COMPUTER SCIENCE 112 - SPRING 2014
SECOND MIDTERM EXAM

Name: Alessandro Orsini

CIRCLE your recitation:

W 3:35 W 5:15 W 6:55 W 1:55 T 6:55 H 6:55 T 5:15 H 5:15

- Be sure your test has 4 questions.
- **DO NOT TEAR OFF THE SCRATCH PAGES OR REMOVE THE STAPLE.**
- Be sure to fill your name and circle your recitation time above, and your name in all subsequent pages where indicated.
- This is a CLOSED TEXT and CLOSED NOTES exam. You MAY NOT use calculators, cellphones, or any other electronic device during the exam.

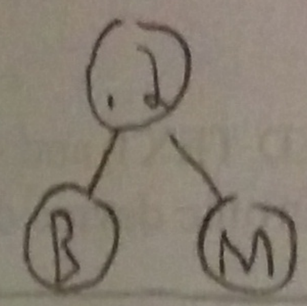
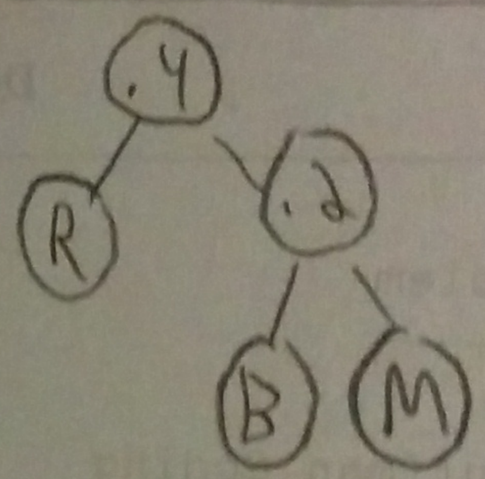
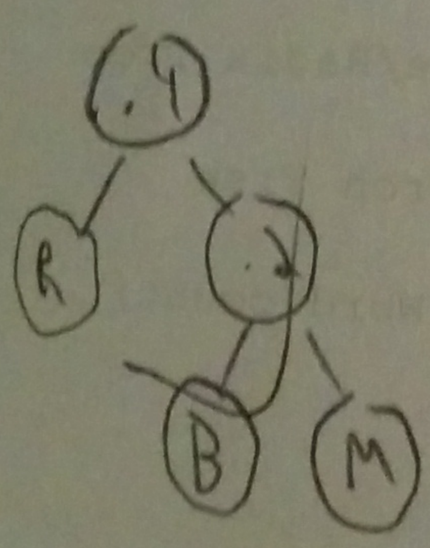
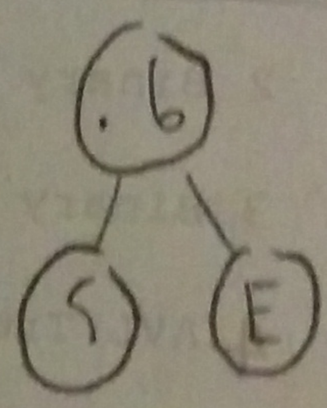
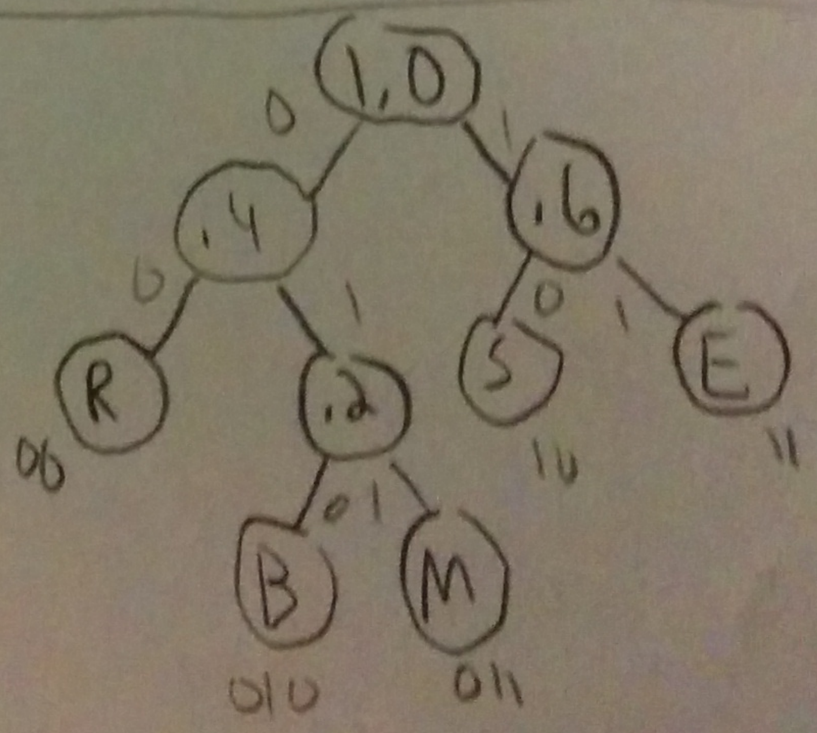
Do not write below this line

