CS515: Algorithms and Data Structures, Fall 2015

Homework 3*

Due: Mon, Nov/16/15

Homework Policy:

1. Students should work on homework assignments in groups of preferably three people. Each group submits to TEACH one set of typeset solutions, and hands in a printed hard copy in class. The hard copy will be graded.
2. You are allowed to discuss the homework with other groups, however, you must mention their names in your submission. Also, you must cite any other source that you use.
3. The goal of the homework assignments is for you to learn solving algorithmic problems. So, I recommend spending sufficient time thinking about problems individually before discussing them with your friends.
4. *I don't know policy:* you may write "I don't know" *and nothing else* to answer a question and receive 25 percent of the total points for that problem whereas a completely wrong answer will receive zero.
5. Algorithms should be explained in plain english. Of course, you can use pseudocodes if it helps your explanation, but the grader will not try to understand a complicated pseudocode.
6. More items might be added to this list. ☺

Problem 1. A flow f is *acyclic* if the subgraph of directed edges with positive flow contains no directed cycles.

(a) Prove that for any flow f, there is an acyclic flow with the same value as f. (In particular, this implies that some maximum flow is acyclic.)

(b) A path flow assigns positive values only to the edges of one simple directed path from s to t. Prove that every acyclic flow can be written as the sum of $O(E)$ path flows.

(c) Describe a flow in a directed graph that cannot be written as the sum of path flows.

(d) A cycle flow assigns positive values only to the edges of one simple directed cycle. Prove that every flow can be written as the sum of $O(E)$ path flows and cycle flows.

Problem 2. We can speed up the Edmonds-Karp 'fat pipe' heuristic, at least for integer capacities, by relaxing our requirements for the next augmenting path. Instead of finding the augmenting path with maximum bottleneck capacity, we find a path whose bottleneck capacity is at least half of maximum, using the following capacity scaling algorithm. The algorithm maintains a bottleneck threshold Δ; initially, Δ is the maximum capacity among all edges in the graph. In each phase,

*Problems are from Jeff Erickson's lecture notes. Looking into similar problems from his lecture notes on recursion and dynamic programming is recommended.

the algorithm augments along paths from s to t in which every edge has residual capacity at least Δ. When there is no such path, the phase ends, we set $\Delta \leftarrow \lfloor \Delta/2 \rfloor$, and the next phase begins.

(a) How many phases will the algorithm execute in the worst case, if the edge capacities are integers?

(b) Let f be the flow at the end of a phase for a particular value of Δ. Let S be the nodes that are reachable from s in the residual graph G_f using only edges with residual capacity at least Δ, and let $T = V \backslash S$. Prove that the capacity (with respect to G's original edge capacities) of the cut (S, T) is at most $|f| + E \cdot \Delta$.

(c) Prove that in each phase of the scaling algorithm, there are at most $2E$ augmentations.

(d) What is the overall running time of the scaling algorithm, assuming all the edge capacities are integers?