Algorithms and Theoretical Computer Science
Ph.D. Qualifying Examination
Spring 2004

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. Write your alias, but *not* your name or NetID, on each page of your answers. This 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 problems completely than to get partial credit on every problem.
- Good luck!

A foolish man proclaimeth his qualifications;
A wise man keepeth them secret within himself;
A straw floateth on the surface of the water,
But a precious gem placed upon it sinketh.

— Sakya Pandita Kunga Gyaltsen (translated by Tarthang Tulku)
A Precious Treasury of Elegant Sayings, stanza 58 (c.1270)

Examinations are formidable even to the best prepared,
for the greatest fool may ask more than the wisest man can answer.

— Charles Caleb Colton (1780–1832)

	1	2	3	4	5	6	7	8	Total
Score									
Grader									

1. The *edit distance* or *Levenshtein distance* between two strings A and B is the minimum number of insertions, deletions or substitutions required to transform A into B or vice versa.

   Let $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ be a finite set of strings over some fixed alphabet Σ. An *edit center* for $\mathcal{A}$ is a string $C \in \Sigma^*$ such that the maximum edit distance from C to any string in $\mathcal{A}$ is as small as possible. The *edit radius* of $\mathcal{A}$ is the maximum edit distance from an edit center to a string in $\mathcal{A}$. A set of strings may have several edit centers, but its edit radius is unique.

   $$\text{editradius}(\mathcal{A}) = \min_{C \in \Sigma^*} \max_{A \in \mathcal{A}} \text{edit}(A, C) \qquad \text{editcenter}(\mathcal{A}) = \arg\min_{C \in \Sigma^*} \max_{A \in \mathcal{A}} \text{edit}(A, C)$$

   (a) Describe and analyze an efficient algorithm to compute the edit radius of three given strings.

   (b) Describe and analyze an efficient algorithm to approximate the edit radius of an arbitrary set of strings within a factor of 2. (Computing the edit radius exactly is NP-hard unless the number of strings is fixed.)

2. A boolean function $\mu : \{0,1\}^n \to \{0,1\}$ is *monotone* if and only if $x \subseteq y$ implies $\mu(x) \le \mu(y)$ for all bitstrings x and y. Here "$\subseteq$" denotes natural the partial order on $\{0,1\}^n$ defined by component-wise comparison: $x \subseteq y$ if and only if $x_i \le y_i$ for all i.

   (a) The *monotone extension problem* is the following: Given finite subsets T and F of $\{0,1\}^n$, is there a monotone boolean function f such that $\mu(x) = 1$ for all $x \in T$ and $\mu(x) = 0$ for all $x \in F$? Describe a polynomial-time algorithm for the monotone extension problem.

   (b) A *minimum-discrepancy monotone function* for sets T and F is a monotone function f that minimizes the *discrepancy*, that is, the number of elements of $T \cup F$ that are "misclassified" by f:

   $$\text{disc}(f, F, T) = \left|(\mu^{-1}(0) \cap T) \cup (\mu^{-1}(1) \cap F)\right|$$

   $$\text{disc}(F, T) = \min_{\text{monotone } \mu:\{0,1\}^n \to \{0,1\}} \text{disc}(f, F, T)$$

   For example, if T and F have a monotone extension, then their minimum discrepancy is 0. Describe a polynomial-time algorithm that finds a minimum-discrepancy monotone function for given sets T and F. *[Hint: Consider the bipartite graph with vertices $T \cup F$ where (t, f) is an edge if and only if $f \subseteq t$.]*

3. Two *disjoint* languages A and B are said to be *recursively separable* if there is a recursive language C such that $A \subseteq C$ and $B \subseteq \overline{C}$, that is, if there is a recursive language that "separates" A and B. Show that there are disjoint recursively enumerable languages A and B that are **not** recursively separable.

