CS21003: Algorithms-I (Theory)

Tutorial – 5 (Binary Trees, BSTs and Height-balanced / AVL Trees)

Date: 05-March-2020

1. Given a binary tree T having n nodes, you are asked to print all its elements in a level-by-level order. For illustration, consider the following example:

```
L1        1
         / \
L2      2   3
       / \   \     Level Order is: 1 -> 2 -> 3 -> 4 -> 5 ->
L3    4   5   6                   6 -> 7 -> 8 -> 9 -> 10 -> 11
         /   /
L4      7   8
       /   / \
L5    9  10   11
```

   What is the time and space complexity of your proposed solution to this problem?

2. Given two binary trees $T1$ and $T2$, write an algorithm/function to detect if $T1$ and $T2$ are *isomorphic*. Two trees are called *isomorphic* if one of them can be obtained from other by a series of flips, i.e. by swapping left and right children (with their decendent subtrees) of a number of nodes. Two empty trees are isomorphic as well. For example, the two following trees are isomorphic with the following subtrees flipped – 2 and 3, `NULL` and 6, 7 and 8.

```
     1                          1
   /   \                      /   \
  2     3                    3     2
 / \   /   <= ISOMORPHIC =>   \   / \
4   5 6                        6 4   5
   / \                              / \
  7   8                            8   7
```

   What is the running time complexity of your proposed approach?

3. Given a Binary Search Tree (BST), T, and two values k_1 and k_2 ($k_1 < k_2$), write an algorithm/function to print all the keys of T in the range $[k_1, k_2]$, i.e. print all x such that $k_1 \leq x \leq k_2$ and x is a key of T. Print all the keys in increasing order. What is the time complexity of your proposed algorithm?

4. Given a sorted array of integers elements, propose a recursive algorithm to create a *balanced* BST using the given sorted array elements. Deduce the running time of your algorithm.

5. Given an AVL tree (height-balaced BST) with n nodes, what is the minimum and maximum height possible for the AVL tree? Prove / Derive your result.