Written Qualifying Exam
Theory of Computation

Tuesday, Dec 16, 2003

This is nominally a *three hour* examination, however you will be allowed up to four hours. All questions carry the same weight. Answer all six questions.

• Please *print* your name on each exam Booklet. Also write your name on the envelope. Answer each question in a *separate* Booklet, and write the problem number on both the inside and outside of the Booklet.

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

Problem 1 [**10 points** NEW BOOKLET PLEASE!]
Let T be a connected undirected DAG, so that it has the structure of a tree. Let V be the set of vertices in T, and let, for v in V, $Adj[v]$ be v's adjacency list. Define the distance between any two vertices v and w in T to be the number of edges on the path from v to w in T. The diameter of T is the largest distance between any pair of vertices in T: $Diam(T) = \max_{v,w \in T} Distance(v, w)$.

Give a linear time algorithm to find the diameter of T.

Hint: If you wish, you can interpret T as a tree with a root. Then some vertex on the path that defines the diameter must be the highest in the tree (i.e., closest to the root). Other approaches might process T as an unrooted structure.

Problem 2 [**10 points** NEW BOOKLET PLEASE!]
A directed graph $G = (V, E)$ is semiconnected if for every pair of distinct vertices $x, y \in V$, there is a directed path from x to y, or a directed path from y to x, or paths in both directions. Give a linear time high level algorithm to determine if G is semiconnected. You can use any standard algorithm as needed without presenting its code, but you **must** give some justification about why your algorithm is correct.

Problem 3 [**10 points** NEW BOOKLET PLEASE!]
The Poor Vertex Cover Problem (PVC) is this:

Instance: $G = (V, E)$ is an undirected graph, and a parameter k.

Question: Is there a subset $S \subset V$ such that S contains k vertices, and at least .1% of the edges in E have at least one endpoint in S?

Show that PVC is NP-Complete.

Problem 4 [**10 points** NEW BOOKLET PLEASE!]
Parts a and b of this question are both solved by dynamic programming.

a. Let L be the set of all strings in $\{0,\ 1\}^*$ where there is one more 0 than 1. Thus, 010, 001, and 100 are the strings of length three in L. Obviously there is an algorithm that can determine if $w \in L$ for any string w, and which runs in linear time (as a function of $|w|$).

Prove that there is a polynomial time algorithm that can recognize the language L^*. It suffices to give a high level recursive specification for the algorithm.

Hint: use dynamic programming.

b. Let L be a language that can be recognized in polynomial time. That means that there is an algorithm A and a constant r such that for any string w, the algorithm can process w in $O(|w|^r)$ time, and $w \in L$ if the algorithm returns "yes" and $w \notin L$ if the algorithm returns "no." Prove that there is a polynomial time algorithm that recognizes L^*. It suffices to give a high level recursive specification for the algorithm.

Hint: use dynamic programming.

c. What is the running time for an efficient implementation of the algorithm in part (b) as a function of $|w|$ and r?

GO TO THE NEXT PAGE

Problem 5 [**10 points** NEW BOOKLET PLEASE!]
Let the string $w = xy$ where $|x| = |y|$. For such a string, define $swap(w) = yx$. For simplicity, let $swap(w) = 1$ if $|w|$ is odd. If L is a language, let $swap(L) = \{swap(w) | \, w \in L\}$, so that $swap(L)$ is just the swap of the strings in L.

a. Show that if L is regular, then $swap(L)$ is not necessarily regular.
Hint: use a language with strings where the swap makes the middle (or so) element special. You might want to pick a nice language that is <u>in</u> 0^*1^*.

b. Show that if L is regular, then $swap(L)$ must be context free. Justify your answer.

c. Show that if L is context free, then $swap(L)$ is not necessarily context free .
Hint: the intuition is to use the fact PDA's have restricted abilities to count sequences of repeated characters. Of course, the pumping lemma is the tool to formalize this fact, but you might want to use the intuition to find a good L where $swap(L)$ is not a CFL.
More Hints: It suffices to pick yet another a nice language in 0^*1^*.

Problem 6 [**10 points** NEW BOOKLET PLEASE!]
The Cops and Robbers problem is as follows. $G = (V, P, B, E, L[*])$ is an undirected graph where each edge $e \in E$ has a length $L[e]$. There are two special disjoint sets of vertices: police stations $P \subset V$ and banks $B \subset V$. At the moment that robbers leave a bank with stolen money, an alarm goes off and police start streaming out of every police station in every direction (because they do not know which bank has been robbed). The time for the police to traverse an edge is $L[e]$, and the time for a robber is also $L[e]$. A vertex $v \in G$ is secure if its distance from the closest police station is less than its distance from the closest bank. Otherwise a vertex is insecure.

a. Give an efficient algorithm to compute the secure and insecure vertices of G.
Note: For simplicity, you can assume that the distances from a bank and a police station to a vertex are never equal.

b. Now suppose that there is just one bank and just one police station, but the robbers can move along an edge three times as quickly as the police. Now the major challenge in designing an algorithm is that a vertex w is insecure only if there is a path from the bank b to w such that the robbers reach each vertex of the path from b to w before the police. Explain how to compute the secure and insecure vertices in this case.
Note: For simplicity, you can assume that the distances from a bank and a police station to a vertex are never equal (when adjusted by the factor of three).
Hint: there are several ways to solve the problem. Probably the simplest way is the most uniform, which is to enhance the basic logic of Dijkstra's SSSP Algorithm to include just a little extra information.