4. For any set P of points on the real line, a *coloring* is a function $\chi : P \to \{-1, 1\}$. The *discrepancy* of an interval $I = [a, b]$ with respect to the coloring χ is defined as follows:

$$\text{disc}(\chi, I, P) = \left| \sum_{p \in P \cap [a,b]} \chi(p) \right|.$$

The discrepancy of a coloring is the maximum discrepancy of any interval with respect to that coloring: $\text{disc}(\chi) = \max_I \text{disc}(\chi, I, P)$.

(a) **[5 pts]** Show that any set P of points on the line has a coloring with discrepancy 1.

(b) **[5 pts]** Suppose the n points of P are given to you one at a time, and you must permanently color each point as soon as it arrives. Suppose further that the points are provided by an adversary who can see the colors you've assigned to the earlier points. Describe an adversary strategy (for choosing the positions of the points) that guarantees that the discrepancy of your final coloring is at least $n/2$.

(c) **[15 pts]** The previous problem shows that if we want to color points in an online fashion, and we want to have low discrepancy, we must allow ourselves to recolor old points. Describe an algorithm that performs a 'small' number of changes to the coloring but guarantees that the discrepancy of the current set of points is always low. In the worst case, how many recolorings does your algorithm perform during a sequence of n insertions? (A recoloring changes the color of exactly one point; your algorithm may perform several recolorings after each insertion.) What is the maximum discrepancy at any time? (This question is deliberately open-ended; the better your bounds, the more interesting your answer is.)

(d) **[20 pts]** Now suppose we allow both insertions and deletions. Formally, you start with the empty set of points, and the adversary repeatedly either inserts a new point or deletes an existing point. Describe an algorithm that maintains a coloring with low discrepancy. How many recolorings does your algorithm perform after n operations, and what is the maximum discrepancy at any time? You can assume that you know n in advance. (Almost any nontrivial answer here would be interesting; as usual, better bounds are more interesting.)

5. Give the tightest upper bounds you can on the worst-case expected running times of the following randomized sorting algorithms. In each case, the input array has n distinct elements; the subroutine IsNotSorted runs in $\Theta(n)$ time; and Random(k) returns an integer chosen uniformly at random from the set $\{1, 2, \ldots, k\}$.

(a) **[18 pts]**

```
BlindMonkeySort(A[1..n]):
  while IsNotSorted(A)
      i ← Random(n − 1)
      if A[i] > A[i + 1]
          swap A[i] ↔ A[i + 1]
```

[Hint: An inversion *is a pair of indices (i, j) such that $i < j$ and $A[i] > A[j]$. If the array has I inversions, how many inversions of the form $(i, i + 1)$ must there be?]*

(b) **[18 pts]**

```
SpasticMonkeySort(A[1..n]):
  while IsNotSorted(A)
      i ← Random(n)
      j ← Random(n)
      if i > j
          swap i ↔ j
      if A[i] > A[j]
          swap A[i] ↔ A[j]
```

[Hint: Prove that swapping any inverted pair decreases the total number of inversions.]

(c) **[14 pts]**

```
EpilepticMonkeySort(A[1..n]):
  if n ≤ 1
      return

  for i ← n down to 2
      swap A[i] ↔ A[Random(i)]

  pivot ← Random(n)
  count ← 0
  for i ← 1 to n
      if A[i] ≤ A[pivot]
          count ← count + 1

  if count = pivot
      SpasticMonkeySort(A[1..pivot − 1])
      SpasticMonkeySort(A[pivot + 1..n])
  else
      SpasticMonkeySort(A[1..n])
```

6. A *set system* is a pair $\mathcal{X} = (X, \mathcal{F})$, where X is a *ground set* of n elements, and $\mathcal{F}$ is a family of subsets of X. A *hitting set* is a subset $S \subseteq X$ (not necessarily an element of $\mathcal{F}$) that intersects every set in $\mathcal{F}$. Computing a minimum-cardinality hitting set is NP-hard.

   (a) **[15 pts]** Describe an approximation algorithm that computes a hitting set whose cardinality is close to the minimum. What is the approximation quality of your algorithm?

