ICS 161 — Algorithms — Winter 1998 — Final Exam

Name:

ID:

1:	out of 15
2:	out of 15
3:	out of 20
4:	out of 15
5:	out of 20
6:	out of 15
total:	out of 100

1. Solve the following recurrences. (Just give the solutions; you do not need to prove that your answers are correct.)

(a) $T(n) = 4T(n/2) + O(n)$

(b) $T(n) = 3T(n/3) + O(n \log n)$

(c) $T(n) = T(n/2) + O(\sqrt{n})$

(15 points)

2. The following routine takes as input a list of n numbers, and returns the first value of i for which $L[i] < L[i-1]$, or n if no such number exists.

```
int firstDecrease(int * L, int n)
{
    for (int i = 1; i < n && L[i] >= L[i-1]; i++) { }
    return i;
}
```

(a) How much time does the routine take, measured as a function of its return value i?

(b) If the numbers are chosen independently at random, then the probability that firstDecrease(L) returns i is $(i-1)/i!$, except for the special case of $i = n$ for which the probability is $1/n!$. Use this fact to write an expression for the average value returned by the algorithm. (Your answer can be expressed as a sum, it does not have to be solved in closed form. Do not use O-notation.)

(c) What is the average case running time of the routine? (Use O-notation.)

(15 points)

3. Suppose you are implementing a spreadsheet program, in which you must maintain a grid of cells. Some cells of the spreadsheet contain numbers, but other cells contain expressions that depend on other cells for their value. However, the expressions are not allowed to form a cycle of dependencies: for example, if the expression in cell E1 depends on the value of cell A5, and the expression in cell A5 depends on the value of cell C2, then C2 must not depend on E1.

(a) Describe an algorithm for making sure that no cycle of dependencies exists (or finding one and complaining to the spreadsheet user if it does exist).

(b) If the spreadsheet changes, all its expressions may need to be recalculated. Describe an efficient method for sorting the expression evaluations, so that each cell is recalculated only after the cells it depends on have been recalculated.

(20 points)

4. Suppose you are given the convex hull of a set of n points, and one additional point (x, y). Describe how to test in time $O(\log n)$ whether or not (x, y) is contained inside the convex hull.

(You may assume that you can test whether the point is above or below a line segment in constant time.)

(15 points)

5.Consider the following approximation to the longest common subsequence of two sequences $A[0 \ldots n-1]$ and $B[0 \ldots m-1]$, each of which contain character values in the range $0 \ldots k-1$: simply find the longest sequence in which all characters are the same.

Equivalently, define $f(S, c)$ to be the number of copies of character c occurring in sequence S, and return $\max_c(\min(f(A, c), f(B, c)))$.

For instance, if the character values are 0 and 1, and the sequences are A =0,1,1,0,1,1,0 and B =1,0,1,0,1, then $f(A, 0) = 3$, $f(A, 1) = 4$, $f(B, 0) = 2$, and $f(B, 1) = 3$, and the length of the approximate longest subsequence would be 3 (there is a common subsequence of 3 ones).

(a) Describe how to implement this algorithm in time $O(m + n + k)$.

(b) What is the approximation ratio for this algorithm, measured as a function of m, n, and k?

(c) Why might it be useful to find approximation algorithms for problems that we already know how to solve exactly in polynomial time?

(20 points)

6. Consider the following two problems.

In P1 we are given as input a set of n squares (specified by their corner points), and a number k. The problem is to determine whether there is any point in the plane that is covered by k or more squares.

In P2 we are given as input an n-vertex graph, and a number k; the problem is to determine whether there is a set of k mutually adjacent vertices. (E.g. for $k = 3$ we are just looking for a triangle in the graph.)

Obviously, the problems are both in NP. There exists a simple translation from P1 to P2: just make a graph vertex for each square, and add an edge between a pair of vertices if the corresponding two squares overlap.

(a) If P1 is NP-complete, would this translation imply that P2 is NP-complete?

(b) If P2 is NP-complete, would this translation imply that P1 is NP-complete?

(c) Show that if we change P1 by making the input be a set of circles instead of squares, then the same translation doesn't work: there exists a set of circles such that no point is covered by 3 or more circles, even though there is a set of 3 mutually adjacent vertices in the corresponding graph.

(15 points)