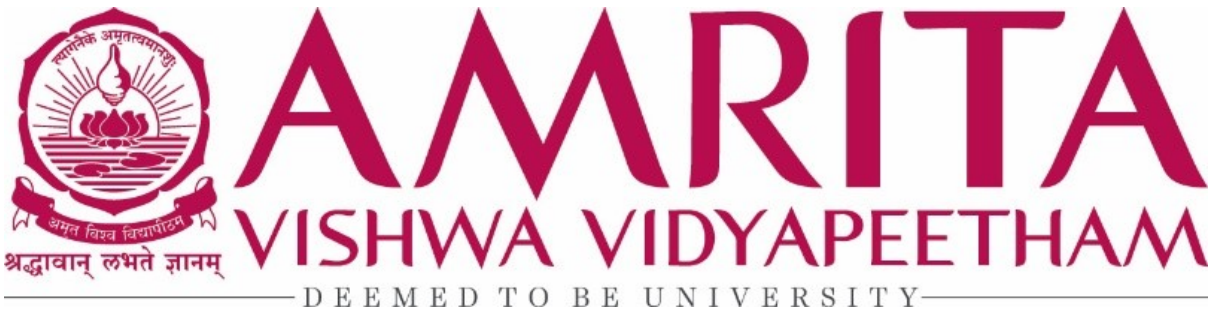
`5Assignment 6

21AIE111

Data Structure and Algorithms – SEM-II

Professor – Dr. Sachin Sir

Submitted By: Vikhyat Bansal [CB.EN.U4AIE21076]



1. Write java code to create a binary tree using arrays.

CODE:

```
// JAVA implementation of tree using array
// numbering starting from 0 to n-1.

import java.io.*;
import java.lang.*;
import java.util.*;

class BTreeArr {

  // Main driver method
  public static void main(String[] args)
  {

    // Creating object of class 2 inside main() method
    Array_imp tree = new Array_imp();

    //Setting root node

    /* Our tree from this input will look something like this*/
    /*         A(0)
              /    \
             /      \
            /        \
      B(1)            C(2)
      /   \            /   \
     /     \          /     \
D(3)       E(4)     F(5)    G(6)
 /   \     /   \    /       /   \
 H(7)     I(8)                        */


    tree.Root("A");

    tree.set_Left("B", 0);
    tree.set_Right("C", 0);
    tree.set_Left("D", 1);
    tree.set_Right("E", 1);
    tree.set_Left("F", 2);
    tree.set_Right("G", 2);
```

```
    tree.set_Left("H", 3);
    tree.set_Left("I", 4);
    tree.print_Tree();
  }
}

class Array_imp {

  // Member variables of this class
  static int root = 0;
  static String[] str = new String[10]; //Array is limited to n
amount of nodes

  public void Root(String key)
  { str[0] = key; }

  public void set_Left(String key, int root)
  {
    int t = (root * 2) + 1;

    if (str[root] == null) {
      System.err.println();
      System.out.printf(
        "Can't set child at " +t+", no parent found");
    }
    else {
      str[t] = key;
    }
  }

  public void set_Right(String key, int root)
  {
    int t = (root * 2) + 2;

    if (str[root] == null) {
      System.out.println();
      System.out.printf(
        "Can't set child at " +t+", no parent found"
        );
    }
    else {
      str[t] = key;
```

```
    }
  }

  public void print_Tree()
  {
    System.out.println();

    // Iterating using for loop
    for (int i = 0; i < 10; i++) {
      if (str[i] != null)
        System.out.print(str[i] +  " ");

      else
        System.out.print("-");  // printing '-' if node is null and
a node is missing in connection.
    }
  }
}
```

OUTPUT:

```
PS D:\CODING\JAVA\VS Code> java .\Assignment_6\BTreeArr.java

A B C D E F G H -I
```

