Algorithms CMSC-37000 Final Exam. March 18, 2008

Instructor: László Babai

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). This final exam contributes 36% to your course grade. **TAKE THIS PROBLEM SHEET HOME.**

WARNING: The bonus problems are underrated.

1. (7+7+5B points) **(EZ or not EZ?)** Given the positive integer n, is it possible to calculate in polynomial time **(a)** $n^{\lfloor \log n \rfloor}$; **(b)** $n^{\lfloor \sqrt{n} \rfloor}$; **(c)** (BONUS) $\lfloor n^{\log n} \rfloor$? Prove your answers.

2. (3+16+0B points) **(Hugin and Munin)** Munin incorrectly recalls the SUBSET-SUM language as $\{(a_1, \ldots, a_n, b) : (\exists S \subseteq [n])(\sum_{i \in S} a_i \geq b)\}$, where the a_i and b are positive integers. Let us call this language MSS. Hugin claims that if MSS $\in$ NPC then RSA can be broken in polynomial time. **(a)** Correct Munin's error. **(b)** Sketch the proof of Hugin's claim by listing a sequence of results stated in class that add up to the claim. Clarity is paramount. Make sure you don't miss an important step. **(c)** (BONUS) Which of these two creatures has wings? What are their relevant attributes?

3. (6 points) **(Huge)** State the number of Boolean functions of n variables.

4. (8+8 points) **(NP-hard)** Recall that a computational task (function) is NP-hard if all NP languages can be Cook-reduced to it. Name two NP-hard languages (decision problems) L and M such that **(a)** L provably does not belong to NP; **(b)** M conjecturally does not belong to NP (under what conjecture?) but this is not a proven fact. Prove that your L and M are NP-hard; and indicate why they satisfy (a) and (b), respectively.

5. (18 points) **(Amortized pawnbroker)** AAAA Enterprises, a pawnshop, stores clients' jewelry items in small compartments of a single box, one item per compartment. The number of compartments is the "size" of the box. The boxes are supplied by POWERS, Inc. in sizes of 2^k for all k. In a "transaction," a client either deposits or reclaims

an item, thereby filling or emptying a compartment. When a box of size n is filled up, AAAA purchases a new box of size $2n$, moves the current items to the new box, and destroys the old box. When $n \geq 8$ and only $n/4$ or fewer items are stored, AAAA purchases a new box of size $n/2$, moves the current items to the new box, and destroys the old box, thus maintaning that its box is always filled to at least 1/4 of its capacity (as long as the box has size ≥ 8). POWERS charges n units for a box of size n. The cost (to AAAA) of destroying such a box is the same (n units); moving items to a new box costs one unit per item moved; depositing and reclaiming an item each cost 1 unit (administrative cost) to AAAA. AAAA started with a box of size 4 and 1 deposit. Prove: throughout the history of AAAA, the amortized cost (to AAAA) of a transaction is always $O(1)$.

6. (12 points) **(Add them up)** Design a data structure that holds a fixed number n of data (reals) $a_1, \ldots, a_n$ and serves the following queries: UPDATE(i, x) (effect: $a_i := x$) and SUM(r, s) (returns $\sum_{i=r}^{s} a_i$). No INSERTs or DELETEs. Do not use AVL trees or other dynamic binary search trees; give a very simple solution. All queries should be performed using $O(\log n)$ arithmetic operations and bookkeeping.

7. (1+ 14 points) **(Matrix-chain product)** We use a black box to multiply matrices. The box charges $k\ell m$ units for multiplying a $k \times \ell$ matrix and an $\ell \times m$ matrix. (This is reasonable: it is the number of multiplications required by the textbook procedure.) We use the box to multiply several matrices. The cost will depend on how we place the parentheses. **(a)** Let A and B be $n \times n$ matrices and x an $n \times 1$ matrix. What is the cost of computing the product ABx as $(AB)x$ and what as $A(Bx)$? Which method is preferable? **(b)** Given the positive integers $k_0, k_1, \ldots, k_n$, find the minimum cost of computing the product $A_1 A_2 \ldots A_n$ using the black box, where A_i is a $k_{i-1} \times k_i$. (Minimum cost corresponds to optimal parenthesization.) Your calculation should take $O(n^3)$ steps, where one step is an arithmetic operation on integers, a comparison of integers, or copying or doing other bookkeeping with integers not greater than $(\sum_{i=1}^{n} k_i)^3$. Make sure you give a clear definition of your variables (the "brain" of your algorithm).

8. (6 points) **(An ∃nglish test)** Show three significantly different ways in which one can express existential quantifiers in the English language. Illustrate each by a meaningful mathematical statement (full sentence) related to the material of this class. Do not use the word "exist(s)" or the phrases "there is/are." Rephrase each of your examples into a mathematical formula.

9. (24+6 points) **(Closest pair) (a)** Describe the Shamos - Hoey algorithm to find the closest pair among n points in the plane in $O(n \log n)$.

(b) State and solve the recurrence to which the algorithm leads.

10. (10 points) **(Stepping up)** Given an array $A[1..n]$ of n real numbers such that $A[1] < A[n]$, find i such that $A[i] < A[i+1]$ $(1 \leq i \leq n-1)$. Name the underlying fundamental procedure. Give a complete pseudocode (giving all details of the fundamental procedure). Your procedure should perform the minimum possible number of comparisons of reals (in the worst case); state the maximum number of comparisons your procedure will use. The only type of comparison permitted is "$x < y$?" Do not assume that n is a power of 2; be careful about **rounding**. Do not analyze the algorithm.

11. (12+12 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 max cost 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. The first 10 points go for a vital subroutine.

12. (10+6B+15B 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. **(b)** (BONUS) Give a randomized algorithm which, with probability $> 1/2$, finds a good assignment in polynomial time. Your algorithm will run in "rounds;" prove that $O(m)$ rounds suffice. **(c)** (BONUS) Give a deterministic polynomial-time algorithm to find a good substitution.

13. (BONUS PROBLEM, 8 points) **(Hashing)** Consider the following hashing scheme. Let the universe U consist of all n-digit positive integers. Select at random a prime number $p < n^4$. Let $H(x) = (x \bmod p)$. (So $0 \leq H(x) \leq p-1$.) Given n data (elements from the universe), prove that the expected number of collisions among them approaches zero as $n \to \infty$. (The data are chosen by an adversary who knows our method and wants to maximize the chance of collision but has no information about the prime number chosen (other than that $p < n^4$). (Note that in this problem, n-digit numbers are hashed down to $O(\log n)$-digit numbers.)

14. (BONUS PROBLEM, 4+1+4 points) **(Batcher's odd-even merge)** **(a)** Describe Batcher's odd-even merge network (as a parallel algorithm with compare-exchange modules) to merge two sorted lists of $n = 2^k$ items. **(b)** State the exact number of time beats the network uses. **(c)** Prove the correctness of the network.