Analysis of Algorithms	**Problem Set 2**
CS 6820 Fall 2014	**Due 5pm, Monday, September 29, 2014**

This problem set has 5 problems with parts of varying difficulty. For full credit for a grade of A, solve at least 4 of the 5 problems. A full solution for each problem includes proving that your answer is correct. If your group cannot solve a problem, but can do some parts, or have partial results, write down how far you got, and why are you stuck.

Students may work on homework in groups of up to 2-3 people. Each group may turn in a single solution set that applies to all members of the group. However, students are asked to understand each of their group's solutions well enough to give an impromptu whiteboard presentation of the solution. You may use any fact we proved in class without proving the proof or reference, and may read the relevant chapters of the Kleinberg-Tardos or Kozen books, provided you state them clearly. However, you may **not use other published papers, or the Web to find your answer**.

Solutions can be submitted on CMS in pdf format (only). Please type your solution or write extremely neatly to make it easy to read. If your solution is complex, say more than about half a page, please include a 3-line summary to help us understand the argument.

Please ask any clarifying questions using Piazza, where we will post all answers.

(1) Assume $G = (V, E)$ is a bipartite graph, and you want to find a large matching in G. Recall that the greedy algorithm finds a matching of size at least $1/2$ of the size of maximum matching. We say that the greedy algorithm is a 2-approximation algorithm for this problem. Suppose you take a maximal matching M (resulting from a run of the greedy algorithm), and try find a set of disjoint augmenting paths. In this question, you are asked to determine if the proposed 2-round greedy algorithm described below is guaranteed to result in a significant improvement, i.e. is a c-approximation for some $c < 2$.

(a) Greedily add edges to a matching M till no more edges can be added.

(b) Set up the directed graph R_M whose s-t paths correspond to augmenting paths, and greedily find a maximal set of disjoint length-5 augmenting paths. So if the s to t distance in G is greater than 5, the algorithm does nothing; if the distance is 5, it runs a single iteration of the Hopcroft-Karp algorithm. (Note that the distance cannot be smaller than 5, as the distance is always odd, and a length 3 path is just an edge that can be added to matching M, contradicting our assumption that M was maximal.)

(2) Consider the bipartite matching problem on a bipartite graph $G = (V, E)$. As usual, we say that V is partitioned into sets X and Y, and each edge has one end in X and the other in Y. Interpret the left-hand vertex set Y as a set of cellphone-using customers that all want to connect to some cell tower, and the right-hand side X as a set of possible connection points. This being a matching problem, each connection point can only accommodate one

customer. A subset of nodes in $A \subset Y$ can be served if and only if there is a matching M so that M matches all the nodes in A to nodes in X.

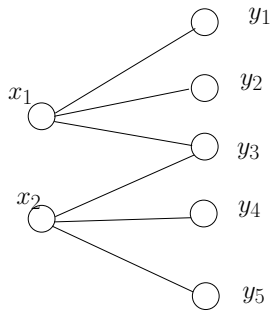

Figure 1: An instance of the Coverage Problem.

For example, in the bipartite graph on the figure, $\{y_1, y_4\}$ can be served, but $\{y_1, y_2\}$ can't. A single node $\{y\}$ can be served if and only if it is adjacent to some edge.

Now, suppose the customers have signed up for different service levels of increasing cost, so that each customer $y \in Y$ pays c_y units of money. The company is thinking about options for how to phrase their service agreement. You are asked to evaluate, for each of two proposals, whether it hurts the overall profit. Suppose you select a subset A of customers that can be assigned to cell tower connection points, with total profit $P = \sum_{y \in A} c_y$. For each option, determine whether you can always select the set A of customers to serve that maximizes the profit while not violating the service agreement.

(I) "Never deny service to someone when you are serving anyone in a lower service level" If two customers y_i and y_j have $c_{y_i} < c_{y_j}$, you may not serve y_i unless you also serve y_j (but among customers with equal c_y values, you may serve any subset).

(II) "Never serve someone when you could serve someone in a higher service level instead" If two customers y_i and y_j have $c_{y_i} < c_{y_j}$, and you serve a set A such that $y_i \in A$ and $y_j \notin A$, then serving $A' = A - y_i + y_j$ must not be possible (that is, there is no matching that matches all the nodes in A').

(3) Suppose we are given a directed network $G = (V, E)$ with a root node r and a set of *terminals* $T \subseteq V$. We'd like to disconnect many terminals from r, while cutting relatively few edges.

We can make this trade-off precise as follows: For a set of edges $F \subseteq E$, let $q(F)$ denote the number of nodes $v \in T$ such that there is no r-v path in the subgraph $(V, E - F)$. Give a polynomial-time algorithm to find a set F of edges that maximizes the quantity $q(F) - |F|$. (Note that setting F equal to the empty set is an option.)

(4) Some of your friends with jobs out West decide to carpool to work. Unfortunately, they all hate to drive, so they want to make sure that any carpool arrangement they agree upon is fair, and doesn't overload any individual with too much driving. Some sort of simple round-robin scheme is out, because no one of them goes to work every single day, and so the subset of them in the car varies from day to day. Here's one way to define fairness: Let the people be labeled $S = \{p_1, ..., p_k\}$. We say that the total driving obligation of p_j over a set of days is the expected number of times that p_j would have driven, had a driver been chosen uniformly at random from among the people going to work each day. More concretely, suppose the carpool plan lasts for d days, and on the ith day a subset $S_i \subset S$ of the people will be going to work. Then, the above definition of the total driving obligation Δ_j for p_j can be written as $\Delta_j = \sum_{i:p_j \in S_i} \frac{1}{|S_i|}$. Ideally, we would like to require that p_j drives at most Δ_j times; unfortunately, Δ_j may not be an integer. So let's say that a driving schedule, a choice of a driver for each day, is *fair* if each p_j is chosen as the driver on at most $\lceil \Delta_j \rceil$ days, i.e., $< \Delta_j + 1$ days.

(a) Prove that for any sequence of sets $S_1, \ldots, S_d$, there exists a fair driving schedule.

- Give an algorithm to compute a fair driving schedule in time polynomial in k and d.

(5) We have seen a minimum cost perfect matching algorithm that repeatedly augments along min-cost augmenting paths starting from an empty matching. Suppose you terminate the algorithm after k iterations. At that point, the matching M has k edges. Is this matching M the minimum-cost one among all matchings containing k edges? Prove that it is or give a counterexample.