13

1. **BST/AVL Tree (25 pts;15+10)**

a) A BST is an AVL tree if *for every node*, the difference in the heights of its children is *at most* one. (The height of an empty tree is -1.) Here's the BSTNode definition:

```
public class BSTNode<T extends Comparable<T>> {
   T data; BSTNode<T> left, right; int height;
   ...
}
```

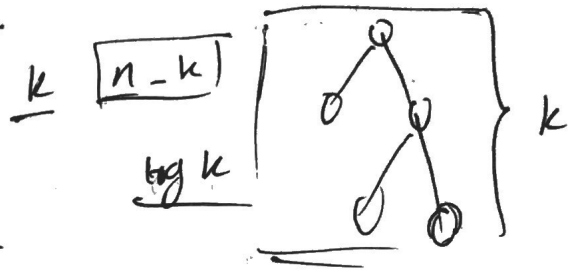
Complete the following RECURSIVE implementation to check if a BST is an AVL tree, assuming the height field of each node has been already filled. (You can use the `Math.abs(int i)` method to get the absolute value of an integer.) You may NOT use helper methods.

```
// Recursive implementation to check is a BST is an AVL tree
public static <T extends Comparable<T>>
boolean isAVL(BSTNode<T> root) {
   /** COMPLETE THIS METHOD **/
if (root == null) { return true; }

what if root.left == null
right

if (Math.abs(root.left.height - root.right.height) <=1) {

if(isAVL(root.left) == true && isAVL(root.right) == true) {

   return true;
} el
}
return false;
}
```

CS 112 Fall '17 Exam 2; netid: _____

b) An AVL tree is used to count the number of times each word (case INsensitive) occurs in a text file. Each node of the tree stores a word, with its count. The tree is ordered alphabetically by words. If there are k distinct words, and n words in all in the file, what would be the worst case big O time to store all the words with their counts in the tree? Count word comparisons (each is unit time) only. Ignore the time taken to read a word from input. Show your work.

1 2 3 4 5 6 7 8 9 10 - - -
W.C. AVL = log n



Worst Case: O(n log n)

The worst case would be if k was equal to n, meaning that every single word in the document is distinct. Additionally, the worst case for inserting into an AVL tree is log n because this means you must traverse to the lowest possible level in the tree. Doing this for every element in the document would be n log n.

2. **Huffman Coding (25 pts, 6+10+9)**

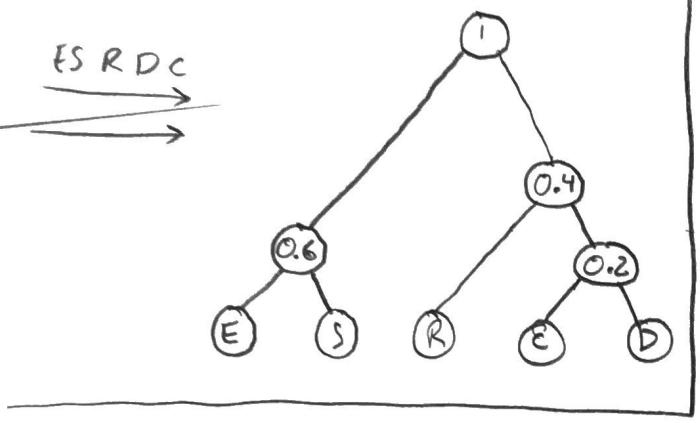
Given the following set of character-probability pairs:

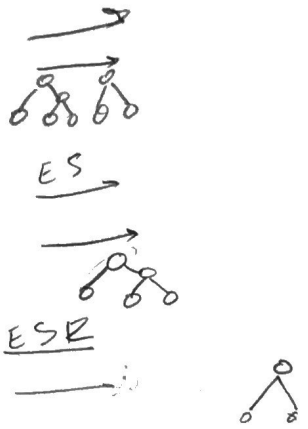
`(C,0.1), (D,0.1), (R,0.2), (S,0.3), (E,0.3)`

6

a) Show the final Huffman tree (you don't need to show any of the steps of the tree build process).

