COMP 360 - Fall 2015 - Assignment 1

Due: 6pm Sept 29th.

General rules: In solving these questions you may consult books but you may not consult with each other. There are in total 110 points, but your grade will be considered out of 100. You should drop your solutions in the assignment drop-off box located in the Trottier Building.

"Oh no, not homework again."

---

1. (5 points) Recall that for every flow f and every cut (A, B), we have $\text{val}(f) = f^{out}(A) - f^{in}(A)$ where $f^{out}(A) = \sum_{\substack{uv \in E \\ u \in A, v \in B}} f(uv)$ and $f^{in}(A) = \sum_{\substack{uv \in E \\ u \in B, v \in A}} f(uv)$. Use this fact to show that $\text{val}(f)$ is equal to the total flow on the edges going into the sink.

2. (10 Points) Five types of packages are to be delivered by four trucks. There are three packages of each type. The capacities of the three trucks are 5, 4, 4, and 3 packages respectively. Set up a maximum flow problem that can be used to determine whether the packages can be loaded so that no truck carries two packages of the same type.

3. (10 points) Prove that if an edge e goes from A to B in a minimum cut (A, B), then *every maximum* flow f uses the full capacity of e, that is $f(e) = c_e$.

4. (15 Points) Is there a polynomial time algorithm that decides whether a flow network has a *unique* minimum cut?

5. (15 Points) Consider a flow network.

   - We say that a node v is *upstream* if for all minimum cuts (A, B), we have $v \in A$.
   - We say that a node v is *downstream* if for all minimum cuts (A, B), we have $v \in B$.
   - We say that a node v is *central* if it is neither upstream nor downstream.

   Give an algorithm that takes a flow network G and classifies each of its nodes as being upstream, downstream, or central. The running time of your algorithm should be within a constant factor of the time required to compute a single maximum flow. You must prove that your algorithm is correct.

6. (15 Points) Let A be an $n \times n$ matrix with each entry equal to either 0 or 1. Let a_{ij} denote the entry in the row i and column j. Entries a_{ii} are called the *diagonal* entries. We call A re-arrangeable if it is possible to re-arrange the rows and then re-arrange the columns so that after the rearrangements all the diagonals entries of the matrix are equal to 1.

- Give an example of an $n \times n$ matrix that is not rearrangeable but for which at least one entry in each row and each column is equal to 1.
- Give a polynomial algorithm based on Max-Flow that determines whether a matrix M with $0-1$ entries is re-arrangeable.

7. (15 Points) Consider the variant of the maximum flow problem where every node v also has an integer capacity $c_v \geq 0$. We are interested in finding the maximum flow as before, but now with the extra restriction that $f^{\text{in}}(v) \leq c_v$ for every node v. Solve this problem using the original maximum flow problem.

8. (10 Points) The goal of this exercise is to show that if we modify the Ford-Fulkerson algorithm so that it does not decrease the flow on any of the edges (i.e. we do not add the opposite edges to the residual graph), then the algorithm might perform very poorly.

   For every positive integer m, construct an example of a flow network whose maximum flow is equal to m, but the modified Ford-Fulkerson algorithm terminates after finding a flow of value 1. In other words, in your flow network, there is an augmenting path with bottleneck 1 such that after augmenting this path there are no augmenting paths left that do not push back flow on any edges.

9. (15 Points) Show that there is always a sequence of at most m augmenting paths that leads to maximum flow where m is the number of edges in the flow network. (Hint: Use the maximum flow to find the paths).