CS 473: Fundamental Algorithms, Fall 2011

Homework 2 (due Monday, 23:55:00, September 12, 2011)

This homework contains three problems. **Read the instructions for submitting the homework on the course webpage. Read the course policies before starting the homework.**

Collaboration Policy: For this homework, Problems 1–3 can be worked in groups of up to three students.

Each student individually have to also do **quiz 2** online.

Submission guidelines: See previous homework.

Version: **1.0**

1. (35 PTS.) Traveling in pink elephants.

   George is a traveling salesperson that sells pink elephants in the US. He is given a list of n cities that he might visit, and a list of m pairs (x_i, x_i') of cities indicating that it is possible to travel from x_i to x_i' (but it might not be possible to travel in the other direction). Georgy (as he known to his friends), needs to figure out his travel plans so that he visits as many distinct cities as possible. Rules of travel:

   (i) Georgy needs to figure out the start and end cities of his travel.
   (ii) One can travel directly only between the prespecified pairs of cities.
   (iii) One is allowed to visit a city several times (of course, only the first visit counts).

   Your tasks should you decide to accept them are

   (A) (20 PTS.) Define the natural graph representing the input. Describe an algorithm to solve the problem if the graph is a DAG. How fast is your algorithm? (The faster, the better.)
   (B) (15 PTS.) Describe an algorithm to solve this problem for a general graph. How fast is your algorithm? (The faster, the better.)

2. (35 PTS.) Arachnids in graphs.

   Let $G = (V, E)$ be an undirected weighted graph (the weights are on the edges and they are all positive). For a set of k vertices $S \subseteq V(\mathsf{G})$, a ***spider*** is a vertex s in the graph, and k paths from s to the vertices of S (each path is a "leg" of the spider). The ***price*** of the spider is the total length of the edges forming these paths. (The paths might share edges, and an edge shared by i legs contribute its weight to the price of the spider i times.)

   (A) (10 PTS.) Describe an efficient algorithm for computing the minimum price spider, given S (the set of k vertices), and G. (I.e., your algorithm need to compute the location of s

and the best spider located at s and its price.) How fast is your algorithm as function of k, n and m? (The faster the better, as usual.)

(B) (5 PTS.) Given a set S as above, a ***web*** for S is a subgraph H of G, such that H is connected and S is a subset of the vertices of H.
We are interested in computing the cheapest possible web for a given set S. We will naturally refer to this problem as the ***web design problem***. We will start with the easy case of $k = 2$. Specifically, describe an algorithm that computes minimum weight web for a set S that contains two vertices of S (this is easy).

(C) (10 PTS.) Assume you are given a recursive procedure **algWebDesignerWeak**(G, S') that can solve the minimum cost web problem, as long as $|S'| < k$. Here k is some fixed number (say 100). Describe an algorithm **algWebDesignerAlmost** that uses **algWebDesignerWeak**(G, S) to solve the problem, for the case that $|S| = k$. How many calls to **algWebDesignerWeak** does your algorithm perform? What is the running time of your algorithm if we ignore the cost of these recursive calls?

(D) (10 PTS.) Describe a recursive algorithm **algWebDesigner** to solve the web design problem that has the same structure as your algorithm for (C). What is the running time of your algorithm as function of n, k and m (the running time is not going to be pretty, but it should be polynomial if k is a constant).

3. (30 PTS.) Misc.

(A) (20 PTS.) I WANT AN ORDER, AND I WANT IT NOW.
Describe an algorithm that given a directed graph G, outputs an ordering $\prec$ of the vertices such that for at least half of the edges of the graph it holds that if $(x, y) \in E(\mathsf{G})$ then $x \prec y$. Show a graph for which any ordering can agree with at most half of its edges.

(B) (10 PTS.) BACK TO VOGSPHERE.
In the 2SAT problem, you are given a set of clauses, where each clause is the disjunction (OR) of two literals (a literal is a Boolean variable or the negation of a Boolean variable). You are looking for a way to assign a value `true` or `false` to each of the variables so that *all* clauses are satisfied—that is, there is at least one true literal in each clause. For example, here's an instance of 2SAT:

$$(x_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee \overline{x}_3) \wedge (x_1 \vee x_2) \wedge (\overline{x}_3 \vee x_4) \wedge (\overline{x}_1 \vee x_4) \quad .$$

This instance has a satisfying assignment: set $x_1 = 1, x_2 = 0, x_3 = 0$, and $x_4 = 1$.
We can solve 2SAT efficiently. Given an instance I of 2SAT with n variables and m clauses, construct a directed graph $G_I = (V, E)$ as follows.

- G_I has $2n$ nodes, one for each variable and its negation.
- G_I has $2m$ edges: for each clause $(\alpha \vee \beta)$ of I (where α, β are literals), G_I has an edge from the negation of α to β, and one from the negation of β to α.

Note that the clause $(\alpha \vee \beta)$ is equivalent to either of the implications $\overline{\alpha} \Rightarrow \beta$ or $\overline{\beta} \Rightarrow \alpha$. In this sense, G_I records all implications in I.
Present a linear time algorithm for finding a satisfying assignment for the given instance by using the algorithm you developed for the PARTY IN VOGSPHERE in the previous homework.

(It is enough to provide the reduction from this problem to the PARTY IN VOGSPHERE problem and prove the reduction is correct. Is is not necessary to repeat the algorithm. In particular, your solution to this question should be reasonably short.)