Algorithms and Theoretical Computer Science

Ph.D. Qualifying Examination Spring 2005

Name:	
Net ID:	Alias:

- Please print your name, NetID, and an alias of your choice in the boxes above. Write your alias, but *not* your name or netID, on each page of your answers. Using an alias will allow us to grade your exam anonymously.
- 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.

#	Score	Grader
1		
2		
3		
4		
5		
6		
7		
8		
Σ		

1. Idiotic Sort [**20 Points**] $\boxed{\text{Sariel}}$

   Given an array A with n (unsorted) distinct numbers in it, the procedure IdioticSort(A) sorts A using the following iterative algorithm. In each iteration, it picks randomly (and uniformly) two indices i, j in the ranges $\{1, \ldots, n\}$; then, if $A[\min(i,j)] > A[\max(i,j)]$ it swaps $A[i]$ and $A[j]$. The algorithm magically stop once the array is sorted. Prove, that with high probablity, the algorithm performs $O(n^2 \log n)$ iterations before stopping. Partial credit will be given for weaker bounds. *Hint:* use the 0-1 principle.

2. Monotonically increasing, or is it decreasing? [**20 Points**] $\boxed{\text{Sariel}}$

   (a) [**10 Points**] Prove the *Erdos-Szekeres Theorem*:

   Theorem 0.1 *Suppose a, b are natural numbers, $n = ab + 1$, and $x_1, \ldots, x_n$, is a sequence of n real numbers. Then this sequence contains a monotonically increasing (decreasing) subsequence of $a+1$ terms or a monotonic decreasing (increasing) subsequence of $b+1$ terms.*

   (b) [**10 Points**] Provide an algorithm as efficient as possible for computing the longest insreasing subsequence for of the sequence $x_1, \ldots,$ x_n.

3. Mad Cow Disease [**20 Points**] $\boxed{\text{Sariel}}$

   In a land far far away (i.e., Canada), the mad cow disease was spreading among cow farms. The cow farms were, naturally, organized as a $n \times n$ grid. The epidemic started when m contaminated cows were delivered to (some) of the farms. Once one cow in a farm has Mad Cow disease then all the cows in this farm get the disease. For a farm, if two or more of its neighboring farms have the disease than the cows in the farm would get the disease. A neighboring farm is a farm directly next to the north, south, east or west; thus a fram has 2, 3 or 4 neighboring farms depending on its location in the grid. We are interested in how the disease spread if we wait enough time.

   - [**5 Points**] Show that if $m = n$ then there is a scenario such that all the farms in the $n \times n$ grid get contaminated.

- [**15 Points**] Prove that if $m \leq n - 1$ then (always) not all the farms are contaminated.

4. PDA Stack [**20 Points**] $\boxed{\text{Mahesh}}$

   Consider a pushdown automata $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$, where Q is the set of states, Σ the input alphabet, Γ the stack alphabet, δ the transition relation, q_0 the initial state, and F the set of final states. Recall that for a state q and stack configuration $\gamma \in \Gamma^*$, we say (q, γ) is reachable if there is some input $\sigma \in \Sigma^*$ such that $(q, \gamma) \in \delta((q_0, \epsilon), \sigma)$, i.e., q with stack γ is reached on reading input σ from the initial configuration. For state $q \in Q$, define

   $$\text{stk}(q) = \{\gamma \mid \gamma \in \Gamma^* \text{ such that } (q, \gamma) \text{ is reachable}\}$$

   Prove that $\text{stk}(q)$ is a regular language, for any $q \in Q$. *Hint:* Define a right congruence $\equiv$ of finite index using the languages $\text{stk}(q')$ such that $\text{stk}(q)$ is union of equivalence classes of $\equiv$.