Problem	Max	Score
1 Huffman Coding	20	16 Yan
2 Binary Tree/Radix Tree	20	20 Jun
3 Binary Search Tree	15	14 Julio
4 AVL Tree (Word count)	20	8 slm
TOTAL	75	58

Given the following set of ...

(B,0.05), (M,0.15), (R,0.2), (S,0.25), (E,0.35)

(a) Build a Huffman tree for this character set. Fill in the following table to show the queue S, which contains the initial leaf nodes, and the queue T, which contains the subtrees as they are built, and eventually the completed Huffman tree. Each line shows the status of the queues at the end of that step. The last step should have a single tree in the queue T. Show each node as a circle with either a character or a probability value inside. (Ties in probability values are broken arbitrarily, and it doesn't matter which dequeue goes left and which goes right when building a subtree.)

Step	Queue S	Queue T
Step 1	B, M, R, S, E	Empty
Step 2	(R, .2), (S, .25), (E, .35)	
Step 3	(S, .25) (E, .35)	
Step 4		
Step 5		

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 units of time did it take in all to build your tree? Show your work.

Step 1: enqueue the 5 pairs = 5 units ✗

Step 2: dequeue 2 pairs, create 2 leaf nodes, create 1 tree, enqueue tree = 6 units ✗

Step 3: dequeue 2 pairs, compare 2nd pair and tree, enqueue 2nd pair, dequeue 1 create a leaf node, create a new tree, enqueue new tree = 8 units

Step 4: dequeue 2 pairs, dequeue 1 tree, compare tree and 2nd pair, enqueue tree, create 2 leaf nodes, make 1 tree, enqueue tree = 9 units

Step 5: dequeue 2 trees, create new tree, enqueue tree = 4 units ✓

Total work = 32 units

c) Suppose a Huffman coded bit string (with 1's and 0's only) is decoded into the character string "SERMBERSEMBERS", using the tree that was built in (a) above. Briefly describe the decoding process. Then derive the total units of time needed to complete the decoding process–specify what operation(s) you are counting towards time.

The decoding process starts at the top of each tree and goes left for a 0 and right for a 1 until it reaches a character. After saving the character it starts at the top of the tree again until there are no 1s or 0s left.

Count each left/right traversal as 1 unit time.

