Banner number: Name:

Final Exam

CSCI 3110: Design and Analysis of Algorithms

December 13, 2007

Group 1		Group 2		Group 3		$\sum$
Question 1.1		Question 2.1		Question 3.1 ☐		
Question 1.2		Question 2.2		Question 3.2 ☐		
Question 1.3		Question 2.3		Question 3.3 ☐		
$\sum$		$\sum$		$\sum$		

Instructions:

- The questions are divided into three groups. You have to answer **all questions in Groups 1 and 2** and **exactly two questions in Group 3**. In the above table, put check marks in the **small** boxes beside the questions in Group 3 you want me to mark. I will not mark unmarked questions. If you mark all three questions, I will choose which two to mark at random.
- Provide your answer in the box after each question. If you absolutely need extra space, use the backs of the pages; but try to avoid it. Keey your answers short and to the point.
- **You are not allowed to use a cheat sheet.**
- If you are asked to design an algorithm and you cannot design one that achieves the desired running time, design a slower algorithm that is correct. A correct and slow algorithm earns you 50% of the marks for the algorithm. A fast and incorrect algorithm earns 0 marks.
- When designing an algorithm, you are allowed to use algorithms and data structures you learned in class as black boxes, without explaining how they work, as long as these algorithms and data structures do not directly answer the questions.
- **Read every question carefully before answering. In particular, do not waste time on an analysis if none is asked for, and do not forget to provide one if it is required.**
- **Do not forget to write your banner number and name on the top of this page.**
- **This exam has 13 pages, including this title page. Notify me immediately if your copy has fewer than 13 pages.**

Question 1.1 (Algorithm design paradigms) 10 marks

Name three algorithm design paradigms and explain for each of them what characterizes an algorithm that is based on this paradigm. At least one of the paradigms you describe should apply to solving optimization problems.

1.

2.

3.

State which of the above paradigms can be used to solve optimization problems. (If you named more than one such paradigm above, list all of them here.)

Question 1.2 (Randomization) **10 marks**

Define the following terms:

Worst-case running time of a deterministic algorithm

Average-case running time of a deterministic algorithm

What is the difference between the average-case running time of a deterministic algorithm and the expected running time of a randomized algorithm?

What is a possible advantage of a randomized algorithm for a given problem compared to a deterministic algorithm whose worst-case running time is *the same* as the expected running time of the randomized algorithm?

Question 1.3 (Computational complexity) 10 marks

What is the difference between a decision problem and an optimization problem?

Define the complexity class P.

Define the complexity class NP.

Define the term *NP-hard problem* and *NP-complete problem*.

In the following table, put a check mark in row x and column y if problem y belongs to complexity class x.

	Sorting	Hamiltonian cycle	Is there a path of length at most k from x to y in graph G?
P			
NP			

Question 2.1 (Correctness proof) 15 marks

Just as a reminder, here is Dijkstra's algorithm:

```
DIJKSTRA($G, s$)
1  mark every vertex $v \in G$ as unexplored
2  $d(v) \leftarrow +\infty$, for all $v \in G$
3  $d(s) \leftarrow 0$
4  insert every vertex $v \in G$ into a priority queue $Q$, with priority $d(v)$
5  while $Q \neq \emptyset$
6       do $v \leftarrow$ DELETEMIN($Q$)
7          mark $v$ as explored
8          for every out-neighbor $w$ of $G$
9          if $d(w) > d(v) + w(vw)$
10           then $d(w) \leftarrow d(v) + w(vw)$
11                DECREASEKEY($Q, w, d(w)$)
```

Prove that, when this procedure finishes, $d(v)$ is the length of a shortest path from s to v in G, provided that all edges in G have non-negative weights.