   (b) **[15 pts]** Consider the case where every set in the family $\mathcal{X} = (X, \mathcal{F})$ has at most 10 elements. Give a better approximation algorithm for this case. What is the approximation quality of your new algorithm?

   (c) **[20 pts]** Prove that for any n and k, there is a set system $\mathcal{X} = (X, \mathcal{F})$ containing $|\mathcal{F}| = O(k \log(n/k))$ sets, each with cardinality $\Omega(n/k)$, such that every hitting set has cardinality greater than k. To make things easy, you can assume that $n \geq k^2$ and that k is larger than some constant.

7. Let $V = \{X_1, X_2, \ldots, X_n\}$ and Σ be disjoint finite sets. A *system of algebraic equations* is a set of equations $\{X_i = P_i \mid 1 \leq i \leq n\}$, where each P_i is a finite set of strings in $(V \cup \Sigma)^*$. A solution for such a system is a tuple $(L_1, L_2, \ldots, L_n)$ of languages, where $L_i \subseteq \Sigma^*$, such that replacing the symbol X_i by the language L_i in each equation satisfies those equations.

   For example, the system of equations

   $$X_1 = 0X_2 + \varepsilon, \qquad X_2 = 1X_1$$

   has as one solution the languages

   $$L_1 = (01)^*, \qquad L_2 = 1(01)^*.$$

   A single system of equations can have many different solutions. The set of solutions can be partially ordered by by component-wise inclusion: $(L_1, L_2, \ldots, L_n) \leq (M_1, M_2, \ldots, M_n)$ if and only if $L_i \subseteq M_i$ for each i.

   (a) Let $(L_1, L_2, \ldots L_n)$ be the *minimal* solution of a system of algebraic equations. Prove that each L_i is a context free language.

   (b) Describe an equation system E such that there are solutions to E that are not context free. In the light of the previous part, the non-context-free solutions obviously have to be non-minimal.

   (c) We say an equation system is *strict* if $P_i \subseteq \varepsilon + (V \cup \Sigma)^*\Sigma(V \cup \Sigma^*)$ for each i, that is, if the right hand side of every equation consists of strings that either have at least one symbol from Σ or are ε. Show that every strict equation system has a *unique* solution. *[Aside: The Greibach normal form of any context-free language is a strict equation system, and this demonstrates their algebraic significance.]*

8. Let T be an arbitrary tree with n vertices and constant maximum degree. You already know that T must have an edge whose removal breaks T into two subtrees, each containing a constant fraction of the vertices. (More precisely, each of the subtrees contains at least $\lfloor n/\Delta \rfloor$ vertices, where Δ is the maximum degree in T.) Furthermore, such an edge can be found in linear time. This balanced partition can be repeated recursively to produce a *hierarchical (tree) decomposition* of T with depth $O(\log n)$, which should be useful in some applications.[1] The recursive definition immediately gives an algorithm that runs in $O(n \log n)$ time.

   You are not happy with this running time, however. After all, the decomposition has an implicit representation of size $O(n)$—the recursion tree of splitting edges—so it is not out of question to look for a linear-time algorithm. (Of course, if you insist on storing a copy of the subtrees at each level, the resulting data structure has size $\Theta(n \log n)$, so you can't hope for better construction time. But we are happy with the implicit representation.)

   Answer *one* of the following problems:

   (a) If you are a purist who prefers deterministic algorithms, describe a deterministic algorithm to construct a hierarchical decomposition of T with depth $O(\log n)$, in $O(n)$ time.

   (b) If you prefer randomized algorithms, the following approach might occur to you immediately: Cut the tree at a uniformly chosen edge and recursively decompose the resulting subtrees. It seems reasonable to expect, just as in quicksort, that this will lead to a balanced decomposition.

      i. Show that you can really implement this idea in linear time. You have at your disposal a function RANDOM(n) that returns a random integer between 1 and n, each with equal probability.

      ii. Prove or disprove that this randomized approach actually produces a decomposition with expected depth $O(\log n)$.

[1] and not just as the subject of a qual question!