R, S, E take 2 units time each B, M take 3

	#	units time	total
R	3	2	6
S	3	2	6
E	4	2	8
B	2	3	6
M	2	3	6

198:112 Spring 2014 Midterm Exam 2; Name: _____

2. Binary Tree/Radix Tree (20 pts, 10+10)

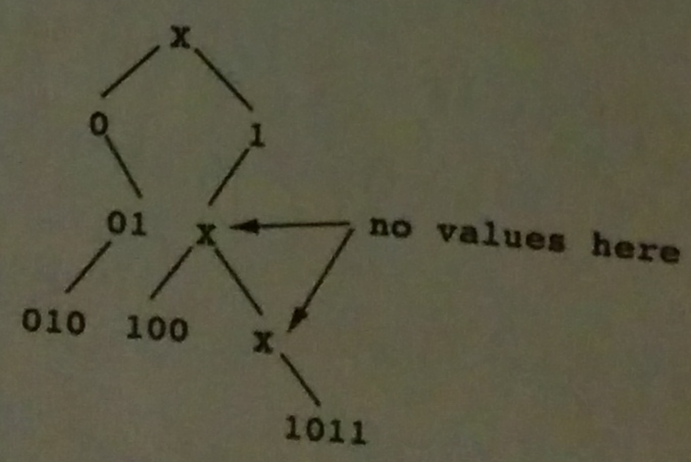
a) Given the preorder and inorder traversals of a binary tree with n nodes, what would be the worst case big O running time to build the tree? Count *only* comparisons between items in the preorder and inorder traversals towards the running time. Clearly show your derivation with explanation of the steps, otherwise you will not get any credit.

10

Worst case run time would be $O(n^2)$ ✓ 3

This is because you start with the preorder list and match it to the inorder list. Worst case it will go through the whole inorder list for each preorder item. In this case each additional item increases run time at a rate of $O(n^2)$ because it goes through n lists.

b) The *radix tree* data structure shown below stores the bit strings 0, 1, 01, 010, 100, and 1011 in such a way that each left branch represents a 0 and each right branch represents a 1.



To find if a bit string exists in this radix tree, start from the root, and scanning the bits of the string left to right, take a left turn if the bit is 0, and a right turn if the bit is 1. If a node can be reached using this sequence, and it does not contain an 'X', the bit string is found, else not found.

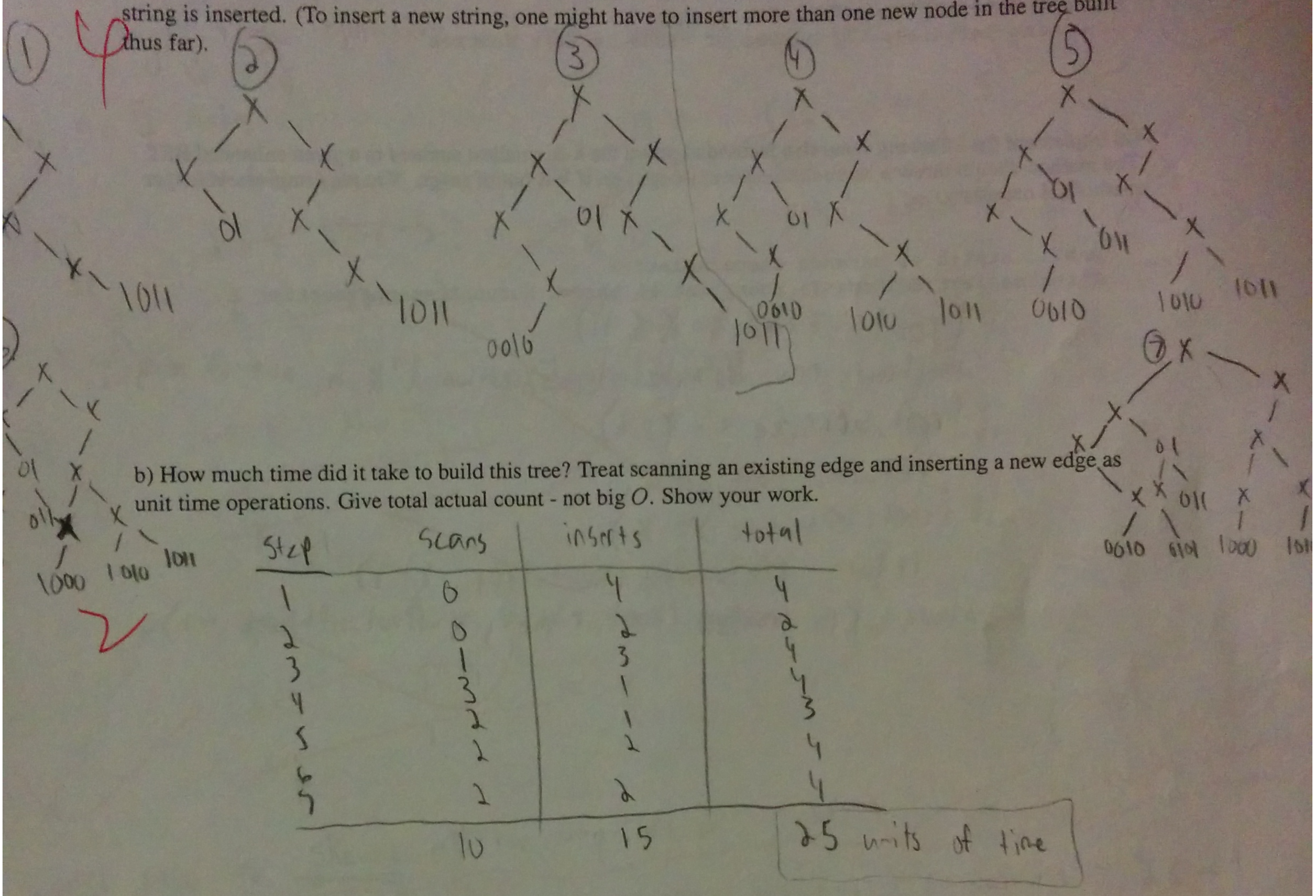
198:112 Spring 2014 Midterm Exam 2; Name: _____

