CS 473: Fundamental Algorithms, Spring 2013
HW 1 (due Monday, at noon, January 28, 2013)

Version: 1.01.

This homework contains two problems. **Read the instructions for submitting homework on the course webpage**.

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 1** online.

---

1. (30 PTS.) The closet mayhem problem.

A new broom closet was built for the Magical Science department of the Unseen University. The department has n students, and every student has a broom. The closet has $m \geq n$ slots where one can place a broom. The ith broom b_i, can be placed only into two possible slots, denoted by x_i and x'_i, where $x_i, x'_i \in \{1, \ldots, m\}$, for $i = 1, \ldots, n$. Given these locations, design an algorithm that decides in polynomial time if there is a legal way to place all of them in the closet, such that no two brooms share a slot. To this end, answer the following.

(A) (5 PTS.) Consider the graph G with $2n$ nodes, where for every broom there are two nodes $[i : x_i]$ or $[i : x'_i]$. For $\alpha \in \{x_i, x'_i\}$ and $\beta \in \{x_j, x'_j\}$, place an edge from $[i : \alpha]$ to $[j : \beta]$, if placing the ith broom at α implies that the jth broom must be placed in the slot β because the other placement of the jth broom is α. How many edges can this graph has in the worst case? What is the running time of your algorithm to compute this graph?

(B) (5 PTS.) If there is a path in G from $[i : \alpha]$ to $[j : \beta]$, then we say that $b_i = \alpha$ ***forces*** $b_j = \beta$. Prove that if $b_i = x_i$ forces $b_j = x_j$ then the "reverse" must hold; that is, $b_j = x'_j$ forces $b_i = x'_i$.

(C) (5 PTS.) Prove that if $[i : x_i]$ and $[i : x'_i]$ are in the same strong connected component of G, then there is no legal way to place the brooms in the closet.

(D) (5 PTS.) Assume that there is a legal solution, and consider a strong connected component X of G involving brooms, say, $b_1, \ldots, b_t$ in G; that is, X is a set of vertices of the form $[1 : x_1], \ldots, [t : x_t]$. Then, prove that $[1 : x'_1], \ldots, [t : x'_t]$ form their own connected component in G. Let this component be the ***mirror*** of X.

(E) (5 PTS.) Prove that if X is a strong connected component of G that is a sink in the meta graph $\mathsf{G}^{\mathrm{SCC}}$, then the mirror of X is a source in the meta graph $\mathsf{G}^{\mathrm{SCC}}$.

(F) (5 PTS.) Consider the algorithm that takes the sink X of the meta-graph $\mathsf{G}^{\mathrm{SCC}}$, use the associated slots as specified by the nodes in X, remove the vertices of X from G and the mirror of X from G, and repeating this process on the remaining graph. Prove that this algorithm generates a legal placement of the brooms in the closet (or otherwise outputs that no such placement exists). Also, describe how to implement this algorithm efficiently. What is the running time of your algorithm in the worst case as a function of n and m.

BTW, for this specific problem, there is a significantly simpler solution. However, the above solution is more general and can be used to solve other problems.

2. (30 PTS.) Heavy time.

Consider a DAG G with n vertices and m edges.

(A) (5 PTS.) Assume that s is a sink in G. Describe how to compute in linear time a set of new edges such that s is the only sink in the resulting graph G (G has to be a DAG). How many edges does your algorithm add (the fewer, the better)?

(B) (10 PTS.) Assume G has a sink vertex s. Some of the vertices of G are marked as being ***significant***. Show an algorithm that in linear time computes all the vertices that can reach s via a path that goes through at least t significant vertices, where t is a prespecified parameter. (Hint: Solve the problem first for $t = 1$ and then generalize.)

(C) (10 PTS.) Assume the edges of G have weights assigned to them. Show an algorithm, as fast as possible, that computes for all the vertices v in G the weight of the ***heaviest*** path from v to s.

(D) (5 PTS.) Using the above, describe how to compute, in linear time, a path that visits all the vertices of G if such a path exists.

3. (40 PTS.) Wishful graph.

Let $\mathsf{G} = (V, E)$ be a directed graph. Define a relation R on the nodes V as follows: uRv iff u can reach v or v can reach u.

(A) (10 PTS.) Is R an equivalence relation? If yes, give a proof, otherwise give an example to show it is false.

(B) (30 PTS.) Call G ***uselessly-connected*** if for every pair of nodes $u, v \in V$, we have that there is either a path from u to v in G, or a path from v to u in G. Give a linear time algorithm to determine if G is uselessly-connected, here linear time is $O(m + n)$, where $m = |E|$ and $n = |V|$.