**Largest BST:-**

Given a binary tree. Find the size of its largest subtree that is a Binary Search Tree.

**Example 1:**

**Input:**

  1

  / \

  4 4

  / \

  6 8

**Output:** 1

**Explanation:** There's no sub-tree with size

greater than 1 which forms a BST. All the

leaf Nodes are the BSTs with size equal

to 1.

**Example 2:**

**Input:** 6 6 3 N 2 9 3 N 8 8 2

  6

  / \

  6 3

  \ / \

  2 9 3

  \ / \

  8 8 2

**Output:** 2

**Explanation:** The following sub-tree is a

BST of size 2:

      2

    /    \

  N      8

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function**largestBst()**that takes the root node of the Binary Treeas its input and returns the size of the largest subtree which is also the BST. If the complete Binary Tree is a BST, return the size of the complete Binary Tree.

**Expected Time Complexity:**O(N).  
**Expected Auxiliary Space:**O(Height of the BST).

**Constraints:**  
1 <= Number of nodes <= 100000  
1 <= Data of a node <= 1000000