Computer Science 3711
Winter 2004

Midterm Exam

February 17, 2004

Instructor:
T. Wareham

NAME: ______________________ STUDENT ID #: ______________

- This exam is out of 60 marks.
- This exam has 7 pages (including this cover page). There are 5 questions, most of which have multiple parts.
- Please answer all questions in the space provided on this exam; if you find it necessary to continue an answer on the back of a sheet of paper, that is fine, but please make a note on the front side, *e.g.*, "answer cont'd on back".

Question		Mark
1.	9	
2.	12	
3.	9	
4.	18	
5.	12	
	60	

1. (9 marks) Circle the letters associated with the appropriate answers in the following multiple choice questions. Note that some of these questions may have more than one answer whose letter needs to be circled.

(a) (3 marks) Which of the following is true of the function $T(n) = 2^{n-4}$?

a) $T(n)$ is $\Theta(2^{n+3})$ b) $T(n)$ is $\Omega(n^{1000})$ c) $T(n)$ is $O(2^{n-10})$

(b) (3 marks) Which of the following is true of the function $T(n) = 9^{\log_3 n}$?

a) $T(n)$ is $O(2^{\frac{n}{64}})$ b) $T(n)$ is $\Theta(3^{\log_9 n})$ c) $T(n)$ is $\Omega(n \log_2 n^2)$

(c) (3 marks) Which of the following is true of the function $T(n) = n^2$?

a) $T(n)$ is $\Theta((n-5)^3)$ b) $T(n)$ is $O(57 \log_2 n)$ c) $T(n)$ is $\Omega((n+4)^2)$

2. (12 marks)

a) (4 marks) Give the recurrence for the time complexity of the following recursive algorithm:

```
procedure FUNKY-REC(m, n)
    if (m < 1)
        sum = 0
        for (i = 1; i <= n; i++)
            sum = sum + (FUNKY-ITR(m) / i) - j
        return(sum)
    else if (m == 1)
        sum = 10
        for (i = 1; i <= m; i++)
            sum = sum + FUNKY-ITR(n) / i * j
        return(sum)
    else if (m > 1)
        sum = 0
        for i = 1 to n * n do
            l = FUNKY-REC(m/5, n)
            sum = FUNKY-ITR([square root of n]) * l
        return(sum)
    else
        return(m)
```

Assume that procedure `FUNKY-ITR(x)` runs in $O(x^4)$ time.

b) (8 marks) Derive an upper bound for $T(n)$ using any method discussed in class or in the textbook, where

$$T(n) = \begin{cases} 0 & n \leq 3 \\ T(n-4) + cn & n > 1 \end{cases}$$

3. (9 marks) Consider the following algorithm:

```
sum = 0;
for i = 1 to n * n do
    j = T1(n)
    if (T2(i, j, n)
        for k = 1 to n do
            if ((k % 2) == 0)
                l = T3(n, k) - 2
            sum = (sum * j / l)
    sum = sum * T3(n)
if ((sum % 2) == 0)
    sum = sum / 2
else
    sum = sum * 2
```

a) (**3 marks**) Give the parameterized asymptotic worst-case time complexity for this algorithm. Please include a term for how many times T2 evaluates to `true` in your expression.

b) (**3 marks**) Give the asymptotic worst-case time complexity of this algorithm when the T1, T2, and T3 operations require $O(1)$, $O(n^2)$, and $O(1)$ time, respectively, and we don't know how many times T2 evaluates to `true`.

c) (**3 marks**) Give the asymptotic worst-case time complexity of this algorithm when the T1, T2, and T3 operations require $O(n^3)$, $O(\log_2 n)$, and $O(n^2)$ time, respectively, and we know that T2 evaluates to `true` $O(n \log_2 n)$ times.

4. (18 marks)

a) (4 marks) How are problems that have only combinatorial solution-space tree (CST) algorithms different from problems that have divide-and-conquer algorithms? Phrase your answer in terms of the problem properties given in class.

b) (14 marks) Consider the following algorithm:

```
DFS-V(sol, C, num)
    if (SIZE(sol) > num)
        print("pruned: decryption uses too many codewords")
    else if (ISCOMPLETE(sol))
        if (SIZE(sol) == num)
            print(sol)
    else
        for (i = 1; i <= NUMCODEWORDS(C); i++)
            if (CANEXTEND(sol, C, i)
                solc = EXTENDCOPY(sol, C, i)
                DFS-V(solc, C, num)
```

The problem solved here prints all decryption-solutions for a ciphertex T relative to a codeword-dictionary C that have num **distinct** codewords in the decryption-sequence (note that this is different from the problem solved in Assignments #2 and 3, in which we were interested in the **total** number of codewords in the decryption-sequence). In this algorithm, sol is a (possibly partial) decryption of T relative to C, SIZE(sol) returns the number of distinct codewords in the decryption-sequence in sol, CANEXTEND(sol, C, i) determines whether or not the decryption in sol can be extended by the codeword indexed i in C, and EXTENDCOPY(sol, C, i) returns a copy of sol in which the codeword indexed i in C has been added to the decryption-sequence. To answer this question, sketch the implicit tree of solution-nodes generated by the algorithm above (marking leaf-nodes which are printed by a circle and nodes that are pruned with a square) with the call DFS-V(sol, C, num) when sol is the empty decryption, $T = baaba$ and C and num have the following values:

i) (7 marks) $C = \{a, b, ab, ba, baa\}$ and $num = 3$:

ii) (7 marks) $C = \{b, aa, ab, ba, aba, aab, baa\}$ and $num = 2$:

5. (12 marks) The length of the longest weighted common subsequence (LWCS) of two strings s and s' is defined by the following recurrence:

$$D(i,j) = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ \max(D(i-1,j-1) + COST(s(i),s'(j)), & \\ \qquad D(i,j-1) - 2, & \\ \qquad D(i-1,j) - 2) & \text{otherwise} \end{cases}$$

For any two symbols x and y, function $COST(x,y)$ returns 3 if $x = y$, 1 if either (i) x and y are both vowels or (ii) x and y are both consonants, and -1 otherwise (recall that a vowel is one of the letters $\{'A','E','I','O','U'\}$ and a consonant is any letter of the alphabet that is not a vowel). Given the above, determine the longest weighted common subsequence of the strings **AUQIF** and **OUF** – that is, fill in the dynamic programming matrix given below (including one backpointer per matrix-cell), show one of the backpointer paths that gives an optimal LWCS (given that traceback starts at the lower right-hand corner matrix-cell, *i.e.*, $D(s|,|s'|)$, and goes back to a matrix cell in row 0 or column 0), and show the LWCS corresponding to this path, *i.e.*, the sequence of matching symbol-pairs induced by the recursive subclause $\max(D(i-1,j-1) + COST(s(i),s'(j))$ and its associated diagonal backpointers.

			A	U	Q	I	F
	i \ j	0	1	2	3	4	5
	0						
O	1						
U	2						
F	3						