University of New Mexico
Department of Computer Science

First Midterm Examination

CS 461 Data Structures and Algorithms
Fall, 2003

Name:	
Email:	

- Print your name and email, *neatly* in the space provided above; print your name at the upper right corner of *every* page. Please print legibly.
- This is an *closed book* exam. You are permitted to use *only* two pages of "cheat sheets" that you have brought to the exam and a calculator. *Nothing else is permitted.*
- Do all five problems in this booklet. *Show your work!* You will not get partial credit if we cannot figure out how you arrived at your answer.
- Write your answers in the space provided for the corresponding problem. Let us know if you need more paper.
- Don't spend too much time on any single problem. The questions are weighted equally. If you get stuck, move on to something else and come back later.
- If any question is unclear, ask us for clarification.

Question	Points	Score	Grader
1	20		
2	20		
3	20		
4	20		
5	20		
Total	100		

1. **Short Answer**

Multiple Choice Questions:

True or False: (circle one, 2 points each)

(a) **True or False**: $n \log n$ is $\Omega(n)$.*Solution: True*

(b) **True or False**: The number of possible subsets of n elements is 2^n. *Solution: True*

(c) **True or False**: The greedy algorithm for $0-1$ knapsack always picks a set of items which has the maximum allowable weight. *Solution: False. Consider the example we went over in class*

(d) **True or False**: The greedy algorithm for fractional knapsack always picks a set of fractions of items which has the maximum allowable weight. *Solution: True*

(e) **True or False**: The greedy algorithm for the activity selection problem always chooses the activity of shortest length. *Solution: False*

(f) **True or False**: If X and Y are sequences that both contain the character a, then some longest common subsequence of X and Y contains the character a. *Solution: False. Consider the strings $X = a\sigma$ and $Y = \sigma a$, where σ is a very long string containing no a's. The $LCS(X, Y) = \sigma$*

Short Answer: Justify your answer briefly (8 points total). **Circle your final answer.**

(a) Solve the following recurrence using the master method. $T(1) = 1$, $T(n) = 2T(n/2) + 1$
Solution: $f(n) = 1$ and $af(n/b) = 2$, so $af(n/b) \geq cf(n)$ for $c > 1$. Thus the leaf nodes dominate the recurrence. The total cost of the leaf nodes is $2^{\log n} = n$, so $T(n) = \Theta(n)$

2. **Annihilators**

Consider the following function:

```
int f (int n){
  if (n==0) return 0;
  else if (n==1) return 1;
  else{
    int val = 3*f (n-1);
    val = val - 2*f (n-2);
    val += 1;
    return val;
  }
}
```

(a) Let $f(n)$ be the value returned by the function f when given input n. Write a recurrence relation for $f(n)$

Solution: $f(n) = 3f(n-1) - 2f(n-2) + 1$

(b) Now give the general form for the solution for $f(n)$ using annihilators. *You need not solve for the constants. Solution: First we annihilate the homogeneous part,* $f(n) = 3f(n-1) - 2f(n-2)$. *Let* $F_n = f(n)$, *and* $F = \langle F_n \rangle$. *Then*

$$\begin{aligned} F &= \langle F_n \rangle & (1)\\ \boldsymbol{L}F &= \langle F_{n+1} \rangle & (2)\\ \boldsymbol{L}^2F &= \langle F_{n+2} \rangle & (3) \end{aligned}$$

Since $\langle F_{n+2} \rangle = \langle 3F_{n+1} - 2F_n \rangle$, *we know that* $\boldsymbol{L}^2F - 3\boldsymbol{L}F + 2F = \langle 0 \rangle$, *and thus* $\boldsymbol{L}^2 - 3\boldsymbol{L} + 2 = (\boldsymbol{L}-2)(\boldsymbol{L}-1)$ *annihilates* F.
Now we must annihilate the nonhomogeneous part $f(n) = 1$. *It's not hard to see that* $\boldsymbol{L} - 1$ *annihilates this nonhomogeneous part. So the annihilator for the entire function* $f(n) = 3f(n-1) - 2f(n-2) + 1$ *is* $(\boldsymbol{L}-1)^2(\boldsymbol{L}-2)$. *Looking this up in the lookup table, we see that* $f(n)$ *is of the form:*

$$\begin{aligned} f(n) &= c_1 2^n + (c_2 n + c_3)1^n & (4)\\ f(n) &= c_1 2^n + c_2 n + c_3 & (5) \end{aligned}$$

2. **Annihilators, continued.**

3. **Asymptotic Notation**

Prove that $\sqrt{n} = O(\frac{n}{\log n})$.