2. Write java code to create a binary tree using linked list.

CODE:

```
// Java program to create complete Binary Tree from its Linked List
// representation

// importing necessary classes
import java.util.*;

// A linked list node

class Driver {
    // Driver program to test above functions
    public static void main(String[] args)
    {
        BinaryTree tree = new BinaryTree();
        tree.push(1); /* Last node of Linked List */
        tree.push(5);
        tree.push(10);
        tree.push(15);
        tree.push(20); /* First node of Linked List */
        BinaryTreeNode node = tree.convertList2Binary(tree.root);

        System.out.println("Inorder Traversal of the"+
                           " constructed Binary Tree is:");
        tree.inorderTraversal(node);

   /* We can use different traversals to go through the tree. */

    }
}

class ListNode
{
    int data;
    ListNode next;
    ListNode(int d)
    {
        data = d;
        next = null;
    }
}
```

```
// A binary tree node
class BinaryTreeNode
{
    int data;
    BinaryTreeNode left, right = null;
    BinaryTreeNode(int data)
    {
        this.data = data;
        left = right = null;
    }
}

class BinaryTree
{
    ListNode head;
    BinaryTreeNode root;

    // Function to insert a node at the beginning of
    // the Linked List
    void push(int new_data)
    {
        // allocate node and assign data
        ListNode new_node = new ListNode(new_data);

        // link the old list off the new node
        new_node.next = head;

        // move the head to point to the new node
        head = new_node;
    }

    // converts a given linked list representing a
    // complete binary tree into the linked
    // representation of binary tree.
    BinaryTreeNode convertList2Binary(BinaryTreeNode node)
    {
        // queue to store the parent nodes
        Queue<BinaryTreeNode> q =
                    new LinkedList<BinaryTreeNode>();

        // Base Case
```

```
        if (head == null)
        {
            node = null;
            return null;
        }

        // 1.) The first node is always the root node, and
        //   add it to the queue
        node = new BinaryTreeNode(head.data);
        q.add(node);

        // advance the pointer to the next node
        head = head.next;

        // until the end of linked list is reached, do the
        // following steps
        while (head != null)
        {
            // 2.a) take the parent node from the q and
            //   remove it from q
            BinaryTreeNode parent = q.peek();

            // 2.c) take next two nodes from the linked list.
            // We will add them as children of the current
            // parent node in step 2.b. Push them into the
            // queue so that they will be parents to the
            // future nodes
            BinaryTreeNode leftChild = null, rightChild = null;
            leftChild = new BinaryTreeNode(head.data);
            q.add(leftChild);
            head = head.next;
            if (head != null)
            {
                rightChild = new BinaryTreeNode(head.data);
                q.add(rightChild);
                head = head.next;
            }

            // 2.b) assign the left and right children of
            //   parent
            parent.left = leftChild;
            parent.right = rightChild;
```

```
            //remove current level node
            q.poll();
        }

        return node;
    }

    // Utility function to traverse the binary tree
    // after conversion
    void inorderTraversal(BinaryTreeNode node)
    {
        if (node != null)
        {
            inorderTraversal(node.left);
            System.out.print(node.data + " ");
            inorderTraversal(node.right);
        }

    }

    void preorderTraversal (BinaryTreeNode node) {
        if (node != null) {
            System.out.print(node.data + " ");
            preorderTraversal(node.left);
            preorderTraversal(node.right);
        }
    }

    void postorderTraversal (BinaryTreeNode node) {
        if(node != null){
            postorderTraversal(node.left);
            postorderTraversal(node.right);
            System.out.print(node.data+" ");

        }

    }

}
```

OUTPUT:

```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\CODING\JAVA\VS Code> java .\Assignment_6\BtreeLL.java
Inorder Traversal of the constructed Binary Tree is:
5 15 1 20 10
```

```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\CODING\JAVA\VS Code> java .\Assignment_6\BtreeLL.java
Preorder Traversal of the constructed Binary Tree is:
20 15 5 1 10
```

```
PS D:\CODING\JAVA\VS Code> java .\Assignment_6\BtreeLL.java
Postorder Traversal of the constructed Binary Tree is:
5 1 15 10 20
```

3. Write java code to insert a node in a binary tree (at internal nodes location)

CODE:

```
import java.util.LinkedList;
import java.util.Queue;

public class InsertInBTree {
    // Java program to insert element in binary tree
    /* A binary tree node has key, pointer to
    left child and a pointer to right child */
    static class Node {
        int key;
        Node left, right;

        // constructor
        Node(int key)
        {
            this.key = key;
            left = null;
            right = null;
        }
    }
    static Node root;
    static Node temp = root;

    /* Inorder traversal of a binary tree*/
    static void inorder(Node temp)
    {
        if (temp == null)
            return;

        inorder(temp.left);
        System.out.print(temp.key + " ");
        inorder(temp.right);
    }

    static void preorder(Node temp)
    {
        if (temp == null)
            return;

        System.out.print(temp.key + " ");
```

```
        preorder(temp.left);
        preorder(temp.right);

    }

    static void postorder(Node temp)
     {
        if(temp == null)
            return;

        postorder(temp.left);
        postorder(temp.right);
        System.out.print(temp.key+" ");

    }

    /*function to insert element in binary tree */
    static void insert(Node temp, int key)
    {

        if (temp == null) {
            root = new Node(key);
            return;
        }
        Queue<Node> q = new LinkedList<Node>();
        q.add(temp);

        // Do level order traversal until we find
        // an empty place.ei
        while (!q.isEmpty()) {
            temp = q.peek();
            q.remove();

            if (temp.left == null) {
                temp.left = new Node(key);
                break;
            }
            else
                q.add(temp.left);

            if (temp.right == null) {
                temp.right = new Node(key);
```

```
                break;
            }
            else
                q.add(temp.right);
        }
    }

    // Driver code
    public static void main(String args[])
    {
          /* Our tree from this input before insertion will look
something like this
                69
              /    \
             /      \
            /        \
         9             5
       /              /   \
      /              /     \
    96              3
                     */

        root = new Node(69);
        root.left = new Node(9);
        root.left.left = new Node(96);
        root.right = new Node(5);
        root.right.left = new Node(3);


        System.out.print(
            "Inorder traversal before insertion: ");
        inorder(root);

        int key = 12;
        insert(root, key);

        int key1 = 19;
        insert(root, key1);

        /* Our tree from this input before insertion will look
something like this
```

```
                69
              /    \
             /      \
            /        \
           /          \
          9            5
        /   \        /   \
       /     \      /     \
      96     12     3
```

```
                69
              /    \
             /      \
            /        \
           /          \
          9            5
        /   \        /   \
       /     \      /     \
      96     12     3      19
```

```
*/

        System.out.print(
            "\nInorder traversal after insertion: ");
        inorder(root);
    }
}
```

