King Fahd University of Petroleum and Minerals

College of Computer Science and Engineering
Information and Computer Science Department

ICS 353: Design and Analysis of Algorithms
Spring 2006-2007
Major Exam 2, Monday April 23, 2007.

Name: Possible Solutions

ID#:

Instructions:

1. This exam consists of 8 pages, including this page, containing 4 questions.
2. You have to answer all 4 questions.
3. The exam is closed book and closed notes. No calculators or any helping aides are allowed. Make sure you turn off your mobile phone and keep it in your pocket if you have one.
4. The questions are equally weighed.
5. The maximum number of points for this exam is 150.
6. You have exactly 90 minutes to finish the exam.
7. Make sure your answers are readable.
8. If there is no space on the front of the page, feel free to use the back of the page. Make sure you indicate this in order for me not to miss grading it.

Question	Max Points	Points
1	25	
2	30	
3	45	
4	50	
Total	150	

Some Useful Formulas:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}, \quad \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=1}^{n} i^3 = \left(\frac{n(n+1)}{2}\right)^2$$

$$\sum_{i=0}^{n} a^i = \frac{a^{n+1}-1}{a-1} \text{ where } a \neq 1, \quad \log\frac{a}{b} = \log a - \log b, \quad \log_b a = \frac{\ln a}{\ln b}$$

1. (25 points) Find asymptotically tight bound, in terms of Θ, for the following function:

$$f(n) = \begin{cases} 1 & n \le 3 \\ 4f(\sqrt{n}) + \log n & n \ge 4 \end{cases}$$

$f(n) = 4f(\sqrt{n}) + \log n$.

Using the change of variable method:

Let $n = 2^k$, i.e. $k = \log n$. +4

$f(2^k) = 4f(2^{k/2}) + k$. "change of domain"

Now, Let $S(k) = f(2^k)$ +4

$S(k) = 4S(k/2) + k$ ...... ⊛

Solving ⊛ using Master Method: +4

$a = 4$, $b = 2$, $k^{\log_2 4} = k^2$ +2

$g(k) = k = O(k^{2-\epsilon})$ for $\epsilon = \frac{1}{2}$.

∴ case 1 of the master thm applies +6

∴ $S(k) = \Theta(k^2)$.

∴ changing back to the domain of f: +5

$f(n) = \Theta(\log^2 n)$.

−12: Using expansion without getting the solution.

2. (30 points) Algorithms based on Induction
   a. (15 points) Use Radix Sort to sort the following *octal* numbers showing all the intermediate steps (a final answer without the intermediate steps is worth zero points):

      30 , 216 , 5 , 44 , 111 , 42 , 236 , 17 , 523 , 7

I L: 030, 216, 005, 044, 111, 042, 236, 017, 523, 007

$k=1$

L_0: 030
L_1: 111
L_2: 042
L_3: 523
L_4: 044
L_5: 005
L_6: 216, 236
L_7: 017, 007

×5

II L: 030, 111, 042, 523, 044, 005, 216, 236, 017, 007

$k=2$

L_0: 005, 007
L_1: 111, 216, 017
L_2: 523,
L_3: 030, 236
L_4: 042, 044
L_5:
L_6:
L_7:

×5

III L: 005, 007, 111, 216, 017, 523, 030, 236, 042, 044

$k=3$

L_0: 005, 007, 017, 030, 042, 044
L_1: 111
L_2: 216, 236
L_3:
L_4:
L_5: 523
L_6:
L_7:

×5

Final L: 5, 7, 17, 30, 42, 44, 111, 216, 236, 523.

b. (15 points) Apply the exponentiation algorithm (whether the iterative or the recursive) to compute 2^{13}.

$exp(2,13) = exp(2,6) * exp(2*6) * 2$ 3

$64 * 64 * 2 = 64 * 128 = 8192$

$exp(2,6) = exp(2,3) * exp(2*3)$ 4

$8 * 8 = 64$

$exp(2,3) = exp(2,1) * exp(2,1) * 2$ 4

$2 * 2 * 2 = 8$

$exp(2,1) = exp(2,0) * exp(2,0) * 2$ 4

$1 * 1 * 2 = 2$

$exp(2,0) = 1$

$13 = (1101)_2$ 5

1101 : $exp = 1*1 = 1$; $exp = 1*2 = 2$

1101 : $exp = 2*2 = 4$; $exp = 4*2 = 8$ 10

1101 : $exp = 8*8 = 64$

1101 : ~~exp = 32 * 32 = 1024; exp = 1024 * 2 = 2048~~

$exp = 64 * 64 = 4096$; $exp = 4096 * 2 = 8192$

3. (45 points) Dynamic Programming
   a. (10 points) Write the recursive solution to the longest common subsequence problem
   b. (25 points) Use dynamic programming to find the length of the longest common subsequence and a longest common subsequence of the two strings: *xyxyxxxzx* and *zxxyzxzy*.
   c. (10 points) Is the longest common subsequence unique in this case? If yes, justify your answer. If no, give two different longest common subsequences.

