CS 473: Algorithms, Fall 2009
HW 8 (due Tuesday, November 3rd in class)

This homework contains three problems. **Read the instruction for submitting homework on the course web page**. In particular, *make sure* that you write the solutions for the problems on separate sheets of paper. Write your name and netid on each sheet.

Collaboration Policy: For this home work, students can work in groups of up to 3 members each. Each group submits only one written solution (some groups will do an oral presentation. Indicate your group members on the homework (netids are needed)).

1. (40 pts) We have seen in class that if f is a flow then one can make it acyclic without changing the value of the flow. This is computationally expensive; it can take $O(nm)$ time. However, one can more easily make a flow *digon*-free. A digon is a pair of edges (u, v) and (v, u). We say that a flow f is digon-free if for any digon either $f(u, v) = 0$ or $f(v, u) = 0$. Note that one can make a given flow f into another flow that is digon-free in $O(m)$ time without affecting its value.

   - (15 pts) Let f and f' be two digon-free flows in G such that $v(f') > v(f)$. Let G_f be the residual flow network defined by G and f. We define a function $f'' : E(G_f) \rightarrow \mathcal{R}^+$ (we think of f'' as $f' - f$) as follows. For an edge e in G, if $f'(e) \geq f(e)$ we set $f''(e) = f'(e) - f(e)$ in the forward edge corresponding to e and $f''(\bar{e}) = 0$ on the backward edge corresponding to e (only if $f(e) > 0$, otherwise $\bar{e}$ does not exist in G_f). If $f(e) > f'(e)$ then we set $f''(\bar{e}) = f(e) - f'(e)$ on the backward edge $\bar{e}$ that corresponds to e and $f''(e) = 0$. Prove that f'' is a flow in G_f of value $v(f') - v(f)$.
   - (10 pts) Let f be a digon-free flow in G. If f^* is a maximum flow in G show that there is an augmenting path in G_f with bottleneck capacity at least $(v(f^*) - v(f))/m$. *Hint:* Use flow-decomposition and the previous part.
   - (10 pts) In this part we analyze the variant of Ford-Fulkerson where in each step the algorithm finds an augmenting path in the residual network with the largest bottleneck capacity (see slide 83 of lecture 14). Our goal is to prove that the algorithm terminates in $O(m \log C)$ iterations. Suppose f_i is the flow after i iterations of the algorithm and f^* is a maximum flow. First show, using the previous part, that $v(f^*) - v(f_{i+1}) \leq (v(f^*) - v(f_i))(1 - \frac{1}{m})$. Second, show that for any i, either the algorithm terminates with a maximum flow by iteration $i+m$ or that $v(f^*) - v(f_{i+m}) \leq (v(f^*) - v(f_i))/e$ where e is Euler's constant (the base of the natural logarithm). You may use the well-known inequality $(1 - 1/k)^k \leq 1/e$ for all $k \geq 1$.
   - (5 pts) Let $T(m)$ be the running time of an algorithm for finding an s-t path with the largest bottleneck capacity. Show that the variant of Ford-Fulkeron discussed in the previous part can be implemented to run in $O((T(m) + m) \cdot m \log C)$ time where C is the minimum-cut value.

2. (30 pts) Let $G = (V, E)$ be a directed graph with non-negative capacity for each edge. For an ordered pair (u, v) of vertices in G, let $\alpha(u, v)$ denote the minimum-cut capacity between

u and v. Show that for any three distinct vertices, u, v, w

$$\alpha(u, w) \geq \min\{\alpha(u, v), \alpha(v, w)\}.$$

3. (30 pts) We have seen an algorithm in class for finding a minimum s-t cut in a flow network. In general there may be several different cuts that all have the same value. Describe an algorithm to find a cut amongst the minimum s-t cuts that has the minimum number of edges. The running time of your algorithm should be close to that of the algorithm for finding an s-t minimum cut.