5. Minimal Finite Automaton [**20 Points**] $\boxed{\text{Mahesh}}$

   For a finite automaton $\mathcal{A} = (Q, \Sigma, q_0, F)$ recall the following useful definitions and notation:

   - A state q is *useful* if there are σ, σ' such that $q \in \delta(q_0, \sigma)$ and $\delta(q, \sigma') \cap F \neq \emptyset$, i.e., q can be reached from the initial state and a final state can be reached starting from q. Recall that removing useless states from an automaton does not change its language.
   - We say $q_1, q_2 \in Q$ are *distinguishable*, if there is $\sigma \in \Sigma^*$ such that σ is accepted starting from exactly one out of q_1 and q_2. Recall that if $\mathcal{A}$ is deterministic without useless states and every pair of states in $\mathcal{A}$ is distinguishable then $\mathcal{A}$ is the minimal automaton.
   - $r(\mathcal{A})$ be the automaton where the transitions of $\mathcal{A}$ are *reversed*, i.e., $q_1 \in \delta^{r(\mathcal{A})}(q_2, a)$ iff $q_2 \in \delta^{\mathcal{A}}(q_1, a)$.
   - $d(\mathcal{A})$ be the automaton obtained by the *subset construction* that is used in determinization.

   In what follows, we assume that we always remove useless states from an automaton, i.e., $r(\mathcal{A})$ is the reversed automaton without useless states, and $d(\mathcal{A})$ is the subset automaton without useless states. Prove that $d(r(d(r(\mathcal{A}))))$ is the minimal automaton recognizing $L(\mathcal{A})$, i.e., by a process of reversing and determinizing two times we can construct the minimal automaton.

6. COVERING POINTS [**20 Points**] $\boxed{\text{JEFF}}$

   Describe and analyze an efficient algorithm for the following problem: Given a set P of n points and a line ℓ in the plane, compute the minimum number of unit-radius circles, each centered on ℓ, needed to cover all the points in P. A point could be covered by more than one circle. If it is not possible to cover all the points, your algorithm should return ∞. Prove that your algorithm is correct.

7. NP-HARD ? [**20 Points**] $\boxed{\text{EDGAR}}$

   In each case determine whether the problem has a polynomial time solution, or whether it is NP-hard. In the first case, you need to describe a polynomial time algorithm; in the latter case, you need to use a reduction (using only well-known NP-hard problems).

   (a) FOLDING THE RULER: You are probably familiar with measuring rulers that consist of equally long wooden segments connected with hinges, so that they can be folded in zig-zag so that they store with length equal to the length of a segment, and they unfold to measure length. Well, you are given a weird ruler whose segments are arbitrary multiples of a base length (so you can think of the length of each segment as an integer). The problem is to fold the ruler, bending some of the hinges and keeping others straight as necessary so as to minimize the folded length of the ruler.

   (b) CHOOSING A PASSWORD: You are opening an account in a system that asks you to choose a password while avoiding certain patterns that are regarded as insecure. The password must be a string with an specified number of characters and the patterns to be avoided are specified by strings of the allowed characters and a special characted $\bullet$, for example, with password length 7, the string $\bullet\bullet a\bullet cd\bullet$ indicates that a password with a, c, d simultaneously in the positions 3,5,6 is not allowed.

8. EXPECTED DISCOVERY TIME [**20 Points**] $\boxed{\text{EDGAR}}$

Consider the following problem. The input is an undirected graph $G = (V, E)$ with associated non-negative edge lengths d_e for $e \in E$, a special vertex $s \in V$ and a probability distribution on the vertices: $0 \leq p_v \leq 1$ fot $v \in V$ with $\sum_v p_v = 1$. Supposed that a *robot* $\mathcal{R}$ starts at s and searches for an object $\mathcal{O}$ located in an unknown vertex of G; the p_v's are an *a priori* probability distribution for the location of $\mathcal{O}$: for each $v \in V$, p_v denotes the probability that $\mathcal{O}$ is in v. The problem is to compute a path to be followed by $\mathcal{R}$ such that the expected time to discover $\mathcal{O}$ is minimized. More precisely, for a path P, the expected time to discover $\mathcal{O}$ is given by

$$\sum_{v \in P} p_v \cdot d(v, P)$$

where $d(v, P)$ denotes the sum of edge lengths over the edges in P from s to the first appearance of v in P (a particular vertex v may appear more than once in P, but it is naturally discovered at the first appearance).

