COMP 271 Design and Analysis of Algorithms
Fall 2008 Final Exam

1. **Multiple Choice I** (12 pts)

   Each question below has exactly one of the following answers.

   (a) $\Theta(1)$ (b) $\Theta(\log n)$ (c) $\Theta(n)$ (d) $\Theta(n \log n)$ (e) $\Theta(n^2)$

   For each question, write down the letter that corresponds to your answer. You do not need to justify your answers. Each correct answer earns you 3 points, an incorrect answer (or no answer) gives you nothing.

   1.1 What is $\sum_{i=1}^{n} \frac{n}{2^i}$?

   1.2 What is the solution of the recurrence $T(n) = T(n/2) + n \log n, T(1) = 1$?

   1.3 In the greatest common divisor (gcd) problem, we are given two positive integers n and m, and the goal is to find the largest integer that divides both n and m. Suppose $n > m$, what is the input size of this problem?

   1.4 What is the running time of the fastest possible algorithm to solve Sudoku puzzles? A Sudoku puzzle consists of a 9×9 grid of squares, partitioned into nine 3×3 sub-grids; some of the squares contain digits between 1 and 9. The goal of the puzzle is to enter digits into the blank squares, so that each digit between 1 and 9 appears exactly once in each row, each column, and each 3×3 sub-grid. The initial conditions guarantee that the solution is unique.

		1						
		2		3				4
			5			6		7
5			1	4				
	7						2	
				7	8			9
8		7			9			
4				6		3		
						5		

A Sudoku puzzle. **Don't try to solve this during the exam!**

2. **[OMITTED FROM SYLLABUS]**
   Multiple Choice II (8 pts)

   Aliens from another world come to Earth and confirm our conjecture that the 3-SAT problem cannot be solved in polynomial time. Then what can you say about each of the following statements? Choose your answers from the three below:

   (a) True (b) False (c) Unknown

For each question, write down the letter that corresponds to your answer. You do not need to justify your answers. Each correct answer earns you 2 points, an incorrect answer (or no answer) gives you nothing.

1.1 $P \neq NP$.

1.2 For a problem X in NP, if we can give a reduction that $X \leq_P$ 3-SAT, then X cannot be solved in polynomial time.

1.3 No NP-complete problem can be solved in polynomial time.

1.4 Given a graph G with n vertices, deciding whether G has a clique of size 10 cannot be done in polynomial time.

3. (16 pts) In the midterm you designed an algorithm to find a local minimum in a one-dimensional array of size n. Now you are asked to design an algorithm to find a local minimum in an $n \times n$ two-dimensional array (a matrix). A number in a matrix is a local minimum if it is smaller than all of its four neighbors (if there are any). You may assume that all the numbers are distinct. For example, in the matrix below, all the underlined numbers are local minima. Your algorithm just needs to find only one of them.

$$\begin{bmatrix} \underline{20} & 30 & 40 & 50 & 60 \\ 31 & 35 & 46 & \underline{21} & 43 \\ 45 & \underline{33} & 44 & 55 & 66 \\ 66 & 36 & 25 & 37 & 56 \\ 99 & 43 & \underline{24} & 75 & 80 \end{bmatrix}$$

For full credits, your algorithm should run in $O(n)$ time. Note that there are n^2 numbers in the matrix. [Hint: use divide-and-conquer.]

4. (16 pts) Give a directed, acyclic graph $G = (V, E)$, a source vertex $s \in V$ and a sink vertex $t \in V$, design and analyze an algorithm to find the longest path (with the maximum number of edges) in the graph that goes from s to t. You can assume that there is at least one path from s to t. For full credits, your algorithm should run in $O(|V| + |E|)$ time. [Hint: You might want to first topologically sort this graph, and you don't need to give the details of topological sorting.]

5. (16 pts) A sequence of numbers $a_1, a_2, a_3, \ldots, a_n$ is *oscillating* if $a_i < a_{i+1}$ for every odd index i and $a_i > a_{i+1}$ for every even index i. For example, the sequence 2, 7, 1, 8, 2, 8, 1, 8, 3 is oscillating. Describe and analyze an efficient algorithm to find a longest oscillating subsequence in a sequence of n integers. Your algorithm only needs to output the length of the oscillating subsequence. For example if the input sequence is 2, 4, 5, 1, 4, 2, 1, your algorithm should output 5, corresponding to the subsequence 2, 4, 1, 4, 1, or 2, 4, 1, 4, 2, or any other such subsequence. For full credits, your algorithm should run in $O(n^2)$ time.

6. (16 pts) Let's consider a long river, along which n houses are scattered. You can think of this river as an axis, and the houses are given by their coordinates on this axis in a sorted order. Your company wants to place cell phone base stations at certain points along the river, so that every house is within 4 kilometers of one of the base stations. Give an efficient algorithm that minimizes the number of base stations used, and show that it indeeds gives the optimal solution. For full credits, your algorithm should run in $O(n)$ time.

7. **[OMITTED FROM SYLLABUS]**
(16 pts) The decision version of the 0-1 Knapsack problem is defined as follows. Given two positive integers V and W, and n objects $o_1, \ldots, o_n$, where o_i has value v_i and weight w_i. Is there a subset S of the objects such that $\sum_{o_i \in S} w_i \leq W$ and $\sum_{o_i \in S} v_i \geq V$? Now you will show that (the decision version of) 0-1 Knapsack is NP-complete.

(a) (2 pts) Show that 0-1 Knapsack $\in$ NP.

(b) (12 pts) Show that 0-1 Knapsack $\in$ NPC by a reduction from the Partition problem, which is shown to be NP-complete in the last tutorial. Recall that in the Partition problem, we are given a set A of positive integers $a_1, \ldots, a_n$ such that $\sum_{i=1}^{n} a_i$ is even, and ask if there is a subset B of A such that $\sum_{a_i \in B} a_i = \sum_{a_i \in A-B} a_i = \frac{1}{2} \sum_{i=1}^{n} a_i$.

(c) (2 pts) Having designed a dynamic programming algorithm for 0-1 Knapsack, someone claims that it is polynomially solvable, thus P=NP. What's wrong with this argument?