Algorithms and Theoretical Computer Science
Ph.D. Qualifying Examination
Spring 2006

- The exam consists of eight written questions, four in the morning and four in the afternoon. You will have three hours for each group of four questions. Please start your answers to each numbered question on a new sheet of paper.
- All else being equal, it is better to solve some of the problems completely than to get partial credit on every problem.

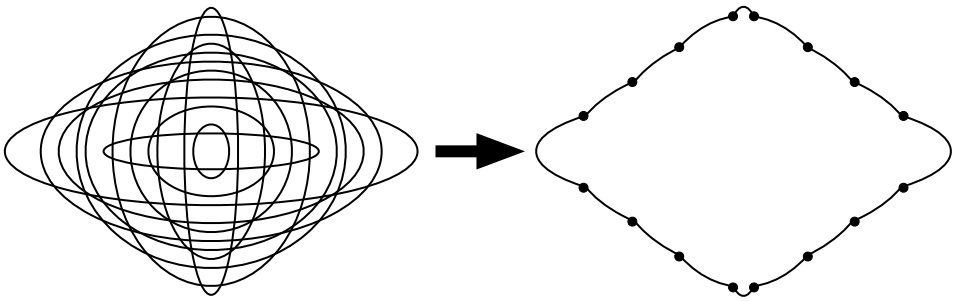
#	1	2	3	4	1	2	3	4	$\sum$
Score									
Grader									

Algorithms and Data Structures

1. Let G be an undirected complete graph with edge weights that satisfy the triangle inequality. For any integer r, an *r-tour* of G is a simple cycle π such that every vertex in G is within distance r of some vertex in π (where distances are computed according to the edge weights).

   - [5 pts] Prove that for any fixed integer r, computing the shortest r-tour of a given graph is NP-Complete.
   - [10 pts] Show that if we know the *vertices* of the shortest r-tour, then we can compute an r-tour that is at most twice as long as the shortest r-tour.
   - [35 pts] Describe a polynomial-time algorithm that outputs an αr-tour of G, of length at most $\beta \cdot \textsc{Opt}_r$, where $\textsc{Opt}_r$ is the length of the optimal r-tour of G and α and β are constants. (You should try to make α and β as small as possible, but any constant values are interesting.)

2. Suppose you are given a set of n axis-aligned ellipses in the plane, all centered at the origin. Describe and analyze an efficient algorithm to compute the union of these ellipses. The output should be a combinatorial description of the boundary of the union. Don't worry about the underlying algebraic details—assume that you can compute any useful function of a constant number of ellipses in constant time. (But please tell us what those useful functions are!)



3. Let $\pi(n)$ denote the number of prime numbers smaller than n.

   (a) Show that the product of all primes p with $m < p \leq 2m$ is at most $\binom{2m}{m}$.

   (b) Using part (a), prove that $\pi(n) = O(n/\log n)$.

   (c) Let p be a prime number, and let m and k be natural numbers. Prove that if p^k divides $\binom{2m}{m}$ then $p^k \leq 2m$.

   (d) Using part (c), prove that $\pi(n) = \Omega(n/\log n)$.

4. (a) [10 pts] In a *rooted ordered binary tree*, each node has a right child, a left child, both, or neither. Sketch an efficient algorithm to compute, given two rooted ordered binary trees A and B, the smallest rooted ordered binary tree that contains both A and B as rooted ordered subtrees.

   (b) [10 pts] In a *rooted binary tree*, each node has zero, one, or two children. Sketch an efficient algorithm to compute, given two rooted binary trees A and B, the smallest rooted binary tree that contains both A and B as rooted subtrees.

   (c) [20 pts] In a *rooted tree*, each node has zero or more children. Sketch an efficient algorithm to compute, given two rooted trees A and B, the smallest rooted tree that contains both A and B as rooted subtrees.

   (d) [10 pts] A *free tree* is a connected acyclic undirected graph. Sketch an efficient algorithm to compute, given two free trees A and B, the smallest free tree that contains both A and B as subtrees.

   In each problem, 'smallest' means 'with the fewest nodes'. In each part, you may use the preceding parts as subroutines. You don't need to describe each algorithm in complete detail; just give a high-level overview and a convincing argument that its running time is polynomial.

Formal Languages and Complexity Theory

1. Suppose $L \in \mathsf{DSPACE}(o(\log\log n))$. Prove that there is an integer n_0 such that L can in fact be recognized in $\mathsf{DSPACE}(\log\log n_0)$; in other words, show that L can be recognized in deterministic *constant* space.

2. Tree automata are a generalization of finite state automata, where the inputs are labeled binary trees instead of strings. More precisely, the input is a finite, oriented binary tree where every node is either a *leaf* with no children, or an *internal node* with a left child, a right child, and a *label* from some finite alphabet Σ.

   Formally, a *tree automaton* is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where just as for standard automata, Q is a finite set of states, $F \subseteq Q$ is the set of accepting states, q_0 is the initial state, and Σ is the input alphabet (in this case, the legal labels of interior nodes of input trees). Finally, $\delta : Q \times Q \times \Sigma \to Q$ is the transition function, which associates a state with each node of the tree inductively as follows.

   - The state associated with each leaf is q_0.
   - The state associated with any internal node v is $\delta(q_L, q_R, a)$, where q_L is the state associated with v's left child, q_R is the state associated with v's right child, and a is the label of v.

   A tree is *accepted* if the state associated with the root is an accepting state.

   (a) State and prove a "pumping lemma" for tree automata that generalizes the standard pumping lemma for finite state automata.

   (b) Using your pumping lemma as part of the proof, exhibit a family of trees that is not accepted by any tree automaton.

3. Recall that a *linear context-free grammar* is a context-free grammar where the right-hand side of every production has at most one non-terminal. A language L is said to be a *linear context-free language* if there is a linear CFG G such that $L = L(G)$. Show that for any linear CFL L and any regular language R, the language $L \cap R$ is a linear CFL.

4. The *leaf complexity* of a Boolean function $f : \{0,1\}^n \to \{0,1\}$, denoted $LC(f)$, is defined as the fewest number of leaves in any formula over the gate set $\{\vee, \wedge, \neg\}$ that computes f.

   (a) Let $\oplus_n$ denote the parity function on n input bits. Show that $LC(\oplus_n) \leq n^2$. You may assume that n is a power of two.

   (b) A *formal complexity measure* is a function $FC : (\{0,1\}^n \to \{0,1\}) \to \mathbb{N}$ that maps n-ary Boolean functions to natural numbers and that has the following properties:

   - *Atomicity:* $FC(x_i) = 1$ for any index i.
   - *Symmetry:* $FC(f) = FC(\neg f)$ for any function f.
   - *Subadditivity:* $FC(f \vee g) \leq FC(f) + FC(g)$ for any functions f and g.

   Prove that $FC(f) \leq LC(f)$ for any formal complexity measure FC and any n-ary Boolean function f.