1. [**5 Points**] Show that computing an optimal search path is an NP-hard problem.
2. [**15 Points**] Show that the problem can be solved in polynomial time for a *chain*, that is, a graph that consists of vertices $v_1, v_2, \ldots, v_n$ and edges $\{v_i, v_{i+1}\}$, $i = 1, \ldots, n-1$. The start vertex can be any vertex in the chain (you may assume it is the middle one.)
3. [**EC Points**] What about a *caterpillar* ? (a caterpillar consists of a chain –the *spine*– and (any number of) tree leaves –the *feet*– attached to the spine). What about an arbitrary tree ?

9. BALLS AND BINS [**20 Points**] $\boxed{\text{EDGAR}}$

Consider the usual experiment of throwing m balls into n bins at random. More precisely, each ball is thrown into a bin selected uniformly at random among the n bins (independently from the other balls). As m grows, the probability that balls collide into the same bin increases.

(a) [**10 Points**] Analize the behavior as a function of m of the probability that at least 2 balls collide into the same bin. In particular, at what (asymptotic) value of m (as a function of n) does this event become "almost certain," that is, with probability at least $1 - O(1/n^c)$, at lest 2 balls collide in some bin.

(b) [**10 Points**] Do the same for the case that at least 3 balls collide into the same bin.

Do your best; partial credit will be granted for your work.

10. LANGUAGE INFERENCE $\boxed{\text{LENNY}}$

Let Σ be a fixed alphabet. We are interested in the ability to infer languages from positive examples, as this is what children appear to do so well. An *admissible presentation* of a language $L \subseteq \Sigma^*$ is an infinite sequence $w_1, w_2, \ldots$ such that each $w_i \in L$, and further, for all $w \in L$, there exists at least one $i > 0$ such that $w = w_i$.

Let M be an "inference" TM that has a read-only input tape and write-only output tape. M is said to *identify* L iff for any admissible presentation of L that is written on the input tape, M writes on its output tape an infinite sequence $i_1, i_2, \ldots$ such that for all sufficiently large n, $i_n = i_{n+1}$ and $L(M_{i_n}) = L$.

A collection U of r.e. languages is said to be identifiable iff there exists a single inference TM M such that for all $L \in U$, M identifies L.

(a) Prove that any finite collection U of r.e. languages is identifiable.

(b) Prove that the regular languages are not identifiable. *Hint:* the problem is knowing whether or not to generalize. Create an adversary that forces a machine to keep changing its mind.

11. A GAME ON A GRID $\boxed{\text{LENNY}}$

A popular toy has an array of $n \times n$ translucent buttons which when pushed toggle a light under the button to light it or turn it off. In addition, the lights under the buttons adjacent to the pushed button are also toggled (adjacent means up or down, left or right, but not diagonally). Thus, if all lights are off initially, and a button that is not on an edge or corner is pressed, then after, a "plus" shape of five buttons will be lit. Edge and corner buttons work as you'd expect – they just toggle fewer other lights because they lack certain neighbors.

The goal is to turn all of the lights off from an initial configuration of lit/unlit buttons.

(a) For all n, is it always possible from any given configuration of an $n \times n$ light problem to toggle the lights so that they are all off? Given an $n \times n$ grid and an initial configuration, is it decidable whether the lights can be turned off? If not, prove it. If so, give as efficient an algorithm as you can, and determine whatever you can about the complexity of the problem.

(b) Answer (a) for a modified version of the toy where instead of toggling on/off, each button/light cycles through three colors in this order: red, blue, green, red, blue green, …, an initial configuration is arbitrary color assignment, and the goal is to force all lights into the "green" state.