CS340 Winter 1996
Ming Li

CS340 Midterm Exam, Feb. 14, 6-8pm, MC2037/2038

Your Name:
Student ID#:

Question	total marks	your marks
1.	16	
2.	10	
3.	5	
4.	5	
5.	20	
6	10	
7.	10	
8.	10	
9.	14	

Write your name and student number on every page.

Indicate clearly how you have obtained your answers, show all the necessary steps, but be concise.

No books or notes.

log means $\log_2$.

If necessary, use the back of the same page to continue with your answer.

Good luck.

1. (16 marks) Answer true or false. (2 marks each question)

   (a) Although a binary tree with n vertices has precisely $n-1$ edges, this is sometimes not true for high degree trees such as B-trees.

   False

   (b) $n + 2\sqrt{n}$ is $O(n^2)$.

   True

   (c) $n^2 + 3n + 4$ is $O(n^2)$.

   True

   (d) $n \log n$ is $O(n)$.

   False

   (e) If $f(n)$ is $O(n)$ and $g(n)$ is $O(n^2)$, then $f(n)g(n)$ is $O(n^3)$.

   True

   (f) For every natural number $n > 0$, it is possible to construct an AVL-tree with n vertices and with height at least $5 \log n$.

   False

   (g) All leaves of a B-tree are on the same level.

   True

   (h) The time complexity of inserting or deleting an element in an AVL tree of n nodes is $O(\log n)$ on average, but $\Omega(n)$ in the worst case.

   False.

2. (10 marks) Give short answers to the following questions.

   (a) The ordinary linked representation of a binary tree with n nodes has exactly $n+1$ unused (or nil) pointers. Why?

   (3 marks)

   Each node has 2 pointers, one to the left child, one to the right child. So, a binary tree with n nodes has 2n pointers.

   There are (n-1) used pointers because every node points to another node, except the root.

   Therefore, there are 2n- (n-1) = n+1 unused pointers.

   (b) How many leaves does a perfect binary tree of height d have?

   (3 marks)

   2^d leaves

   (c) Starting from an empty AVL-tree, insert 1, then insert 3, then insert 2. Draw the trees resulting from each insertion, and indicate whether a single or double rotation is necessary.

   (4 marks)

   A double rotation is necessary after inserting 3.

   The resulting AVL tree has 2 as its root.

   (Only 1 point is given if the student says "single rotation" even if the final tree is correct.)

3. (5 marks) Let $T(1) = 2$ and for all $n > 1$, $T(n) = T(n-1) + n - 1$. Solve $T(n)$.

```
[Sol'n 1] T(n) = T(n-1)+(n-1) = T(n-2)+(n-2)+(n-1) = ...      <1 mark>
               = T(1)+1+2+3+...+(n-2)+(n-1) = 2+(1/2)n(n-1)  <5 marks>
[Sol'n 2] T(n) = [T(n)-T(n-1)]+[T(n-1)-T(n-2)]+...+[T(2)-T(1)]+T(1)
               = (n-1)+(n-2)+...+1+T(1) = 2+(1/2)n(n-1)      <5 marks>
```

4. (5 marks) What is the value returned by the following function? Express your answer as a function of n.

```
function mystery(n)
    r:=0;
    for i:=1 to n-1 do
      for j:=i+1 to n do
        for k:=1 to j do
           r:=r+1;
    return(r);
```

```
The return value is:

r = SUM[i=1..n-1]{SUM[j=i+1..n]{SUM[k=1..j]{1}}}        <1 mark>
  = SUM[i=1..n-1]{SUM[j=i+1..n]{j}                      <2 marks>
  = SUM[i=1..n-1]{(1/2)(i+1+n)(n-(i+1)+1)}              <3 marks>
  = SUM[i=1..n-1]{(1/2)(n+i+1)(n-i)}
  = (1/2)SUM[i=1..n-1]{n^2+n-(i^2+i)}
  = (1/2)(SUM[i=1..n-1]{n^2+n} - SUM[i=1..n-1]{i^2}-SUM[i=1..n-1]{i})
                                                        <4 marks>
  = (1/2)[(n-1)(n^2+n) - (1/6)(n-1)n(2n-1) - (1/2)n(n-1)]
                                                        <5 marks>
  = (1/2)[(1/6)(6(n^3-n) - (2n^3-3n^2+n) - 3(n^2-n))]
  = (1/12)[6n^3-6n-2n^3+3n^2-n-3n^2+3n]
  = (1/12)[4n^3-4n]
  = (1/3)(n^3-n)
```

5. (20 marks) Consider this tree. Give short answers.

