Exam 3 Review Packet

10/22/2023 - M107

1. True / False Section (General Knowledge)
   1. **T F**  The array in a binary min heap is always in ascending sorted order.
   2. **T F**  Fixing an imbalance in a height balanced tree, after inserting a value and an imbalance is found, is O(1).
   3. **T F**  After insertion in an Editor Tree, you need to recalculate the height at some nodes to determine the new balance code
   4. **T F**  In a hash table, the purpose of quadratic probing is to solve the problem of clustering that occurs with linear probing.
   5. **T F**  Inserting a node in the right subtree of node N in an AVL tree may cause the rank of N to change.
   6. **T F**  A single right rotation at node N in an AVL tree may cause the rank of N to change.
2. A binary tree with height 4…
   1. can have a minimum number of nodes of? \_\_\_\_\_\_\_\_
   2. can have a max number of leaf nodes of? \_\_\_\_\_\_\_\_
   3. must have at least how many nodes if it is balanced? \_\_\_\_\_\_\_\_
3. Consider an editor tree using nodes with the following fields

char element;

Node left;

Node right;

int rank;

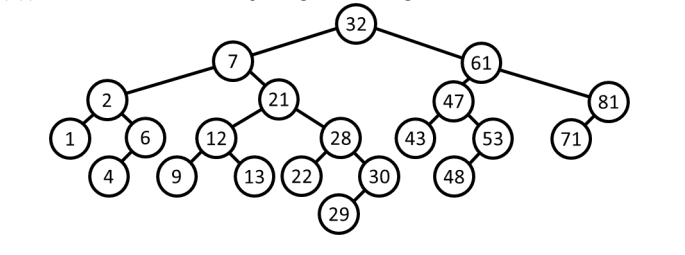
Code balance;

Write a method that returns the total number of left subtrees that are not null.

// p might be null or a NULL\_NODE.

public static int leftSubtrees(Node p){

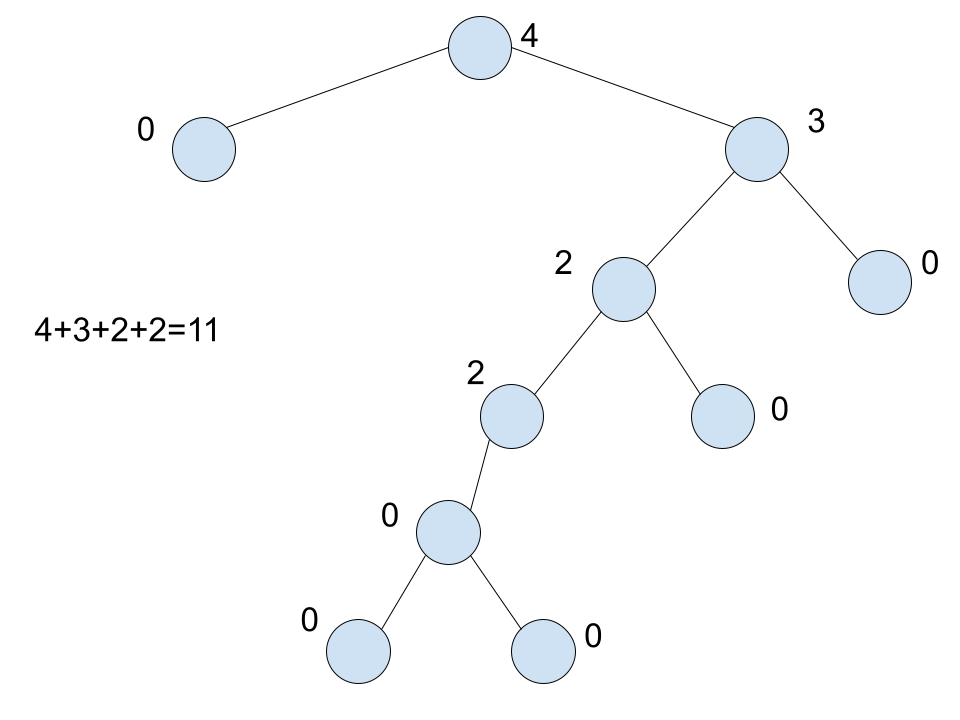
1. Consider the AVL tree below. (a)-(d) Fill in the table below. For each part, begin with the original tree.



|  |  |  | Single, double, or none? | Left or right? | Root of rotated subtree before rotation (i.e., “imbalanced node”) | Root of rotated subtree after rotation |
| --- | --- | --- | --- | --- | --- | --- |
| Part a. | Insert 70 | 1st rotation |  |  |  |  |
| 2nd rotation |  |  |  |  |
| Part b. | Insert 5 | 1st rotation |  |  |  |  |
| 2nd rotation |  |  |  |  |
| Part c. | Delete 6 | 1st rotation |  |  |  |  |
| 2nd rotation |  |  |  |  |
| Part d. | Delete 43 | 1st rotation |  |  |  |  |
| 2nd rotation |  |  |  |  |

1. totalImbalance (coding)

Given a BST, find the total imbalance of the tree. Defined as the sum of the positive difference between every left and right subtree for every node in the tree.



1. numCollidedEntries (coding)

In the hash package, implement the method int numCollidedEntries(). We simplified the implementation of the StringHashSet to use linear probing and to not grow the array and re-hash entries. You are to return the number of entries that must have collided with at least one other item when inserted. You can determine it by writing code to examine the internal hash array using knowledge of how linear probing works.

Note: do not re-run add() or contains() to determine this –changing either of those methods or adding fields to the class is not allowed. You may refer back to your implementation of the StringHashSet assignment as needed.

1. Pre-order + inorder -> tree

In HW4, you were given the preorder and inorder traversals of a tree, and asked to determine the shape of the tree. In this problem you are to turn that informal process into actual code to automatically build a tree based on these two traversals. Complete this method in the PreorderInorderBuild class:

/\*\* \* Returns the root of the tree that has the given traversal strings.

\* @param pre The pre-order traversal

\* @param in The in-order traversal.

\* @return The root of the tree described above.

\*/

public static BinaryNode buildFromPreOrderInorder(String pre, String in)