# CS608-SPRING2023: ALGORITHMS & COMPUTING THEORY

**Assignment#3** **-** **TOTAL** **POINTS:** **100** **DUE** **DATE:** **04/16/2023** **(April** **16th)**

**Team** **Assignment**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Questions** | **Points** | **Self-**  **Assessment** |
| **1** | **Validate** **Binary** **Search** **Tree**  Determine if a given root of a tree is a valid binary search tree (BST) A valid BST is defined as follows:   * Given root, the **left** **subtree** of a node contains only nodes with keys **less** **than** **the** **node's** **key**. * Given root, the **right** **subtree** of a node contains only nodes with keys **greater** **than** **the** **node's** **key**. * Ensure that both the left and right subtrees are also binary search trees.   **Example:**    **Input:** root = [2,1,3]  **Output:** true | **30** | The code includes methods for creating the tree from user input and determining whether it is a valid binary search tree (BST), as well as for implementing a binary tree data structure with a TreeNode class.  The isValidBST method checks node values against upper and lower boundaries in a recursive manner to validate the BST property, whereas the buildTree function builds the tree level-wise utilizing a queue for speedy building.  Points: 30 |
| **2** | **Balanced** **Binary** **Tree**  Determine if a binary tree is height-balanced.  A height-balanced binary tree is defined as a binary tree in which the left and right subtrees of every node differ in height by no more than 1.  **Example:** | **40** | The binary tree is built level-by-level in the code using a queue-based strategy, assuring proper development of the left and right child nodes.  The code uses a recursive function to calculate each node's height, making it possible to quickly determine whether a property is height-balanced by comparing height disparities.  Points: 40 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Input:** root = [3,9,20,null,null,15,7]  **Output:** true |  |  |
| **3** | **Convert** **Sorted** **Array** **to** **Binary** **Search** **Tree**  Given an integer array, where the elements are sorted in **ascending** **order**, convert it to a **height-balanced** binary search tree.  A **height-balanced** binary tree is a binary tree in which the depth of the two subtrees of every node never differs by more than one.  **Example** **1:**    **Input:** nums = [-10,-3,0,5,9]  **Output:** [0,-3,9,-10,null,5]  **Explanation:** [0,-10,5,null,-3,null,9] is also accepted: | **30** | The code successfully creates a balanced binary search tree (BST) from a sorted array using a recursive method, producing an effective solution with an O(n) time complexity.  The program handles input and output adequately, accepting user input, saving tree nodes in level-order, producing output in the chosen format, and correctly handling empty arrays and None values.  Points: 30 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

**Submission**

* Submit a python **notebook**(of file type **.ipynb**) with comments above each code block/line explaining its purpose. Also, submit **screenshot** of the result/output you get.
* You may not be graded full points if your program doesn’t execute or produce the intended results.
* Late submission up to one week after the **due** **date** will incur a **10%** **loss** of total points earned. 5% every week thereafter until the end date.

## Be careful not to share your code. You may lose points by sharing your work. Similarity scores will be checked.

* Attach this file with self-assessment. This is for your reference if you answered the question completely.