Honors Algorithms/Written Qualifying Exam

Thursday, Dec. 19, 2002

This is nominally a *three hour* examination, however you will be allowed up to $3\frac{1}{2}$ hours. All questions carry the same weight. Answer all six questions.

• Please *print* your name and the question number on each exam booklet. Write your name and number on the envelope. each question in a *separate* booklet, and *number* each booklet according to the question.

Read the questions carefully. Keep your answers brief. Assume standard results, except where asked to prove them.

Problem 1 [**10 points**] *New Exam booklet PLEASE*
Let the Double Trouble problem be the following:

Given: An undirected graph $G = (V, E)$ with positive integer edge cost $c(i, j)$ for each edge (i, j) in E.

Question: Does G contain a Hamiltonian circuit C where the sum of the costs of the edges in C is exactly half of the sum of all edge costs in the graph G?

Prove that Double Trouble is NP-Complete.

Problem 2 [**10 points**] *New Exam booklet PLEASE*
The Monopoly Restaurant problem is the following. You are given a linear row of restaurants of sizes $S[1..n]$. The objective is to specify merging operations so that all of the restaurants are merged into a single very large restaurant. The merging rules are as follows:

At any step, only a pair of consecutive restaurants can be merged into a new restaurant that is the physical union of the two buildings.

The size of the new restaurant is the sum of the sizes of the two restaurants being merged.

The cost of a merge is equal to the size of the larger restaurant.

The total cost of a sequence of $n-1$ mergings is the sum of the costs of the individual mergings.

The problem is to give an efficient algorithm that computes the minimum total cost for creating one restaurant from the original n restaurants.

Problem 3 [**10 points**] *New Exam booklet PLEASE*
In this problem, we maintain an unsorted linear list L of nodes, beginning with an empty list. There are three operations:

$InsertAfter(x, y)$, which inserts y after x in L. (If $x = Nil$, then x is inserted as the first item in the list.)

$Delete(x)$, which removes x from L.

$Rank(x)$, which returns the number of nodes in L that appear prior to x in the list L.

Please assume that the operations are legal, so that x is already in L, and y is not. Also assume that the variable x, in each case, is provided as a pointer to the relevant node in L.

Explain how to store L by using standard data structures with suitable enhancements so that each of the three operations will run in $O(\log |L|)$ time, where $|L|$ is the number of elements in L.

Go to the next page

Problem 4 [**10 points**] *New Exam booklet PLEASE*
Let $G = (V, E)$ be a directed graph with weighted edges and where all cycles have positive weight. We say that two paths from i to j in G are distinct if the sequence of vertices encountered when traversing the two paths from i to j are not exactly the same. (Thus, they must have some different edges.) Present an algorithm that computes, for each pair of vertices i, j the number of distinct shortest paths from i to j in G. That is, the algorithm should compute n^2 answers, one for each of the n^2 pairs of vertices. (However, the question for the number of paths from i to i is uninteresting, and can be zero or one as you wish.)

Problem 5 [**10 points**] *New Exam booklet PLEASE*
The Tower of Hanoi problems below all begin with n red rings of decreasing size stacked on pole A, with poles B and C empty. The objective is to move the rings one-by-one so that they all wind up on pole B. The rules are that on any stack, the rings of a given color must always have smaller rings above larger rings. Whenever a ring is removed from the top of one stack, it must be immediately placed on the top of another.

a. Let $T(n)$ be the exact number of ring moves for the best solution to the Tower of Hanoi problem with n rings on pole A. Present the recurrence equation for $T(n)$.

b. Now change the problem as follows. Let ring k be the k-th smallest ring, so that ring 1 is the smallest. Let all of the rings be red. To move a size k red ring from the top of one stack on pole X to the top of another on pole Y (that has no smaller rings), the process begins by transforming the ring at no cost as follows. If $k = 1$, there is no change, and the ring is moved in a single step. If $k > 1$, ring k is replaced by a stack of $k - 1$ green rings of sizes 1 to $k - 1$ sorted by size, smallest on top. These replacement rings are still on pole X. Now move the entire stack of $k - 1$ green rings from pole X to Pole Y according to the usual Towers of Hanoi rules. The extra requirements are that green rings must be above all red rings regardless of the size of the red ones, but the green rings must obey the usual size requirement that smaller sizes cannot be below larger size green rings. Once the green stack is on pole Y, it is transformed back into a single red ring of size k.

Present the recurrence equation for $U(n)$, the total number number of red and green ring moves for an efficient solution to this problem. Include, in the count each actual ring move; there is no cost for the transformation from a red ring into the green rings and back. You can refer, in the equation, to the solution for $T(n)$ in part a.

c. The Unlimited Color Problem (UCP). Now let unlimited colors be available, and modify the usual solution as follows. Whenever a ring of size $k > 1$ is to be moved, replace it by $k - 1$ rings of a new color with sizes 1 to $k - 1$ stacked with the smallest on top, and solve the subproblem for the $k - 1$ rings first, as in part b. Note that every move of a ring of size k, for $k > 1$ (and any color) now requires the number of steps that is needed to solve the fully recursive UCP on $k - 1$ rings (of another color). Let $W(n)$ be the number of ring moves needed for this problem. Present the recurrence equation for $W(n)$.

Go to the next page

Problem 6 [**10 points**] *New Exam booklet PLEASE*
Let s be a string over an alphabet Σ.

> We say that $OneReverse(s)$ is the collection of all strings uvy where $s = uv^{\mathcal{R}}y$, and $v^{\mathcal{R}}$ is the reverse of v.

Thus, $OneReverse(abcd) = \{abcd, bacd, cbad, dcba, acbd, adcb, abdc\}$.

> If $L \subseteq \Sigma^*$ is a language, we say that $OneReverse(L)$ is the union of all sets $OneReverse(s)$ for s in L.

a. Show that if L is regular, then $OneReverse(L)$ is regular. Major hint: use nondeterminism. That is, show how to transform a DFA that recognizes L into an NFA that recognizes $OneReverse(L)$.

b. Show that if L is context free, then $OneReverse(L)$ is not necessarily context free. Major hints:

> (i) You can use the fact that if H is a context free language and J is a regular language, then $H \bigcap J$ is also context free.
>
> (ii) Let the underlying alphabet be $\Sigma = \{a, b, c, d\}$