COMP 271 Design and Analysis of Algorithms
Spring 2009 Final Exam

1. **Quick-Answer Questions** ($3 \times 4 = 12$ pts)

   For each question below, write down the asymptotic (using Θ) result. You do not need to justify your answers.

   1.1 What is the solution of the recurrence $T(n) = T(n/2) + 1, T(1) = 1$? You can assume that n is a power of 2.

   1.2 What is the solution of the recurrence $T(n) = 2T(n/4) + 1, T(1) = 1$? You can assume that n is a power of 4.

   1.3 In Randomized Quicksort, we select a pivot randomly every time. Suppose that one day you got a small "magic device" that will tell you in $O(1)$ time the perfect pivot (i.e., the median) every time, then what is the running time of Quicksort if you use this device instead of choosing the pivots randomly? Note that now the algorithm is deterministic.

   1.4 Suppose we have an alphabet of n characters $A = \{a_1, \ldots, a_n\}$. We are given a text in which the frequencies of these characters are $2^0, 2^1, 2^2, \ldots, 2^{n-1}$, respectively. In the Huffman code of A on this text, what is the length (in terms of bits) of the codewords for a_1, $a_{n/2}$, and a_n, respectively?

2. **[OMITTED FROM SYLLABUS]**
   Multiple Choice ($2 \times 4 = 8$ pts)

   For each of the following statements, indicate whether it is (a) true, (b) false, or (c) unknown based on our current scientific knowledge. You do not need to justify your answers.

   1.1 $P \subseteq NP$.

   1.2 $NPC \cap P = \emptyset$.

   1.3 If SAT can be solved in $O(n^9)$ time, then all NP-complete problems can be solved in polynomial time.

   1.4 UNSAT $\in NP$.

   (UNSAT is the following problem: Given a boolean formula ϕ, if for all assignments to the variables, ϕ is always false, then return "yes"; if there is one assignment that makes ϕ true, then return "no".)

3. (20 pts) Suppose you have k sorted arrays, each with n numbers, and you want to combine them into a single sorted array of kn numbers. We are going to use the MERGE procedure in mergesort for this task. Recall that given two sorted arrays of sizes x and y, MERGE combines them into one sorted array in $O(x + y)$ time. (You don't need to describe the MERGE procedure.)

   (a) (10 pts) One strategy is to first MERGE the first two arrays, then merge in the third, then merge in the fourth, and so on. What is the running time of this algorithm (in terms of k and n)?

   (b) (10 pts) Give a more efficient solution to this problem, and analyze its running time. For full credits your algorithm should run in time $O(nk \log k)$.

4. (15 pts) Suppose that the stairway from the student halls to the academic building (LG7 to LG5) has n stairs. Being tired of climbing the stairs one by one, you decide to adopt a different pattern every day to class, supposing that you can cover 1, 2, or 3 stairs with each step. For example, if $n = 5$, then $(1, 2, 2), (2, 1, 2), (1, 3, 1), (3, 2), \ldots$ are all considered different patterns. Design and analyze an algorithm that computes the total number of different patterns for a given n. For full credits, your algorithm should run in $O(n)$ time. For this problem, assume that addition, substraction, multiplication, or division between any two integers takes constant time, no matter how large the integers are. [Hint: If your algorithm is correct, you will realize that you will never be able to try all possible patterns during your days at UST.]

5. (15 pts) Describe and analyze an efficient algorithm to find the *length* of the longest substring that appears both forward and backward in an input string $T[1..n]$. The forward and backward substrings must not overlap. (Note that if they are allowed to overlap, the problem can be solved by finding the longest common substring of T and the reverse of T.) Here are several examples:

   - For $T =$ ALGORITHM, your algorithm should return 0.
   - For $T =$ CPEGCOMP, your algorithm should return 1, for the substring C or P.
   - For $T =$ MANYDYNAMICPROGRAMMING, your algorithm should return 4, for the substring MANY. Note that neither MANYD or MANYDYNAM count, since each overlaps its backwards occurrence in T.

   For full credits, your algorithm should run in $O(n^2)$ time.

6. (15 pts) Suppose n customers arrive at ABNF[1], each having some business at the (only) counter, and it takes time t_i to serve customer i. ABNF doesn't care about any particular customer but wants to make the n customers happy as a whole. So they will rearrange the customers so that the *total waiting time* is minimized. The total waiting time is just the sum of the waiting times of each customer, and the waiting time of one customer is the total service time of all the previous customers. Design and analyze an algorithm that takes $t_1, t_2, \ldots, t_n$ as inputs, and outputs the optimal order in which the customers should be served. Prove that your algorithm indeed produces the optimal order.

7. **[OMITTED FROM SYLLABUS]**
   (15 pts) Prove that the folowing *Dense Subgraph (DS)* problem is NP-complete: Given a graph G and two integers k and m, does G have a subset of k vertices, $V' \subseteq V$, such that the subgraph induced by V' has at least m edges? The subgraph *induced* by V' is the graph consisting of all vertices in V' and all edges in G that connects two vertices in V'. [Hint: Try to reduce from the DCLIQUE problem. But you can also reduce from any problem that we have shown to be NP-complete in class, the tutorials, and homeworks.]

[1] A Bank that Never Fails