HW 8 (due Monday, Noon, April 13, 2015)

OLD CS 473: Fundamental Algorithms, Spring 2015 Version: **1.07**

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

---

1. (30 PTS.) Accumulator Faience strikes back.

The Accumulator Faience department at UISB[1] has n members. For obvious reasons, after all it is a faience department, they need to participate in various pottery competitions yearly. There are m competitions each year and the jth competition needs p_j participants. Let $T_i \subseteq \{1, 2, \ldots, m\}$ be the set of competitions that the ith participant has volunteered for. A competition assignment consists of sets $T'_1, T'_2, \ldots, T'_n$ where $T'_i \subseteq \{1, 2, \ldots, m\}$ is the set of competitions that participant i will participate in. A ***valid*** competition assignment has to satisfy two constraints:

(i) for each participant i, $T'_i \subseteq T_i$, that is each member is only participating in competitions that he/she has volunteered for, and

(ii) each competition j has p_j participants assigned to it, or in other words j occurs in at least p_j of the sets $T'_1, T'_2, \ldots, T'_n$.

Unfortunately, often there is no valid competition assignment because people specializing in Accumulator Faience rarely venture outside their homes or offices[2]. To overcome this, the definition of a valid assignment is relaxed as follows. Let ν be some integer. An assignment $T'_1, T'_2, \ldots, T'_n$ is now said to be valid if

(i) $|T'_i \setminus T_i| \leq \nu$ and

(ii) each competition j has exactly p_j people participating in it.

The new condition (i) means that a member i may participate up to ν competitions not on the list T_i that he/she/it volunteered for. Describe an algorithm, as fast as possible, to check if there is a valid competition assignment with the relaxed definition. What is the running time of your algorithm?

2. (30 PTS.) Augmenting Paths in Residual Networks.

You are given an *integral* instance G of network flow. Let C be the value of the maximum flow in G.

(A) (6 PTS.) Given a flow f in G, and its residual network G_f, describe how to compute, as fast as possible, the largest capacity augmenting path flow from s to t. Formally, given a path π in the residual network, its ***residual capacity*** is $c_f(\pi) = \max_{e \in \pi} c_f($
Prove the correctness of your algorithm.

(B) (6 PTS.) Prove, that if the maximum flow in G_f has value T, then the augmenting path you found in (A) has capacity at least T/m.

(C) (6 PTS.) Consider the algorithm that starts with the empty flow f, and repeatedly applies (A) to G_f (recomputing it after each iteration) until s and t are disconnected. Prove that this algorithm computes the maximum flow in G.

(D) (6 PTS.) Consider the algorithm from (C), and the flow g it computes after m iterations. Prove that $|g| \geq C/10$ (here 10 is not tight).

(E) (6 PTS.) Give a bound, as tight as possible, on the running time of your algorithm, as a function of n, m, and C.

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,

[1] The university of Illinois, in Shampoo-Banana

[2] Or cardboards, if they are homeless and unemployed.

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).