a) Given the following strings:

1 2 3 4 5 6 7

1011, 01, 0010, 1010, 011, 1000, 0101

Starting with an empty tree, build a radix tree to store these strings, showing the radix tree after *each* new string is inserted. (To insert a new string, one might have to insert more than one new node in the tree built thus far).



b) How much time did it take to build this tree? Treat scanning an existing edge and inserting a new edge as unit time operations. Give total actual count - not big O. Show your work.

Step	scans	inserts	total
1	0	4	4
2	0	2	2
3	1	3	4
4	3	1	4
5	2	1	3
6	2	2	4
7	2	2	4
	10	15	25 units of time

c) How much time would it take to (1) insert one new binary string of k bits into a radix tree? (2) an arbitrary number of binary strings but whose total length is n bits? Again, total actual count, not big O. Show work.

① k units of time

② n units of time

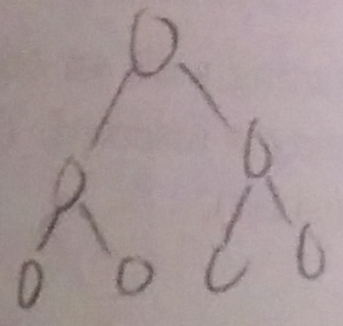
This is because whether you are inserting or scanning, it takes a unit to go down 1 level. Because of this each bit goes down another level and hence takes another unit of time.

3. Binary Search Tree (15 pts;8+7)

Suppose each binary search tree node is modified by adding a field that stores the number of nodes in its LEFT subtree. This modification is useful to answer the question: what is the k-th SMALLEST element in the binary search tree? ($k = 1$ means smallest element, $k = 2$ means second smallest element, etc.)