OUTPUT:

(Inorder)

```
PS D:\CODING\JAVA\VS Code> java .\Assignment_6\InsertInBTree.java
Inorder traversal before insertion: 96 9 69 3 5
Inorder traversal after insertion: 96 9 12 69 3 5 19
```

(Preorder)

```
PS D:\CODING\JAVA\VS Code> java .\Assignment_6\InsertInBTree.java
Preorder traversal before insertion: 69 9 96 5 3
Preorder traversal after insertion: 69 9 96 12 5 3 19
```

(Postorder)

```
PS D:\CODING\JAVA\VS Code> java .\Assignment_6\InsertInBTree.java
Postorder traversal before insertion: 96 9 3 5 69
Postorder traversal after insertion: 96 12 9 3 19 5 69
```

4. Write a java code to delete a node in a binary tree (at internal nodes location).

CODE:

```
public class BinaryTreeNode {

    class Node {
        int data;
        Node left, right;

        public Node(int item) {
            data = item;
            left = right = null;
        }
    };

    Node root;

    BinaryTreeNode() {
        root = null;
    }

    private Node insert_rec(int d, Node r) {
        if(r == null) {
            return new Node(d);
        }
        else {
            if(d<r.data)
                r.left = insert_rec(d,r.left);
            else
                r.right = insert_rec(d,r.right);
            return r;
        }

    }

    public void insert( int d) {
        root = insert_rec(d,root);
    }

    public boolean isLeaf(){
        return (root.right==null && root.left==null);
    }
```

```
    private void inorder_rec(Node r) {
        if(r==null){
            // System.out.print("-");
        }
        else {
            inorder_rec(r.left);
            System.out.print(r.data+" ");
            inorder_rec(r.right);
        }
    }

    public void inorder() {
        System.out.println("Inorder traversal of the binary tree is:
");
        inorder_rec(root);

        System.out.println();
    }

    private void preorder_rec(Node r) {
        if(r==null){
            // System.out.print("-");
        }
        else {
            System.out.print(r.data+" ");
            preorder_rec(r.left);
            preorder_rec(r.right);
            }
    }

    public void preorder() {
        System.out.println("Preorder traversal of binary tree is:
");
        preorder_rec(root);

        System.out.println();
    }

    private void postorder_rec(Node r) {
        if(r==null){
            // System.out.print("-");
```

```
        }
        else {
            postorder_rec(r.left);
            postorder_rec(r.right);
            System.out.print(r.data+" ");

        }
    }

    public void postorder() {
        System.out.println("Postorder traversal of binary tree is:
");
        postorder_rec(root);
        System.out.println();
    }


    public int min(Node r) {
        while(r.left != null)
            r=r.left;
        return r.data;
    }

    public int max(Node r) {
        while(r.right != null)
            r=r.right;
        return r.data;
    }


    private Node delete_rec(int key, Node r) {
        if(r==null)
            return r;
        else if(key<r.data)
            r.left = delete_rec(key,r.left);
        else if(key>r.data)
            r.right = delete_rec(key,r.right);
        else {
            if(r.left==null)
                return r.right;
            else if(r.right==null)
                return r.left;
```

```
            else {
                r.data = min(r.right);
                r.right = delete_rec(r.data,r.right);
            }
        }
        return r;

    }

    public void delete(int k){
        delete_rec(k,root);
    }


    public static void main(String []args){
        BinaryTreeNode b = new BinaryTreeNode();
        b.insert(5);
        b.insert(7);
        b.insert(6);
        b.insert(2);
        b.insert(10);

        /* Our tree from this input before insertion will look
something like this
                  5
                /    \
               /      \
              /        \
            2           7
                      /   \
                     /     \
                    6      10
                     */

        b.delete(5);

            /* Our tree after deletion of node with value 5 will
look something like this
                  6
                /    \
               /      \
              /        \
```

```
            2           7
                         \
                          \
                          10
                    */

        b.inorder();
        b.preorder();
        b.postorder();
    }
}
```

OUTPUT:

```
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\CODING\JAVA\VS Code> java .\Assignment_6\DeleteinBtree.java
Inorder traversal of the binary tree after deletion is:
2 6 7 10
Preorder traversal of binary tree after deletion is:
6 2 7 10
Postorder traversal of binary tree after deletion is:
2 10 7 6
```

THANK YOU!!