Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institute of Theoretical Computer Science
Thomas Holenstein, Ueli Maurer, Angelika Steger, Peter Widmayer

January 30, 2012

Algorithms, Probability, and Computing Fall 2011 Final Exam

Candidate:

First name: ....

Last name: ....

Student ID (Legi) Nr.: ....

I attest with my signature that I was able to take the exam under regular conditions and that I have read and understood the general remarks below.

Signature: ....

General remarks and instructions:

1. You can solve the 6 exercises in any order. **You should not be worried if you cannot solve all the exercises!** Not all points are necessary in order to get the best grade. Usually, it pays off to solve fewer tasks but these cleanly. Select wisely, read all tasks carefully first. They are not ordered by difficulty or in any other meaningful way.
2. Check your exam documents for completeness (2 cover pages and 3 pages containing 6 exercises).
3. Immediately inform an assistant in case you are not able to take the exam under regular conditions. Later complaints are not accepted.
4. Pencils are not allowed. Pencil-written solutions will not be reviewed.
5. No auxiliary material allowed.
6. Attempts to cheat/defraud lead to immediate exclusion from the exam and can have judicial consequences.
7. Provide only one solution to each exercise. Cancel invalid solutions clearly.
8. **All solutions must be understandable and well-founded. Write down the important thoughts in clear sentences and keywords. No points will be awarded for unfounded or incomprehensible solutions (except in the multiple-choice parts). You can write your solution in English or German.**
9. You do not need to reprove things thats were already proved in the lecture. But if you want to prove something *different* then you must point out all details that need to be done differently in your proof.
10. Make sure to write your student-ID (**Legi-number**) on **all** the sheets (but **your name only on this cover sheet**).

Good luck!

	achieved points (maximum)	reviewer’s signature
1	(20)	
2	(32)	
3	(32)	
4	(32)	
5	(32)	
6	(32)	
Σ	(180)	

Exercise 1 - Multiple Choice (20 Pts)

Consider the following 4 claims and mark the corresponding boxes. Grading: 2 points for a correct marking without a correct justification, 5 points for a correct marking with a correct short justification, and -2 points for a wrongly marked box (you will receive non-negative total points in any case).

(a) The onion of a set of n points in the plane consists of at most $\mathcal{O}(\log n)$ convex hulls.

[ ] True [ ] False

Justification: ....

....

(b) Let $M \in \{0,1\}^{n \times n}$ be a fixed non-zero matrix and let $x, y \in \{0,1\}^n$ be vectors chosen independently and uniformly at random. The probability that there is an odd number of pairs $(i,j) \in \{1..n\}^2$ such that simultaneously $x_i = 1$ and $y_j = 1$ and $M_{ij} = 1$, is at least $\frac{1}{4}$.

[ ] True [ ] False

Justification: ....

....

(c) Let $x_1, x_2, \ldots, x_n$ be mutually independent binary random variables. Then $x_1 \oplus x_2 \oplus \ldots \oplus x_n$ is uniformly distributed *if and only if* at least one of $x_1, x_2, \ldots, x_n$ is uniformly distributed.

[ ] False [ ] True

Justification: ....

....

(d) The complete graph K_4 on four vertices admits a Pfaffian orientation.

[ ] False [ ] True

Justification: ....

....

Exercise 2 - Mixed Tasks (32 Pts)

(a) **CSP Problems.** For any $d \geq 2, k \geq 2$, describe a (d,k)-CSP H on k variables which is satisfied if and only if all variables receive the same value. How many constraints does your formula have?

(b) **Bounded Occurrence CSP Problems.** Let F be a (d,k)-CSP where $d \geq 2$ and $k \geq 3$ are fixed numbers. Recall that a variable $x \in \mathrm{vbl}(F)$ *occurs* r times in F if there are exactly r constraints $C \in F$ such that $x \in \mathrm{vbl}(C)$. Prove that if each variable occurs at most $\frac{d^k}{3k}$ times in F, then F is satisfiable and a satisfying assignment can be found in polynomial time.

(c) **Descendants in Random Search Trees.** Prove that in a random search tree on n nodes (according to the usual distribution obtained from inserting n keys in a uniformly random order), each fixed key i has *on expectation* $\mathcal{O}(\log n)$ number of descendants.

(d) **Line Arrangements Ordered Randomly.** Let L be a (fixed) arrangement of n lines in general position in the plane such that no two vertices of the arrangement have the same x-coordinate. Prove that if we insert these lines one after the other in a uniformly random ordering, then the vertex of largest x-coordinate changes $\mathcal{O}(\log n)$ times *on expectation.*

Exercise 3 - Limited Verifier Capabilities (32 Pts)

(a) Let L be a language and $V(x, w)$ a verifier with the following properties. The verifier expects a proof w of size polynomial in $|x|$ for the statement $x \in L$. It first reads x, tosses $\mathcal{O}(\log |x|)$ random coins, reads one bit of the proof then accepts or rejects. If $x \in L$, then there exists a proof w such that the verifier accepts with probability 1. If $x \notin L$, then for all w, the verifier rejects with probability at least $\frac{1}{2}$. Prove that in this case $L \in \mathrm{P}$, i.e. there is a polynomial time algorithm deciding the language.

(b) Let L be a language and $V(x, w)$ a verifier with the following properties. The verifier expects a proof w of size polynomial in $|x|$ for the statement $x \in L$. It first reads x, tosses $\mathcal{O}(|x|)$ random coins, reads one bit of the proof then accepts or rejects. If $x \in L$, then there exists a proof w such that the verifier accepts with probability 1. If $x \notin L$, then for all w, the verifier rejects with probability at least $\frac{1}{2}$. Prove that in this case there is a polynomial time randomized algorithm A deciding L, i.e. if $x \in L$, A outputs 'yes' on input x, if $x \notin L$, it outputs 'no' with probability at least $\frac{1}{2}$.