ESRDC





b) Assume that enqueue, dequeue, creating a leaf node, creating a new tree out of two subtrees, and picking the minimum of two probabilities all take unit time. Ignore the time for all other operations. How many total units of time (exact number, not big O) did it take to build your tree in part (a)? Show your work. (Assume all queues were initially empty.)

9

enqueue : 9

dequeue : 6 8

create leaf : 5

create new tree : 4

pick min. : 6 4

30

198:112 Fall '17 Exam 2; netid: _____

c) Given a document with 64 distinct characters, with equal number of occurences of each, what would be the ratio of the length of the Huffman coded document to the original, if each character in the input was coded in 8 bits? Show your analysis.

$2^6 = 64$ distinct characters
equal occurence

6 / 8

64 distinct characters, in a large document. Original input coded 8 bits for each character. Huffman is more efficient, only using 6 bits, per character.

9

3. **Assorted (25 pts;9+9+7)**

a) Suppose you insert 125 integer keys into a hash table with an initial capacity of 25 and a load factor threshold of 2. The hash code is the key itself, and the function key **mod** table_capacity is used to map a key to a table location. Derive the total units of time required to insert all keys, ONLY counting one unit of time to map the key to the table, and one unit of work to insert an entry into a linked list. Ignore all other operations (such as allocating array, etc.) Assume a rehash doubles the table capacity. Show your work.

keys 1 - 51 = 51 * 2 = 102

rehash

keys 1-51 (102 units) + 52-101 (100) units = 202

rehash

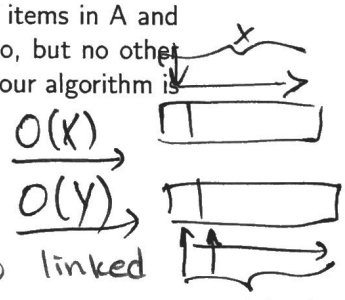
keys 1-51(102) + 52-101(100) + 102-125(48) = 250

Total : 250 + 202 + 102 = 554

4

b) Consider two binary search trees A and B. (They are NOT AVL trees, just plain BSTs.) BST A has x nodes, and BST B has y nodes. Assume there are no duplicates in the entire set of items in A and B. Describe the fastest algorithm to output the combined set of items in A and B in sorted (ascending) order. You can use additional array space if you need to, but no other data structure. Derive the worst case big O running time of your algorithm. (If your algorithm is not the fastest, it will get at most 4 points.)

O(x + y)



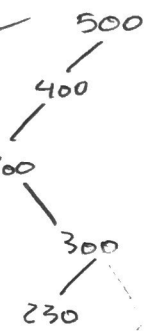
The worst cast situation would be two linked lists essentially, or completely skewed binary search trees. In this case, every single element of both "trees" would have to be checked while performing an inorder traversal on each tree.

Describe Algorithm ?

198:112 Fall '17 Exam 2; netid: _____

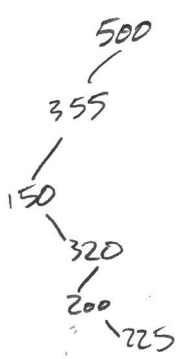
c) Suppose a BST (not AVL) stores some integers in the range 1 to 500. We will perform a search for the value 225, and keep a list of the values encountered on the search path through the BST. For each of the following sequences of values, say whether or not it is a possible search path. If yes, show the actual path (with branches), if not, explain why.

i) 500, 400, 200, 300, 230, 350, 225



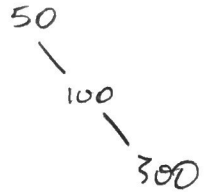
No - can't search for 350 after you're gone left of 300

ii) 500, 355, 150, 320, 200, 225



Yes

iii) 50, 100, 300, 40, 120



No - can't search for 40 after you're gone right of 50.