You are given the following enhanced BSTNode class definition:

```
public class BSTNode<T extends Comparable<T>> {
    T data; BSTNode<T> left, right;
    int leftSize;  // number of nodes in left subtree
    ...
}
```



(a) Implement the following **recursive** method to return the k-th smallest element in a given enhanced BST. The method should throw a `NoSuchElementException` if k is out of range. You may implement helper methods if necessary.

```
public static <T extends Comparable<T>>
T kthSmallest(BSTNode<T> root, int k) throws NoSuchElementException {
    if (root == null || k < 1)
        throw new NoSuchElementException("k is out of range");
    if (root.leftSize == k-1)
        return root.data;
    if (root.left.size >= k)
        return kthSmallest(root.left, k-1)
    return kth smallest(root.right, k - root.leftSize - 1);
}
```

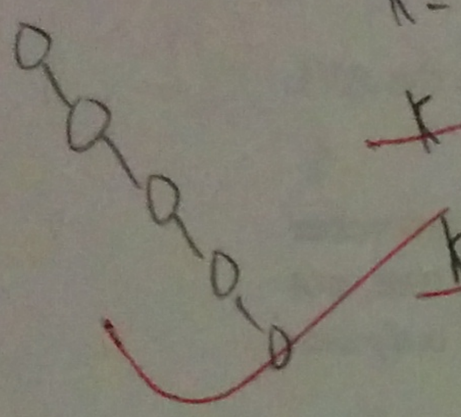
198:112 Spring 2014 Midterm Exam 2; Name: _____

6

(b) Compute the *most* number of units of running time (not big O) for your implementation, for a tree with n nodes. Draw <u>all</u> patterns of tree shape and k value for that shape, for which this occurs. State the basic unit time operations you are counting (ignore time to compare a pointer against null), show how many times each of these operations are done for each of the scenarios, then add up the operations to get a number that is a function in n and/or k.

unit time operations will be comparing <u>left+size == k-1</u>, <u>left+size ≥ k</u>,

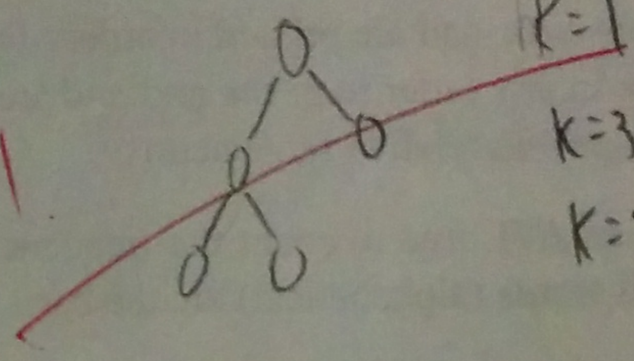
alike n: 5 nodes



k=1 time: 1 k=1

~~k=3 time: 5~~

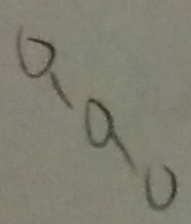
~~k=5 time: 9~~



k=1 time: 5

k=3 time: 3

k=5 time: 3

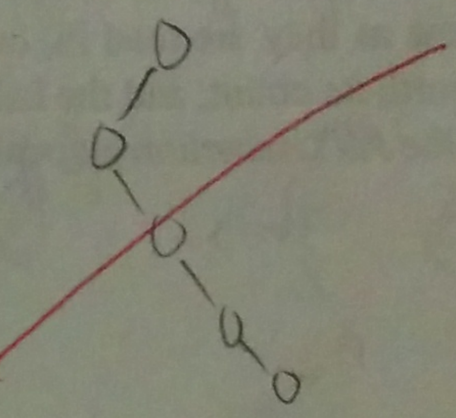


~~k=1 time: 1~~

~~k=2 time: 3~~

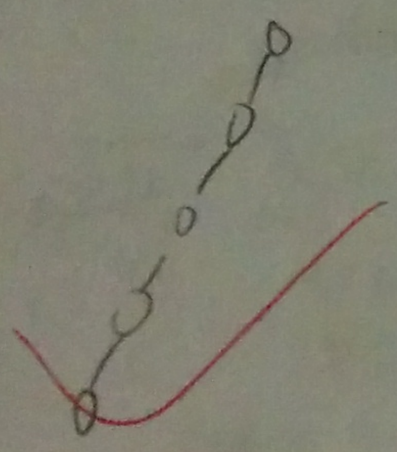
~~k=3 time: 5~~

k=n



k=1 time: 1

k=3 time:



k=1 9

~~k=3~~ 5

~~k=5~~ 1

k=1

worst case is <u>skewed</u> all the way left or right with k value

for right skew there are <u>k equality comparisons</u> and k-1 ≥ comparisons = 2k-1

for left skew there are $2n - \underset{k=1}{(2k-1)}$ comparisons

The worst cases for both of these are the same

4. Word Count (20 pts;6+7+7)

You have been asked to write a program to count the number of times each word occurs in a text file, and print them grouped according to word length. Below is a sample text file on the left, and the expected output of your program on the right:

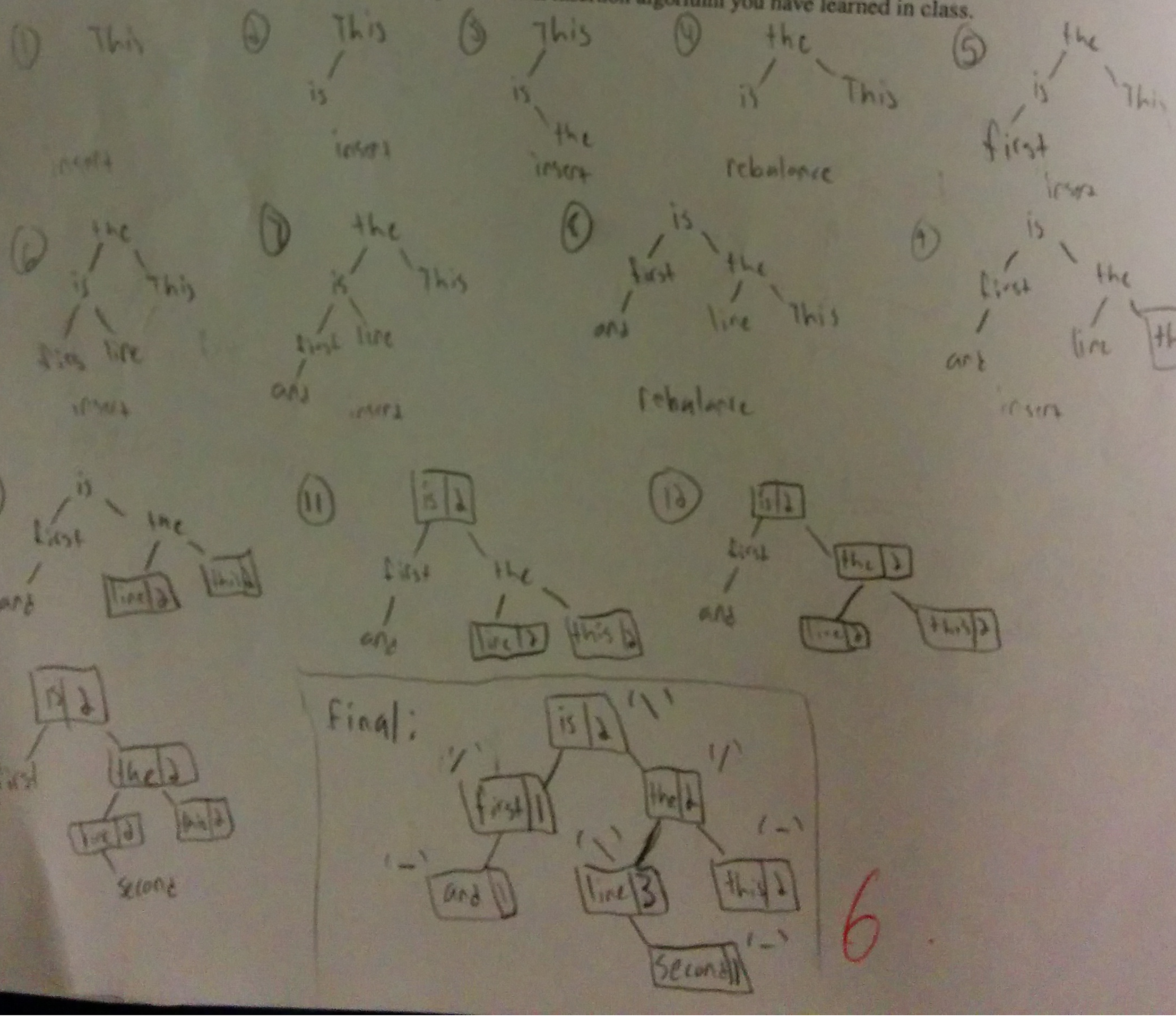
```
This is the first line,
and this line is the second line.
```

```
is: 2
the: 2
and: 1
this: 2
line: 3
first: 1
second: 1
```