a. $L(i,j) = \emptyset$ $\quad i=0$ or $j=0$ — 2

$= 1 + L[i-1, j-1] \quad i>0, j>0, A[i] = B[j]$. — 4

$= \max\{L[i-1,j], L[i,j-1]\} \quad i>0, j>0, A[i] \neq B[j]$ 4

b.

		x	y	x	y	x	x	x	z	x	
	0	0	0	0	0	0	0	0	0	0	+ 2
z	0	←0	←0	←0	←0	←0	←0	←0	↖1	←1	+ 2
x	0	↖1	←1	↖1	←1	↖1	↖1	↖1	←1	↖2	+ 2
x	0	↖1	←1	↖2	←2	↖2	↖2	↖2	←2	↖2	+ 2
y	0	↑1	↖2	←2	↖3	←3	←3	←3	←3	←3	+ 2
z	0	↑1	↑2	←2	↑3	←3	←3	←3	↖4	←4	+ 2
x	0	↖1	↑2	↖3	←3	↖4	↖4	↖4	←4	↖5	+ 2
z	0	↑1	↑2	↑3	←3	↑4	←4	←4	↖5	5←	+ 2
y	0	↑1	↖2	↑3	↖4	←4	←4	←4	↑5	←5	+ 2

\+ 2

Length = 5

LCS = X X y X Z $\quad$ + 5

c. No: LCS_1 : X X Y X Z

LCS_2 : X X y z x

6

4

4. (50 points) Divide and Conquer Algorithms

   a. (15 points) The first recurrence equation describing the running time of the SELECT algorithm was given by $T(n) = T(\lfloor n/5 \rfloor) + T(0.7n + 1.2) + cn$. Explain where each term in this equation came from?

1. $T(\lfloor \frac{n}{5} \rfloor)$: Cost of finding the median of Medians after dividing the array into $\lfloor \frac{n}{5} \rfloor$ gps of 5 elts each.

5

2. $T(0.7n+1.2)$: Worst case call to select on A_1 or A_3 where $0.7n+1.2$ is the maximum # of elts in A_1 or A_3.

5

3. cn : The cost of the rest of the operations in SELECT including partitioning to find A_1, A_2, A_3 & sorting the $\lfloor \frac{n}{5} \rfloor$ gps.

5

   b. (10 points) Explain when does the worst case running time of Algorithm QuickSort occur and derive its running time complexity in $\Theta()$ notation?

It occurs when the array is sorted in ascending order.

5

$T(n) = (n-1) + T(n-1) + T(\phi) = (n-1) + T(n-1)$

where T represents the # of elt comparisons

3

$= (n-1) + (n-2) + \cdots + 3 + 2 + 1$

$= \sum_{i=1}^{n-1} i = \frac{(n-1)n}{2} = \Theta(n^2).$ 2

c. (15 points) Modify the algorithm QuickSort such that it will always exhibit a worst case running time of $\Theta(n \log n)$ using the same divide and conquer idea. (i.e. the worst case running time in part a will never occur in the modified algorithm.)

```
ModQuickSort ( A, low, high)
    if  low < high  {
        m = SELECT ( A, low, high, ⌈(low+high)/2⌉ );
        Find elt m & place it in A[low] by
        Swapping it with A[low] if necessary.
        w = Split ( A, low, high);
        ModQuickSort (A, low, w-1);
        ModQuickSort (A, w+1, high);
    }
```

Here, we will always choose the median as a pivot, hence ensuring "best case" performance.

No answer or exactly same algorithm: −15

Answer not using Select Algorithm: −13

d. (10 points) If you were working as a mathematician and you needed to sort an array of real numbers with a very large size, (say more than 1 million entries), would you use the original QuickSort algorithm or the modified QuickSort algorithm? Justify your answer.

If the input doesn't come "sorted", I prefer the original since the SELECT has a hidden cost that may negatively affect the performance.