CS 473: Fundamental Algorithms, Fall 2011

Homework 8 (due Monday, 23:55:00, October 31, 2011)

Collaboration Policy & submission guidelines: See homework 1.
Each student individually have to also do **quiz 8** online.

Version: **1.3**

1. (30 PTS.) Cellphones and services.

   (This question is harder than the other questions. You should probably do the other two problems first.)

   Consider an assignment problem where we have a set of n stations that can provide service, and there is a set of k requests for service. Say, for example, that the stations are cell towers and the requests are cell phones. Each request can be served by a given set of stations. The problem so far can be represented by a bipartite graph G: one side is the stations, the other the customers, and there is an edge (x, y) between customer x and station y if customer x can be served from station y. Assume that each station can serve at most one customer. Using a max-flow computation, we can decide whether or not all customers can be served, or can get an assignment of a subset of customers to stations maximizing the number of served customers.

   Here we consider a version of the problem with an addition complication: Each customer offers a different amount of money for the service. Let U be the set of customers, and assume that customer $x \in U$ is willing to pay $v_x \geq 0$ for being served. Now the goal is to find a subset $X \subset U$ maximizing $\sum_{x \in X} v_x$ such that there is an assignment of the customers in X to stations.

   Consider the following greedy approach. We process customers in order of decreasing value (breaking ties arbitrarily). When considering customer x the algorithm will either "promise" service to x or reject x in the following greedy fashion. Let X be the set of customers that so far have been promised service. We add x to the set X if and only if there is a way to assign $X \cup \{x\}$ to servers, and we reject x otherwise. Note that rejected customers will not be considered later. (This is viewed as an advantage: If we need to reject a high-paying customer, at least we can tell him/her early.) However, we do not assign accepting customers to servers in a greedy fashion: we only fix the assignment after the set of accepted customers is fixed. Does this greedy approach produce an optimal set of customers? Prove that it does, or provide a counterexample.

   (Observe that deciding if a certain set of customers can be served by the given stations is solvable using the algorithm described in problem 3.)

2. (30 PTS.) Flooding.

   Network flow issues come up in dealing with natural disasters and other crises, since major unexpected events often require the movement and evacuation of large numbers of people in a short amount of time.

   Consider the following scenario. Due to large-scale flooding in a region, paramedics have identified a set of n injured people distributed across the region who need to be rushed to

hospitals. There are k hospitals in the region, and each of the n people needs to be brought to a hospital that is within a half-hour's driving time of their current location (so different people will have different options for hospitals, depending on where they are right now).

At the same time, one doesn't want to overload any one of the hospitals by sending too many patients its way. The paramedics are in touch by cell phone, and they want to collectively work out whether they can choose a hospital for each of the injured people in such a way that the load on the hospitals is *balanced*: Each hospital receives at most $\lceil n/k \rceil$ people.

Give a polynomial-time algorithm that takes the given information about the people's locations and determines whether this is possible.

3. (40 PTS.) Matching via augmenting paths.

Given an undirected, bipartite graph $G = (V, E)$, where $V = L \cup R$ and all edges have exactly one endpoint in L, let M be a matching in G (i.e., a collection of edges that do not share an endpoint). We say that a simple path P in G is an ***augmenting path*** with respect to M if it starts at an unmatched vertex in L, ends at an unmatched vertex in R, and its edges belong alternatively to M and $E \setminus M$. (This definition of an augmenting path is related to, but different from, an augmenting path in a flow network.) In this problem, we treat a path as a sequence of edges, rather than as a sequence of vertices. A shortest augmenting path with respect to a matching M is an augmenting path with a minimum number of edges.

Given two sets A and B, the *symmetric difference* $A \oplus B$ is defined as $(A - B) \cup (B - A)$, that is, the elements that are in exactly one of the two sets.

(A) (10 PTS.) Show that if M is a matching and P is an augmenting path with respect to M, then the symmetric difference $M \oplus P$ is a matching and $|M \oplus P| = |M| + 1$.

(B) (10 PTS.) Given two matchings M and M^* in G, show that every vertex in the graph $G' = (V, M \oplus M^*)$ has degree at most 2. Conclude that G' is a disjoint union of simple paths or cycles. Argue that edges in each such simple path or cycle belong alternatively to M or M^*. Prove that if $|M| \leq |M^*|$, then $M \oplus M^*$ contains at least $|M^*| - |M|$ vertex-disjoint augmenting paths with respect to M.

(C) (10 PTS.) Given a bipartite graph G and a matching M that is not a maximum matching, describe how to orient the edges of G and modify the vertices of G, such that deciding if G contains an augmenting path, can be done by running **DFS** of **BFS** on the resulting graph. In particular, describe a linear time algorithm (using this approach) for computing an augmenting path to M in the graph G if it exists.

(D) (10 PTS.) Describe an $O(nm)$ time algorithm for computing a maximum matching in a bipartite graph using the algorithm from (B).