Algorithms, Midterm, Nov 5 2014 (in class, total 30pts)

November 6, 2016

You can use any NPC problem we mentioned in class or homeworks for your reductions.

1. (3pts) You are given a directed graph G with edge weights (the weight may be positive or negative or 0). The zero-weight-cycle problem asks if there is a simple cycle in G so that the total weight of the cycle is exactly 0. Prove the problem is NP-Complete.

2. (3pts) Find a polynomial-time algorithm that takes as input a sequence of n symbols, and a number k, and produces as output the number of distinct k-character subsequences. For instance, if the input is the string "food" and the number $k = 2$, the output should be 4. There are four distinct two-character subsequences of food: they are fo,fd, oo, and od.

3. (3pts) We are given two sets of binary strings $A = \{a_1, \ldots, a_m\}$ and $B = \{b_1, \ldots, b_n\}$. We consider the following problem: does there exist any string u such that u is both a concatenation of strings in A (i.e., of the form $a_{i_1} a_{i_2} \ldots a_{i_k}$) and that in B? It appears that the problem is obviously in NP since such a string u itself would be a certificate. However, the argument is not quite correct. What is the problem? (1pt) The problem for you is to show the problem is in NP rigorously. (2pts)

4. (3pts) Show that a graph with maximum degree Δ can be colored in $\Delta + 1$ colors such that no edge is incident on two vertices with the same color.

5. (3pts) Suppose you are a stockbroker who want to sell some stocks in the next n days. Initially, you have x shares. Suppose have a accurate market model that predict the stock prices for the next n days. It predicts the price will take the values $p_1, p_2, \ldots, p_n$. However, if you make large sales, it further decrease the price, indicated by a function $f()$. More precisely, if you sell y_1 shares on day 1, you obtain a price $p_1 - f(y_1)$ (your income would be $y_1(p_1 - f(y_1))$). If you further sell y_2 shares on day 2, you obtain a price $p_1 - f(y_1) - f(y_2)$ (your additional income would be $y_2(p_1 - f(y_1) - f(y_2))$).

   Design an efficient algorithm that sells the stock optimally (i.e., maximize your total income). Your algorithm can be polynomial in x, n and $\max_i p_i$.

6. (4pts) You are helping a group of ethnographers analyze some oral history data they have collected by interviewing someone. From the interviews, they've learned about a set of n people whom we denote by $P_1, \ldots, P_n$. They have also collected facts about when these people lived relative to one another. Each fact has one of the following two forms: (1) For some i and j, person P_i died before person P_j was born; or (2) For some i and j, the life spans of P_i and P_j overlapped at least partially. What they would like you to determine is whether the data they have collected is at least internally consistent, in the sense that there could have existed a set of people for which all the facts theyve learned simultaneously hold. Given an efficient algorithm to do this: either it should produce proposed

dates of birth and death for each of the n people so that all the facts hold true, or it should report that no such dates can exist.

7. (4pts) Let us say that a graph $G = (V, E)$ is a $near - tree$ if it is connected and has at most $n + 8$ edges, where $n = |V|$. Give an algorithm with running time $O(n)$ that takes a near-tree G with costs on its edges, and returns a minimum spanning tree of G. You may assume that all the edge costs are distinct.

8. (6pts) Let's go back to the original motivation for the Minimum Spanning Tree Problem. We are given a connected, undirected graph $G = (V, E)$ with positive edge lengths $\{l_e\}$, and we want to find a spanning subgraph of it. Now suppose we are willing to settle for a subgraph $H = (V, F)$ that is "denser" than a tree, and we are interested in guaranteeing that, for each pair of vertices $u, v \in V$, the length of the shortest $u - v$ path in H is not much longer than the length of the shortest $u - v$ path in G. By the length of a path P here,we mean the sum of l_e over all edges e in P.

   Here?s a variant of Kruskal?s Algorithm designed to produce such a subgraph.

   - First we sort all the edges in order of increasing length. (You may assume all edge lengths are distinct.)
   - We then construct a subgraph $H = (V, F)$ by considering each edge in order.
   - When we come to edge $e = (u, v)$, we add e to the subgraph H if there is currently no $u - v$ path in H. (This is what Kruskal's Algorithm would do as well.) On the other hand, if there is a $u - v$ path in H, we let d_{uv} denote the length of the shortest such path; again, length is with respect to the values $\{l_e\}$. We add e to H if $3l_e < d_{uv}$.

   In other words, we add an edge even when u and v are already in the same connected component, provided that the addition of the edge reduces their shortest-path distance by a sufficient amount.

   Let $H = (V, F)$ be the subgraph of G returned by the algorithm.

   (a) Prove that for every pair of nodes $u, v \in V$, the length of the shortest $u - v$ path in H is at most three times the length of the shortest $u - v$ path in G.

   (b) Despite its ability to approximately preserve shortest-path distances, the subgraph H produced by the algorithm cannot be too dense. Let $f(n)$ denote the maximum number of edges that can possibly be produced as the output of this algorithm, over all n-node input graphs with edge lengths. Prove that

$$\lim_{n\to\infty} \frac{f(n)}{n^2} = 0$$

9. (3pts) Show the following problem is strongly NP-Complete. We are give a set T of tasks. Each task t has a length $\ell_t \in \mathbb{Z}^+$, and a time interval $[a(t), d(t)]$ during which it must be processed. The goal is to process all tasks in a single machine (only one task can be processed at each time slot) subject to the above constraint. (if you only show it is NP-Complete, you get 2pts).