Tree

**Please read turn-in checklist at the end of this document before you start doing exercises.**

**Section 1: Pen-and-paper Exercises**

1. Given a BST and a positive integer k, find the k\_th smallest element in the BST.

For example, in the following BST, if k = 3, then output should be 10, and if k = 5, then output should be 14.



Assume the tree is balanced, and the tree height is O(log n). Design an O(n) time algorithm to solve this problem.

(i) describe the idea behind your algorithm in English (2 points);

First create an ArrayList<Node> and then put the elements in order using a recursive inOrder method. After it is put in order you can return the k-1 value of the ArrayList<Node> to get the k\_th smallest element.

(ii) provide pseudocode (5 points);

inOrder(Node r) // inOrder method used within kthSmallestElement()

if (r != null)

inOrder(r.left);

x.add(r);

inOrder(r.right);

kthSmallestElement(Node p, int k)

x = new ArrayList<Node>(size(p));

inOrder(p);

return x.get(k-1);

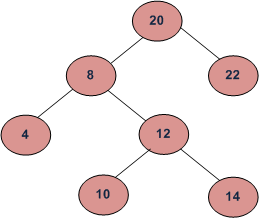
(iii) analyze its running time (3 points).

Every node is visited thus it isO(n)

**Note: Full credit (10 points) will be awarded for an algorithm that is O(n). Algorithms that are slower than O(n) will be scored out of 3 points.**

1. Given two values k1 and k2 (where k1 < k2) and a root pointer to a Binary Search Tree. Print all the keys of tree in range k1 to k2 in increasing order.

For example, in the following BST, if k1 = 10 and k2 = 22, then your function should print 10, 12, 14, 20 and 22.



Assume the tree is balanced, and the tree height is O(log n). Design an O(n) time algorithm to solve this problem.

(i) describe the idea behind your algorithm in English (2 points);

The method will take in parameters for the node(p), the lower bound, and the upper bound. If the node is null then nothing is done. If the node is not null then recursive call k1k2Range for p.left. Check if k1 is less than or equal to p.data && k2 is greater than or equal to p.data. If it is print out p.data and if not just go to the next line. Regardless the last line will run which is another recursive call this time using p.right. This runs until p is null.

(ii) provide pseudocode (5 points);

public static void k1k2Range(Node p, int k1, int k2)

{

// Complete the method to find all values in [k1,k2] in a BST

// Feel free to change the return type, parameters

if (p != null)

{

k1k2Range(p.left,k1,k2);

if(k1<=p.data && k2>=p.data) {

System.out.print(p.data + " ");

}

k1k2Range(p.right,k1,k2);

}

}

(iii) analyze its running time (3 points).

Every node is visited thus it isO(n)

**Note: Full credit (10 points) will be awarded for an algorithm that is O(n). Algorithms that are slower than O(n) will be scored out of 3 points.**

**Section 2: Java Implementation**

1. Implement the problem 1 in Java.

Note:

Find a file called BST.java in assignment 7 folder.

Complete the method of kthSmallestElement ().

Test your method in the main method provided following the comments.

1. Implement the problem 2 in Java.

Note:

Find a file called BST.java in assignment 7 folder.

Complete the method of k1k2Range ().

Test your method in the main method provided following the comments.

1. We discussed how to find the minimum and maximum node in a BST.

Implement findmin in Java.

Note:

Find a file called BST.java in assignment 7 folder.

Complete the method of findmin ().

Test your method in the main method provided following the comments.

1. Implement findmax in Java.

Note:

Find a file called BST.java in assignment 7 folder.

Complete the method of findmax ().

Test your method in the main method provided following the comments.

**TURN-IN CHECKLIST:**

1. **Answers to Section 1 (.doc/.txt/.pdf), and to Section 2 (all your source Code (.java files)). Remember to include your name, the date, and the course number in comments near the beginning of your code/report.**
2. **Create a folder and name it 'FirstName\_LastName\_assignment\_7'. In the newly created folder copy and paste your files (.doc/.txt/.java files). Then compress the folder, and push it to iLearn.**