Algorithms CMSC-37000 Final Exam. March 17, 2009

Show all your work. **Do not use text, notes, or scrap paper.** When describing an algorithm in pseudocode, **explain the meaning of your variables** (in English). WARNING: the bonus problems are underrated. This exam contributes 36% to your course grade. **Take this problem sheet home** for your amusement.

1. (1 point) **(Spell)** Spell the singular of "vertices." Print your answer.

2. (30 points) **(Sandwich)** Find three languages L_1, L_2, L_3 over the same alphabet such that $L_1 \subset L_2 \subset L_3$ and $L_2 \in \mathrm{P}$ while L_1 and L_3 are undecidable.

3. (8 + 20 + 2B points) **(Divide et impera)** A divide-and-conquer algorithm reduces a problem instance of size n to two instances of size $n/2$ each. The overhead (cost of the reduction) is $O(\sqrt{n})$. **(a)** State the recurrenct inequality for the complexity $T(n)$. **(b)** Prove: $T(n) = O(n)$. (c) What does the title of this problem mean? In what language?

4. (8+15+8B+8B points) **(Boolean functions)**

   (a) What is the number of Boolean functions in n Boolean variables?

   (b) Construct a 3-CNF formula (CNF formula with exactly 3 literals per clause) which is NOT satisfiable. Make your formula as short as possible. Prove that your formula is indeed not satifiable.

   (c) (BONUS) Prove: almost all 3-CNF formulas with $m = 7n$ clauses are not satifiable. Here n is the number of variables. A random clause is obtained by selecting a triple of distinct variables at random and assigning each variable either itself or its negation by flipping three coins. A random 3-CNF is the AND of m independently chosen random clauses (so repetition is possible). (Checknote: the size of the sample space is $(8\binom{n}{3})^m$.)

   (d) (BONUS) Find an explicit Boolean function in $n \geq 4$ variables which cannot be represented as a 3-CNF formula. Your function must have a very simple (mathematical) description. Prove.

5. (16+3+10+4 points) **(Modular exponentiation)** Given the positive integers a, b, m, compute the quantity $a^b \pmod{m}$ in polynomial time. **(a)** Describe your algorithm in ELEGANT pseudocode. Your algorithm must NOT make recursive calls and must NOT make explicit use of the binary expansion of b. **(b)** Name the method used. **(c)** State the loop invariant from which the correctness of the algorithm immediately follows. **(d)** If Alice wants to send Bob an RSA-encrypted message and Bob wishes to decrypt it, who needs to perform modular exponentiation?

6. (28 points) **(Interval scheduling)** The "weighted interval scheduling" problem takes as input a list of n intervals $(s(i), t(i))$ and corresponding weights $w_i > 0$ and asks to find a set of disjoint intervals among these of maximum total weight. Solve this problem in $O(n)$ plus sorting whatever needs to be sorted. Hint: dynamic programming. Half the credit goes for a clear definition of the array of problems to be solved (the "brain" of the algorithm), including a statement of what needs to be sorted.

7. (12+6 points) **(Huffman code)** (a) The Qwerty language uses the 6-letter alphabet $\{Q, W, E, R, T, Y\}$. The Huffman code for the alphabet is $\{0, 10, 110, 1110, 11110, 11111\}$ (in this order). Find a frequency distribution that results in this code. (b) Prove that no frequency distribution over this alphabet could result in the Huffman code $\{0, 10, 110, 1110, 11110, 111110\}$.

8. (10+10+10B points) **(Large numbers)** **(a)** Given $n \geq 1$, prove that $n!$ cannot be computed in polynomial time. Clearly state the two relevant quantities about which you claim that one is not polynomially bounded as a function of the other. **(b)** Can the quantity $n^{\lfloor \log n \rfloor}$ be computed in polynomial time? Prove your answer. **(c)** Given the positive integers k and m, compute $F_k \pmod{m}$ in polynomial time (F_k is the k-th Fibonacci number). Assuming both m and k are n–bit integers, estimate the time; state the exponent of n.

9. (8+4+18+6 points) **(Determinant)** Let $A = (a_{i,j})$ be an $n \times n$ matrix. **(a)** Define $\det(A)$. Prove: if$A$ is integral (all entries $a_{i,j}$ are integers) then $\det(A)$ is an integer. **(b)** Assume A is integral. Define the bit-length of A. **(c)** Assume A is integral. Prove that the bit-length of the integer $\det(A)$ is not greater than the bit-length of A. **(d)** Describe the significance of statement (c) to the complexity of Gaussian elimination.

10. (16+16 points) **(Critical path)** Let G be a weighted DAG (directed acyclic graph) (edge (i, j) is assigned weight $w(i, j) \in \mathbb{R}$). Find the cost of a <u>max cost</u> path from a vertex s to a vertex t ("critical path"). Your algorithm should run in linear time. Describe your algorithm in

pseudocode. You may not refer to known subroutines. Half the credit goes for a vital subroutine.

11. (15+20 points) **(Good assignments)** Given a 3-CNF formula with m clauses, a "good assignment" is an assignment of Boolean values to the variables that satisfies at least $7m/8$ of the clauses. (Each clause involves 3 distinct variables.) **(a)** Prove: a good assignment always exists. If you use random variables, state the probability space you are referring to. **(b)** Give a deterministic algorithm which finds a good assignment in polynomial time. Prove that your algorithm is correct.

12. (8+18+6 points) **(a)** Define the min-cost spanning tree problem (input, output). Make sure you specify the conditions the input needs to satisfy. **(b)** Jarník's (a.k.a. Prim's) algorithm grows a tree from a start node. Describe the algorithm in pseudocode. **(c)** Name the three abstract data structure operations required for the implementation of the algorithm.

13. (10+10 points) **(B-trees)** What is the **(a)** minimum **(b)** maximum number of keys stored in a 3-4-5-6-tree (B-tree with parameter $t = 3$) of height h? Give simple closed-form expressions. Prove.

14. (12+12 points) **(Batcher's sort)** **(a)** Batcher's odd-even merging network has depth (parallel time) $M(n)$. Write a recurrence for $M(n)$. Evaluate $M(n)$ for $n = 2^k$. **(b)** Batcher's sorting network has depth $S(n)$. Write a recurrence for $S(n)$. Evaluate $S(n)$ (exactly) for $n = 2^k$.