(a) What is the height of the tree?

(b) Is this tree a binary search tree?

(c) Is this tree an AVL tree?

(d) Is this tree a B-tree of order 3?

(e) What is the internal path length of this tree?

(f) What is the external path length of this tree?

(g) Assuming all elements in the tree are equally likely to be queried, what is the average number of comparisons in a successful search for this tree? In an unsuccessful search (assuming also it is equally likely to end up at any external node)?

(h) List the nodes of the tree according to inorder traversal.

(i) List the nodes of the tree according to preorder traversal.

(j) List the nodes of the tree according to postorder traversal.

```
Two marks for each question.
a)
the height of the tree is the maximum depth of a leaf which is 4
b)
yes, it's inorder, therefore a binary search tree
c)
no, it's not balanced, therefore not an AVL tree
d)
no, it doesn't have uniform depth, so it's not a B-tree
e)
internal path length is sum of depth over internal nodes,
which is 0+1+2+3+4 = 10
f)
```

```
external path length is sum of depth over leaves (one for each
nil pointer), which is 1+2+3+4+5+5 = 20
g)
average cost of succesful search is one plus internal path length
divided
by number of internal nodes, which is 1 + 10/5 = 3
average cost of unsuccesful search is external path length divided
by number of leaves, which is 20/6 = 3 1/3
h)
2 3 4 5 6
i)
6 2 5 3 4
j)
4 3 5 2 6
```

6. (10 marks) Given letters A,B,C,D,E,F,G with respective probabilities 0.001 (A), 0.002 (B), 0.03 (C), 0.04 (D), 0.1 (E), 0.3 (F), 0.527 (G). Construct the Huffman code for these letters using Huffman's algorithm. (Draw the tree.)

```
Merge nodes A and B to get a new node AB with frequency .003,
then merge AB with C to get a new node ABC with frequency .033,
then continue merging to get nodes ABCD with frequency .073,
ABCDE with frequency .173, ABCDEF with frequency .473, and
ABCDEFG with frequency 1.0.

The resulting codes are:

A - 000000
B - 000001
C - 00001
D - 0001
E - 001
F - 01
G - 1
```

7. (10 marks) Give an AVL-tree (try to give the smallest) for which the deletion of a node requires two double rotations. Draw the tree, and explain why two rotations are needed.

```
+-------+-----+
|       |     |
+--+    +--
|  |    |
        +
^       |
delete
```

8. (10 marks) Consider the following B-tree T, of order 3 (2-3 tree). Perform the following two operations (on 2-3 trees). For each operation, use the **original** tree T. You must draw the tree after each operation and indicate how you have obtained the result.

(a) Insert 100 into T.

(b) Delete 15 from T (use the original T).

```
(a) 100 is added to the 45-55 node to give a 45-55-100 node,
    which overflows the node.  The node is split, and 55 is
    sent to the parent, giving a 30-40-55 node.  This is in
    turn split, and 40 is sent to the parent to give a 20-40
    node, which is a 3-node so the algorithm terminates.

(b) After 15 is deleted, 5 and 10 are consolidated.  This
    causes underflow in the parent, so the 30 key and its
    child 25 are borrowed from the parent's sibling.  20
    becomes the index of the parent of the 5-10 node, 25
    becomes the right child, and 30 becomes the index of
    the root.
```

9. (14 marks) Give an efficient algorithm which finds the maximum element and the minimum element in an input (unsorted) array $A[1..n]$. You must try to use the smallest number of comparisons.

Clearly and *briefly* describe your algorithm (not the program) first. Then give its number of comparisons in terms of the size of the array. Note, do not use the big-O notation for this problem. Precisely count the number of comparisons your algorithm uses in the worst case. Hint: There are two ways of doing this. One way is the divide and conquer method. You can assume that the array size n is a power of 2.

———

(1) First partition A into $n/2$ pairs and make $n/2$ comparisons.

(2) Choose MIN from the $n/2$ "losers" in (1).

(3) Choose MAX from the $n/2$ "winners" in (1).

Time complexity: (1) takes $n/2$ comparisons. (2) and (3) each takes $n/2 - 1$ comparisons. So in total $3n/2 - 2$.