Solution: Goal: Give c *and* n_0 *such that* $\sqrt{n} \leq c\frac{n}{\log n}$ *for all* $n \geq n_0$*. The inequality we want then is:*

$$\sqrt{n} \leq c\frac{n}{\log n} \tag{6}$$

$$\sqrt{n} * \frac{\log n}{n} \leq c \tag{7}$$

$$\frac{\log n}{\sqrt{n}} \leq c \tag{8}$$

$$\tag{9}$$

So for $c = 1$ *and* $n_0 = 1$*, it's the case that* $\sqrt{n} \leq c\frac{n}{\log n}$ *for all* $n \geq n_0$

4. **Substitution Method**
Consider the following recurrence:

$$T(n) = 2^{2-n} * T(n-1) * T(n-1)$$

where $T(1) = 2$.

Show that $T(n) = 2^n$ by induction. Include the following in your proof: 1)the base case(s) 2)the inductive hypothesis and 3)the inductive step.

Solution: Base Case: $T(1) = 2$ *which is in fact* 2^1.
Inductive Hypothesis: For all $j < n$, $T(j) = 2^j$
Inductive Step: We must show that $T(n) = 2^n$, *assuming the inductive hypothesis.*

$$\begin{aligned} T(n) &= 2^{2-n} * T(n-1) * T(n-1) && (10) \\ T(n) &= 2^{2-n} * 2^{n-1} * 2^{n-1} && (11) \\ T(n) &= 2^n && (12) \end{aligned}$$

where the inductive hypothesis allows us to make the replacements in the second step.

5. **Dynamic Programming**

Recall that in the string alignment problem, we want to align two strings so as to minimize the cost of the alignment. In the original variant of the problem, the cost of an alignment was defined to be the number of columns which contain a blank plus the number of columns that have two different characters in them. For example, for the two strings FOOD and MONEY, the following alignment has a cost of 4.

F	O	O		D
M	O	N	E	Y

Consider a new variant of the string alignment problem where *the cost of an alignment is defined to be the number of columns which contain a blank plus* 10 *times the number of columns that have two different characters in them.* For example in this new variant, the alignment above would have a cost of 31 instead of 4.

(a) The recurrence relation for the minimal cost of aligning two strings A and B in the original variant of the string alignment problem is given in the formula below. Recall that $E(i,j)$ is the minimal cost of aligning $A[0..i]$ and $B[0..j]$. Give the modifications needed to get a recurrence relation for the minimal cost in the new variant of the problem. *To do this, you will need to change a single number in the formula below.* Please cross out the value to change and write the new value next to the crossed out one.

$$\begin{aligned} E(0,j) &= j \text{ for all } j, \\ E(i,0) &= i \text{ for all } i \\ E(i,j) &= \min \left\{ \begin{array}{l} E(i-1,j)+1, \\ E(i,j-1)+1, \\ E(i-1,j-1) + \left\{ \begin{array}{l} 0 \text{ if } A[i]=B[j] \\ 1 \text{ if } A[i] \neq B[j] \end{array} \right\} \end{array} \right\} \end{aligned}$$

(b) Now use this new recurrence to find the minimum alignment cost under this new variant for the two strings *ab* and *cb*. Do this by filling in the nine entries in the following dynamic programming table. Also include the arrows used to reconstruct a minimal solution. To the right of the table, give an alignment which achieves the minimal cost.

		a	b
c			
b			

Solution: In the very last line, need to change the 1 *to a* 10.

$$\begin{tikzcd}
 & & a & b \\
 & 0 \arrow[r] \arrow[d] & 1 \arrow[r] \arrow[d] & 2 \arrow[d] \\
c & 1 \arrow[r] \arrow[d] & 2 \arrow[r] \arrow[d] \arrow[dr] & 3 \\
b & 2 & 2 & 2
\end{tikzcd}$$

Alignment:

a	-	b
-	c	b