Note that words are counted *ignoring case*, and are printed in order of word length. In any group of words of the same length, the printout can be in any order (e.g. *the* and *and* are both printed before words of greater length, but they need not be in any specific relative sequence.)

You implement the program using an AVL tree to count the frequencies of words. Each node in the AVL tree has a word (with its count), and words (alphabetical) are the basis of the AVL ordering.

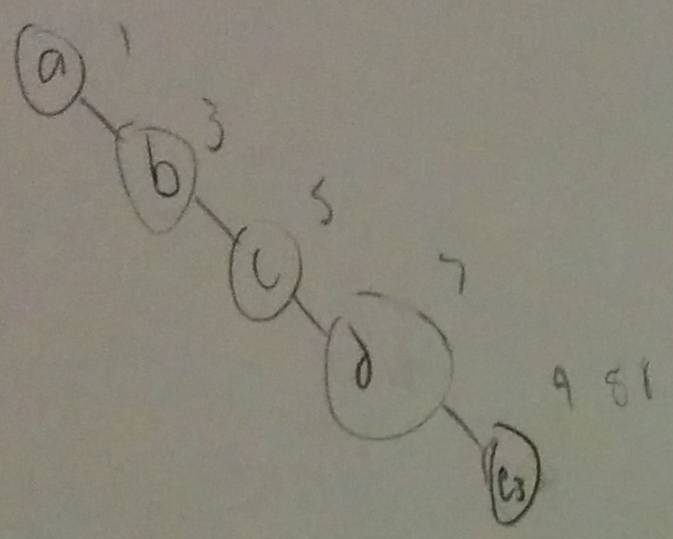
a) Show the final AVL tree after *all* the words from the sample input file have been inserted. (Remember that words will be inserted one at a time as they are read in, but you only need to show the final tree, not all the intermediate trees). Show the word, its count, and the balance factor at each node. There is *only one correct AVL tree answer*, according to the AVL insertion algorithm you have learned in class.



198:112 Spring 2014 Midterm Exam 2; Name: _____

b) If there are k distinct words, and n words in all (in the sample file, $k = 7$ and $n = 12$), how much time will it take in the worst case (big O) to store all the words with counts? Count word comparisons (each is unit time) only. Ignore the time taken to read a word from input.

worst case the words would be all stored one direction.
For example: a, b, c, d, e, e, e



1 unit for first insertion
+ 2(n-1) units for each subsequent insertion
- k units for each duplicate at end

Total: $2n - k - 1$

$O(n - k)$

c) Describe the most efficient algorithm and data structures you can think of, to print the output. Taking m to be the maximum word length, derive (show your work!) the worst case time big O running time of your algorithm. (You will get AT MOST HALF THE CREDIT if your algorithm is not the most efficient.)

Use a queue. For each word length, add all words in the tree of that length to your queue. Do this until word length > m.
Finally dequeue each item, printing the word and its count.

m: max word length n: # of words

going through each length: $O(m)$
traversing through tree: $O(k)$

total: $O(mk)$

$m \cdot O(k)$
$= O(mk)$.
$+ O(k)$.
$= O(mk)$.

2pts. slow.