Visit(tree); // visit the root
 }
}
```

# Level Order Traversal

```
public static void levelOrder(BinTreeNode tree)
{ // level order traversal of tree
 Queue q;
 while(tree != null) {
 Visit(tree); // visit tree
 // put children on the queue
 if (tree.left != null) q.add(tree.left);
 if (tree.right != null) q.add(tree.right);
 // get next node to visit
 if (q.isEmpty())
 tree = null;
 else
 tree = q.delete();
 }
}
```

# Uses for Traversals



# Dictionary Implementations

# Binary Search Trees

- An addition to our possible dictionary implementations
- We're interested in operations:
  - Search
  - Insert
  - Delete
  - Output in ascending order of keys
  - FindMin
  - FindMax

# Definition

- Binary tree
- Each element has an associated key
- Keys are not duplicated
- For any node, all nodes in its left subtree (if any) have keys smaller than its key and all nodes in its right subtree (if any) have keys larger than its key

# Binary Search Tree Operations

- Search
  - Time Complexity
- Insert
  - Time Complexity
- Deletion
  - Of a leaf
  - Of a node with one non-empty subtree
  - Of a node with two non-empty subtrees
  - Time Complexity
- Output Ordered List
- FindMin
- Find Max