Question 2.2 (Analysis of algorithms) 15 marks

The following algorithm is an algorithm for multiplying two square matrices in subcubic time. In the description of the algorithm, we use n to denote the number of rows and columns in the two input matrices, and $m_{i,j}$ denotes the entry in row i and column j of matrix M. We assume that n is a power of 2. If $n \geq 2$, we use $M_{1,1}$, $M_{1,2}$, $M_{2,1}$, and $M_{2,2}$ to denote the top-left, top-right, bottom-left, and bottom-right quarter of matrix M, respectively.

```
STRASSEN($A, B$)
 1  if $n = 1$
 2      then return $a_{1,1} \cdot b_{1,1}$
 3  $M_1 \leftarrow$ STRASSEN($A_{1,1} + A_{2,2}, B_{1,1} + B_{2,2}$)
 4  $M_2 \leftarrow$ STRASSEN($A_{2,1} + A_{2,2}, B_{1,1}$)
 5  $M_3 \leftarrow$ STRASSEN($A_{1,1}, B_{1,2} - B_{2,2}$)
 6  $M_4 \leftarrow$ STRASSEN($A_{2,2}, B_{2,1} - B_{1,1}$)
 7  $M_5 \leftarrow$ STRASSEN($A_{1,1} + A_{1,2}, B_{2,2}$)
 8  $M_6 \leftarrow$ STRASSEN($A_{2,1} - A_{1,1}, B_{1,1} + B_{1,2}$)
 9  $M_7 \leftarrow$ STRASSEN($A_{1,2} - A_{2,2}, B_{2,1} + B_{2,2}$)
10  $C_{1,1} \leftarrow M_1 + M_4 - M_5 + M_7$
11  $C_{1,2} \leftarrow M_3 + M_5$
12  $C_{2,1} \leftarrow M_2 + M_4$
13  $C_{2,2} \leftarrow M_1 - M_2 + M_3 + M_6$
14  return $C$
```

Analyze the running time of this algorithm, that is, provide an expression $T(n) = \Theta(f(n))$ and prove that this is indeed the right bound for the running time of the algorithm.

Question 2.3 (Lower bounds) 10 marks

The idea behind every NP-hardness proof we discussed in class was that of a polynomial-time reduction. A similar idea can be used to prove lower bounds for the running times of algorithms or data structure operations. In class, I mentioned that, using comparisons only to determine the correct order of elements, it is impossible to sort in $o(n \log n)$ time. Using this fact, prove that it is impossible to construct a comparison-based dictionary data structure that supports INSERT and SUCCESSOR operations in $o(\log n)$ time and the MINIMUM operation in $o(n \log n)$ time. (*Hint: Assume that you had such a dictionary. Can you use it to design a comparison-based sorting algorithm that runs in* $o(n \log n)$ *time? This would lead to a contradiction.*)

Question 3.1 (Divide and conquer) 15 marks

Given a sequence of numbers $S = (x_1, x_2, \ldots, x_n)$, define the *largest smaller predecessor* (LSP) of x_i to be $\max(\{x_j \mid j < i \text{ and } x_j < x_i\} \cup \{-\infty\})$. Develop an $O(n \log n)$-time algorithm that determines the LSP of every element in S. Provide a discussion of your algorithm that establishes its correctness and running time.

Question 3.1 (Continued)

Question 3.2 (Dynamic programming) **15 marks**

Here is a problem closely related to the sequence alignment problem discussed in class: Given two strings $X = x_1 x_2 \dots x_m$ and $Y = y_1 y_2 \dots y_n$, a *common substring* of X and Y is a string $Z = z_1 z_2 \dots z_k$ that can be obtained from both X and Y by removing an appropriate set of letters. Formally, we have $z_h = x_{i_h} = y_{j_h}$, for appropriate indices $1 \le i_1 < i_2 < \cdots < i_k \le m$ and $1 \le j_1 < j_2 < \cdots < j_k \le n$. Z is a *longest common substring* (LCS) of X and Y if there is no common substring of X and Y that is longer than Z. Develop an algorithm that finds an LCS of X and Y in $O(mn)$ time.

Question 3.2 (Continued)

Question 3.3 (Graph algorithms) **15 marks**

In class, we discussed the Bellman-Ford shortest path algorithm, which computes correct shortest paths from a source vertex *s* to all other vertices in a directed graph *G* even in the presence of negative edge weights. Of course, distances are well defined only if *G* does not contain a negative cycle (ie, a directed cycle whose total edge weight is negative). Therefore, it would be useful to be able to detect when *G* contains a negative cycle. Using the Bellman-Ford algorithm as a starting point, develop an algorithm that correctly decides whether *G* contains a negative cycle. Your algorithm doesn't have to report a negative cycle, only give a correct yes/no answer. The running time of your algorithm should be O(nm). Prove that your algorithm is correct.

Question 3.3 (Continued)