Algorithms and Theoretical Computer Science
Ph.D. Qualifying Examination
Spring 2003

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		
$\sum$		

Please start your answers for each numbered question on a new sheet of paper. Write your alias, but *not* your name or NetID, on each page of your answers.

1. A sequence $\langle a_1, a_2, \ldots, a_k \rangle$ of distinct real numbers is *monotone* if either $a_1 < a_2 < \cdots < a_k$ or $a_1 > a_2 > \cdots > a_k$. Prove *Dilworth's Theorem*: Any sequence of n distinct real numbers contains a monotone subsequence of length at least $\sqrt{n}$. For example, the sequence $\langle 3, 1, 4, 5, 9, 2, 6, 8, 7 \rangle$ contains the monotone subsequence $\langle 3, 5, 6, 8 \rangle$.

2. This question asks you to to design and analyze a *randomized incremental* algorithm to select the kth smallest element from a given set of n elements (from a universe with a linear order).

   In an *incremental* algorithm, the input consists of a sequence of elements $x_1, x_2, \ldots, x_n$. After any prefix $x_1, \ldots, x_{i-1}$ has been considered, the algorithm has computed the kth smallest element in $x_1, \ldots, x_{i-1}$ (which is undefined if $i \le k$), or if appropriate, some other invariant from which the kth smallest element could be determined. This invariant is updated as the next element x_i is considered.

   Any incremental algorithm can be *randomized* by first randomly permuting the input sequence, with each permutation equally likely.

   (a) Describe an incremental algorithm for computing the kth smallest element.

   (b) How many comparisons does your algorithm perform in the worst case?

   (c) What is the expected number (over all permutations) of comparisons performed by the randomized version of your algorithm? *[Hint: When considering x_i, what is the probability that x_i is smaller than the kth smallest so far?]* You should aim for a bound of at most $n + O(k \log(n/k))$. Revise (a) if necessary in order to achieve this.

3. A *swap* transforms one permutation into another by exchanging one pair of adjacent elements. The *swap distance* between two permutations is the minimum number of swaps required to transform one permutation into the other. For example, the swap distance between $\langle 1, 4, 3, 2 \rangle$ and $\langle 1, 2, 3, 4 \rangle$ is 3; we can transform one permutation into the other with three swaps

$$\langle 1, 4, \underline{3, 2} \rangle \to \langle 1, \underline{4, 2}, 3 \rangle \to \langle 1, 2, \underline{4, 3} \rangle \to \langle 1, 2, 3, 4 \rangle$$

   but not with two swaps.

   (a) Describe an efficient algorithm to compute the swap distance between two given permutations of the set $\{1, 2, \ldots, n\}$.

   (b) Prove that your algorithm is optimal in some appropriate model of computation.

   (c) Prove that the expected swap distance between two random permutations of $\{1, 2, \ldots, n\}$ is $\Theta(n^2)$.

4. Let $f : \{0,1\}^* \to \{0,1\}^*$ be a polynomial-time reduction from A to B. In addition, let f be one-to-one, f^{-1} be computable in polynomial time, and f be length-increasing, that is, $|f(w)| > |w|$ for all w. Let g be a polynomial-time reduction from B to A, such that g is also one-to-one and length-increasing, and g^{-1} is computable in polynomial time. Show that there is a reduction h from A to B such that h is a bijection and both h and h^{-1} are computable in polynomial time. *[Hint: Try to partition A into sets A_1 and A_2 such that $B = f(A_1) \cup g^{-1}(A_2)$, and membership of a string in A_1 is decidable in polynomial time.]*

Please start your answers for each numbered question on a new sheet of paper. Write your alias, but *not* your name or NetID, on each page of your answers.

5. Every year, upon their arrival at Hogwarts School of Witchcraft and Wizardry, new students are sorted into one of four houses (Gryffindor, Hufflepuff, Ravenclaw, or Slytherin) by the Hogwarts Sorting Hat. The student puts the Hat on their head, and the Hat tells the student which house they will join. This year, a failed experiment by Fred and George Weasley filled almost all of Hogwarts with sticky brown goo, mere moments before the annual Sorting. As a result, the Sorting had to take place in the basement hallways, where there was so little room to move that the students had to stand in a long line.

   After everyone learned what house they were in, the students tried to group together by house, but there was too little room in the hallway for more than one student to move at a time. Fortunately, the Sorting Hat took CS 373 many years ago, so it knew how to group the students as quickly as possible. What method did the Sorting Hat use?

G	H	R	R	G	G	R	G	H	H	R	S	R	R	H	G	S	H	G	G
Harry	Ann	Bob	Tina	Chad	Bill	Lisa	Ekta	Bart	Jim	John	Jeff	Liz	Mary	Dawn	Nick	Kim	Fox	Dana	Mel

$$\Downarrow$$

G	G	G	G	G	G	G	H	H	H	H	H	R	R	R	R	R	R	S	S
Harry	Ekta	Bill	Chad	Nick	Mel	Dana	Fox	Ann	Jim	Dawn	Bart	Lisa	Tina	John	Bob	Liz	Mary	Kim	Jeff

   (a) More formally, you are given an array of n items, where each item has one of four possible values, possibly with a pointer to some additional data. Describe an algorithm that rearranges the items into four clusters in $O(n)$ time using only $O(1)$ extra space.

   (b) Now suppose there are k possible values instead of four. Describe and analyze an efficient algorithm that rearranges the items using only $O(\log k)$ extra space.

   (c) Describe and analyze a faster algorithm (if possible) that uses $O(k)$ extra space.

   (d) Describe and analyze an efficient algorithm that uses $O(1)$ extra space.

6. Let f be a polynomial time computable (partial) function. We say that f is *honest* if it doesn't map very long inputs to very short ones, that is, if and only if there is a polynomial p such that for all $y \in \text{range}(f)$, there exists x such that $f(x) = y$ and $|x| \le p(|y|)$.

   Prove that P = NP if and only if every honest partial polynomial time computable function has a polynomial-time computable inverse.

7. The NP-hard Subset Sum problem asks, given a multiset X of n integers and an integer k, whether X contains a submultiset whose elements sum to k. For example, given the multiset $\{\underline{3}, 1, 4, 1, 5, \underline{9}, 2, \underline{6}, \underline{5}, 3, \underline{5}, 8, \underline{9}, 7, 9, \underline{3}, 2, 3, 8, 4, 6, \underline{2}, 7\}$ and the integer 42, the algorithm would return True, since
$$3 + 9 + 6 + 5 + 5 + 9 + 3 + 2 = 42.$$

   Describe an algorithm that solves the Subset Sum problem in time $O(n + f(k))$, where the function $f(k)$ is independent of n and as small as possible.

8. Show that there exists an infinite set $A \subseteq \{0,1\}^*$ having no infinite recursively enumerable subset. *[Hint: Think of defining $\overline{A}$ such that every infinite r.e. set has a non-empty intersection with $\overline{A}$; but ensure that A is infinite.]*