Hint: In the case that $x \notin L$, for $i = 1, 2, \ldots, |w|$, consider the probability $p_{i,0}$ that the i-th bit is queried and that the verifier rejects when given a '0' for that bit. Likewise, $p_{i,1}$ is the probability that the i-th bit is queried and the verifier rejects when given a '1' for that bit. Prove that $\exists i : \min\{p_{i,0}, p_{i,1}\} \geq \frac{1}{2|w|}$. Then argue that if $x \notin L$, sufficiently many calls to the verifier lead to a contradiction with a good probability.

Exercise 4 - Approximating the Minimum Cut (32 Pts)

You recall that the algorithm BasicMinCut computes a guess for the size of a minimum cut of a (multi)graph G by repeatedly contracting a uniformly random edge until there are only two vertices left and then returning the number of edges running between these two vertices.

As usual, denote the size of a minimum cut of G by $\mu(G)$. We have derived in the lecture that the number L_G which BasicMinCut outputs (on input G) is always at least $\mu(G)$, and the probability that $L_G = \mu(G)$ is $\Omega(n^{-2})$.

Consider the following slightly modified algorithm BasicMinCut': just like BasicMinCut, it repeatedly contracts a uniformly random edge until there are only two vertices left. But instead of just returning the number of edges between those two vertices in the very end, it returns the smallest degree of any vertex observed during the execution of the algorithm. That is if $G_0, G_1, G_2, \ldots, G_{n-2}$ is the sequence of graphs encountered during contraction, with $G_0 = G$ and $|V(G_{n-2})| = 2$, it returns

$$L_G := \min_{0 \leq i \leq n-2} \min_{v \in V(G_i)} \deg(v).$$

Prove that

(a) BasicMinCut' can be implemented so as to run in time $\mathcal{O}(n^2)$,

(b) $L_G \geq \mu(G)$ always holds,

(c) for any fixed $\alpha > 0$, the sucess probability

$$p_\alpha(n) := \min_{G \text{ a graph on } n \text{ vertices}} \Pr[L_G \leq (1+\alpha)\mu(G)]$$

satisfies the recurrence

$$p_\alpha(n) \geq \left(1 - \frac{2}{(1+\alpha)n}\right) p_\alpha(n-1).$$

Using (c), one can prove that for any fixed $\alpha > 0$, $p_\alpha(n) \in \Omega(n^{\frac{-2}{1+\alpha}})$, but this is just calculation and we do not ask you to do this here.

Exercise 5 - Searching within Pictures (32 Pts)

Suppose you are given a larger image A and a smaller image B and you would like to determine whether the image B appears somewhere within the image A as a rectangular region. To make things simpler, let us assume that the images are 8-bit grayscale and square, that is $A \in \{0...255\}^{m\times m}$ and $B \in \{0...255\}^{n\times n}$ are matrices of dimensions $m \times m$ and $n \times n$ respectively, with entries that are numbers from 0 through 255.

In this task, we are looking for a randomized algorithm that runs in $\mathcal{O}(m^2 n)$ time and outputs either a pair of indices (i, j) such that B is identical to the region $[i, i+n-1] \times [j, j+n-1]$ in A, or 'no' if there is no match. The algorithm is allowed to produce a wrong result with probability at most $\frac{1}{2}$.

You may assume that there is an oracle giving you prime numbers of any size in constant time. Also, as usual, our computational model carries out all arithmetic operations (on numbers of size polynomial in m and n) in constant time.

(a) Find a mapping from images $B \in \{0...255\}^{n\times n}$ to bivariate polynomials p_B (i.e. polynomials in two variables) such that two polynomials $p_B, p_{B'}$ are identical iff B and B' are the same picture.

(b) Suppose $B \neq B'$ are distinct pictures of size $n\times n$. For a prime number $q \in \mathbf{N}$, bound the probability that p_B and $p_{B'}$ evaluate to the same number modulo q if independent uniform random values are substituted for their variables.

(c) Exhibit an algorithm meeting the requirements described above (of course along with a proof).

Exercise 6 - Reductions Involving the Discrete Logarithm Problem (32 Pts)

Let G be a cyclic group of order n, where n is even, and g a generator of G. In the whole exercise, 'efficient' is to mean 'in polynomial time', which in the present context of course means 'polynomial in $\log n$'. Below, $x \in \{0..n-1\}$ always.

(a) Show that given any g^x, the LSB (least significant bit) of x can be computed efficiently.

(b) Suppose you have an efficient algorithm O that will give you, on input an element g^x where x is even, the first root $g^{x/2}$.

Give an efficient algorithm that on input *any* $g^x \in G$ determines x, using queries to O.

(c) Suppose now you have an efficient algorithm O' that will give you, on input an element $g^x \in G$ where x is even and uniformly random (among the even numbers), the first root $g^{x/2}$ with probability $\frac{1}{2} + \epsilon$ or 'fail' with probability $\frac{1}{2} - \epsilon$, where $0 < \epsilon < \frac{1}{2}$ is a fixed constant. (Note: the randomness is over the input instances, i.e. the algorithms are successful for at least $\frac{1}{2} + \epsilon$-fraction of the inputs).

Give an efficient randomized algorithm that on input *any* $g^x \in G$ (not random) determines x, using queries to O'. Your algorithm should *always* give the correct answer but it may have a polynomial running time only *on expectation.*

HINT: First assume that $x < n/2$. Then think about how to get rid of this assumption.