CS 473: Fundamental Algorithms, Spring 2011

Discussion 14

May 3, 2011

P1. RECURRENCES.

Solve the following recurrences:

(A) $T(n) = T(\lfloor n/15 \rfloor) + T(\lfloor n/10 \rfloor) + 2T(\lfloor n/6 \rfloor) + \sqrt{n}$
(B) $T(n) = T(n-1) + 2n - 1$, where $T(0) = 0$

P2. GRAPHS.

Let $G = (V, E)$ be an unweighted, undirected graph and let u and v be two vertices of G. Describe a linear time algorithm to find the number of shortest paths from u to v. Note that we only want the number of paths as there may be an exponential number of them. Give an example graph with an exponential number of (u, v)-paths.

P3. SHORTEST PATHS REDUCTION.

Consider a system of m linear inequalities over n variables $\{x_1, \cdots, x_n\}$. The $k$inequality is in the form $x_{k_1} - x_{k_2} \le t_k$ for $1 \le k_1, k_2 \le n$ and constant t_k (could be positive, zero or negative). The task is to present an algorithm that finds a solution of this system or indicates that the system has no solution.

(A) Build a graph G on vertex set $\{v_1, \cdots, v_n, s\}$. Put a directed edge from s to every v_i with weight 0. If the k-th inequality is $x_{k_1} - x_{k_2} \le t_k$, then put a directed edge from v_{k_2} to v_{k_1} in the graph with weight t_k. Let $d(s, v_i)$ be the length of the shortest path from s to v_i in the graph G. Are the values $d(s, v_i)$ guaranteed to exist? What algorithm could we use to compute all the values $d(s, v_i)$ if they exist?
Assuming that all $d(s, v_i)$ exist, prove that $x_i = d(s, v_i)$ for $1 \le i \le n$, is a solution to our linear system.
(B) Now assume that at least one of the values $d(s, v_i)$ does not exist. Prove that the linear system has no solution.

P4. DYNAMIC PROGRAMMING.

The spring ROTC picnic at UIUC has fallen on a rainy day. The ranking officer decides to postpone the picnic and must notify everyone by phone. Here is the mechanism she uses to do this.

Each ROTC person on campus except the ranking officer reports to a unique *superior officer*. Thus the reporting hierarchy can be described by a tree T, rooted at the ranking officer, in which each other node v has a parent node u equal to his or her superior officer. Conversely, we will call v a *direct subordinate* of u. See the figure in which A is the

ranking officer, B and D are direct subordinates of A, and C is the direct subordinate of B.

To notify everyone of the postponement, the ranking officer first calls each of her direct subordinates, one at a time. As soon as each subordinate gets the phone call, he or she must notify each of his or her direct subordinates, one at a time. The process continues this way until everyone has been notified. Note that each person in this process can only call direct subordinates on the phone; for example, in the figure, A would not be allowed to call C.

We can picture this process as being divided into *rounds*. In one round, each person who has already learned of the postponement can call one of his or her direct subordinates on the phone. The number of rounds it takes for everyone to be notified depends on the sequence in which each person calls their direct subordinates. For example, in the figure, it will take only two rounds if A starts by calling B, but it will take three rounds if A starts by calling D.

Give an efficient algorithm that determines the minimum number of rounds needed for everyone to be notified.
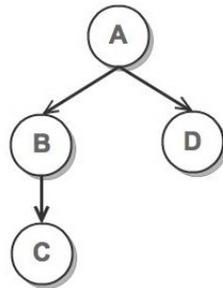
P5. Flow Reduction.

You're organizing the First Annual UIUC Computer Science 72-Hour Dance Exchange, to be held all day Friday, Saturday, and Sunday. Several 30-minute sets of music will be played during the event, and a large number of DJs have applied to perform. You need to hire DJs according to the following constraints:

- Exactly k sets of music must be played each day, and thus $3k$ sets altogether.
- Each set must be played by a single DJ in a consistent music genre (ambient, dubstep, trip-hop, J-pop, etc.).
- Each genre must be played at most once per day.
- Each candidate DJ has given you a list of genres they are willing to play.
- Each DJ can play at most three sets during the entire event.

Suppose there are n candidate DJs and g different musical genres available. Describe and analyze an efficient algorithm that either assigns a DJ and a genre to each of the $3k$ sets, or correctly reports that no such assignment is possible.

P6. PEBBLING PROBLEM.

Pebbling is a solitaire game played on an undirected graph G, where each vertex has zero or more pebbles. A single pebbling move consists of removing two pebbles from a vertex v and adding one pebble to an arbitrary neighbor of v. (Obviously, the vertex v must have at least two pebbles before the move.) The PebbleDestruction problem asks, given a graph $G = (V, E)$ and a pebble count $p(v)$ for each vertex v, whether there is a sequence of pebbling moves that removes all but one pebble. Prove that PebbleDestruction is NP-complete.

P7. 3-DIMENSIONAL MATCHING PROBLEM.

We begin by discussing the 3-Dimensional Matching problem, which can be motivated as a harder version of the Bipartite Matching problem. The Bipartite Matching problem can be viewed in the following way: We are given two sets X and Y, each of size n, and a set P of pairs drawn from $X \times Y$. The question is: Does there exist a set of n pairs in P so that each element in $X \cup Y$ is contained in exactly one of these pairs?

Bipartite Matching is a problem we know how to solve in polynomial time, but things get much more complicated we move from ordered pairs to ordered triples. Consider the following 3-Dimensional Matching problem:

Given disjoint sets X, Y, Z, each of size n, and given a set $T \subseteq X \times Y \times Z$, does there exist a set of n triples in T so that each element of $X \cup Y \cup Z$ is contained in exactly one of these triples?

Such a triple is called a *perfect three-dimensional matching*. Show that 3-Dimensional Matching is NP-complete by reducing from 3-SAT.