**King Fahd University of Petroleum & Minerals**

**College of Computer Science and Engineering**

**Information and Computer Science Department**

**ICS 202 – Data Structures**

# Lab 06: Binary and Binary Search Trees

**Objectives**

The objective of this lab is to design, implement and use binary and binary search trees.

**Outcomes**

After completing this Lab, students are expected to:

• Understand Generic classes for binary and binary search trees.

• Implement methods for binary and binary search trees.

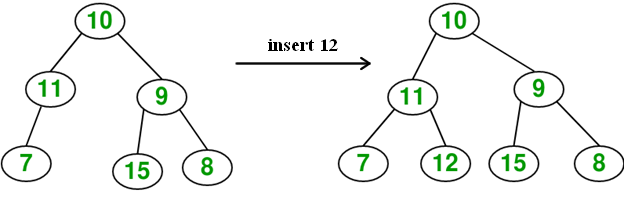
• Traverse binary trees (breadth-first, depth-first)

• Insert into and delete from binary and binary search trees.

**BinaryTree insertion and deletion**

There are no fixed rules for inserting and deleting from a Binary-tree. In our Binary-tree implementation, we use the insertion and deletion algorithms given below:

* Given a binary tree and a key, insert the key into the binary tree at the first position available in level order traversal.



Note: We can create a Binary tree without using the insert method. We do this by creating the root node and then linking it with other nodes:

BinaryTree<Integer> tree = new BinaryTree<Integer>();

BTNode node1 = new BTNode(7);

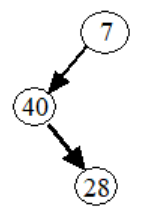
BTNode node2 = new BTNode(40);

BTNode node3 = new BTNode(28);

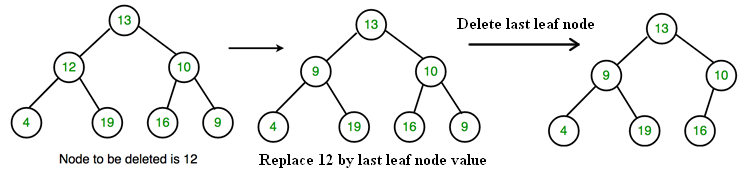
tree.root = node1

node1.left= node2;

node2.right = node3;



* Given a binary tree, delete a node from it by making sure that tree shrinks from the bottom (i.e. the deleted node is replaced by the last leaf node).

****

**Note:**

**For the purpose of this lab you are allowed to use the given BinaryTree, BST and BTNode classes only.**

**Lab Tasks**

1. Write a recursive instance method **public boolean subtreesHaveEqualNumberOfNodes()** of the **BinaryTree<T>** class that returns true if the invoking BinaryTree<T> object has an equal number of nodes in its left and right subtrees. Your method must throw java.lang.UnsupportedOperationException if the invoking tree is empty.
2. Write a recursive instance method **public int numOneChildNodes()** of the **BinaryTree<T>** class that counts the number of one-child nodes in the invoking BinaryTree<T> object. Your method must throw java.lang.UnsupportedOperationException if the invoking tree is empty.

1. Write a recursive instance method **public String getPathToLeafNode(T e)** ofthe **BST<T>** class that returns the path to a leaf node with data e. Your method must throw appropriate exceptions of the tree is empty, or if e is not the data of a leaf node, or if a node with data e does not exist in the tree.
2. Write a **recursive** instance method **int getNodeLevel(T e)** to find the level of a node with key **e** of a binary search tree. Your method must throw java.lang.UnsupportedOperationException if the invoking tree object is empty. It must throw **java.util.NoSuchElementException** if there is no node with data **e** in the invoking **BST<T>** object.
3. **Test program for BinaryTree.**

Write a test program that creates the binary tree shown below without using the insert metho. It then generates the output shown in the sample program run.

A picture containing accessory, necklet, key, bicycle

Description automatically generated

A sample program run is:

The number of one-child nodes in the tree is 2  
 The root subtrees have equal number of nodes: false  
 The key 5 is in the tree.  
 Preorder Traversal is:   
 1 2 4 5 12 3 8   
 Inorder Traversal is:   
 4 2 12 5 1 3 8   
 Before deleting key 3, level order traversal of binary tree is:   
 1 2 3 4 5 8 12   
 The tree is:   
 tR----1  
 L----2  
 | L----4  
 | R----5  
 | L----12  
 R----3  
 R----8  
  
 After deleting key 3, level order traversal of binary tree is:   
 1 2 12 4 5 8   
 The tree is:   
 tR----1  
 L----2  
 | L----4  
 | R----5  
 R----12  
 R----8

1. **Test program for BST**

MMM  
 Complete the given **BST\_Driver.java** to generate the following output:

Path from root to node G is: D F H G  
  
 The BST is:   
 tR----D  
 L----B  
 | L----A  
 | R----C  
 R----F  
 R----H  
 L----G  
 R----J  
   
 Preorder traversal: D B A C F H G J   
 Inorder traversal: A B C D F G H J   
 Postorder traversal: A C B G J H F D   
 Level order traversal: D B F A C H G J   
 Level order traversal by levels:   
 D  
 B F  
 A C H  
 G J  
 The level of node G is: 3  
 Tree after deleting node D by copying:

tR----C  
 L----B  
 | L----A  
 R----F  
 R----H  
 L----G  
 R----J