COMP 362 - Winter 2015 - Assignment 1

Due: 6pm Jan 30th.

General rules: In solving these questions you may consult books but you may not consult with each other. You should drop your solutions in the assignment drop-off box located in the Trottier Building.

1. (10 Points) For every integer $n \geq 2$, construct a flow network on n vertices such that every s,t-cut (A, B) is a minimum cut. In your construction all the capacities must be strictly positive, and the underlying graph must be connected. How many minimum cuts does this network have?

   Solution: For example we can consider the graph where there is an edge of capacity 1 from s to every internal vertex, and an edge of capacity 1 from every internal vertex to t, plus an edge with capacity 1 from s to t. Each one of the 2^{n-2} possible cuts is of capacity $n-1$.

2. (20 Points) Consider the following two algorithms for finding the maximum flow:

```
Algorithm 1: Scaling max-flow
• Initally set $f(e) := 0$ for all edges $e$.
• Set $\Delta$ to be $\max c_e$ rounded down to a power of 2.
• While $\Delta \geq 1$:
•       While there is an $s,t$-path $P$ in $G_f(\Delta)$:
•             Augment the flow using $P$ and update $G_f(\Delta)$.
•       Endwhile.
•       Set $\Delta := \Delta/2$.
• Endwhile.
• Output $f$.
```

```
Algorithm 2: The fattest path algorithm
• Initally set $f(e) := 0$ for all edges $e$.
• While there exists an $s,t$-path in $G_f$:
•     Augment the flow using the fattest $s,t$-path $P$ in $G_f$.
•     Update $G_f$.
• Endwhile.
• Output $f$.
```

In the second algorithm the fattest means the largest bottleneck. From the class we know that the number of augmentations in Algorithm 1 is at most $2m\lceil\log_2 K\rceil$, where K is the maximum capacity of an edge. Deduce from this that the number of augmentations in Algorithm 2 is also at most $2m\lceil\log_2 K\rceil$.

Solution (sketch): The paths found in Algorithm 2, can also be used in Algorithm 1. Indeed to verify whether there is a path with Bottleneck$(P, f) \geq \Delta$, it suffices to consider the fattest path P.

3. (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.

   Solution: Split every vertex into two vertices v^{in} and v^{out} and put a directed edge from v^{in} to v^{out} with capacity c_v. The input edges of the original vertex v now are pointed to v^{in} and the output edges of the original v leave v^{out}.

4. (a) (10 points) Show that for every flow network, there exists an execution of the Ford-Fulkerson algorithm that never decreases the value of the flow on any of the edges (i.e. never "pushes back" the flow on any of the edges).

   Solution: See the solution to the last question.

   (b) (5 points) On the other hand, 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 terminate without finding the maximum flow.

Solution: Consider the example from the class. $V(G) = \{s, t, a, b\}$. The edges $\{sa, sb, ab, at, bt\}$ are all of capacity 1. If we choose (s, a, b, t) as the first augmenting path, then we find a flow of value 1, while the maximum flow is 2.

5. (20 points) Let $(G, s, t, \{c_e\})$ be a flow network, and let F be the set of all edges e that belong to some minimum cut. That is, there exists at least one minimum cut (A, B) such that e goes from A to B. Give a polynomial time algorithm that finds all edges in F. What is the running time of your algorithm.

   Solution: An edge e is in F if decreasing its capacity decreases the value of the max-flow.

```
• Find the maximum flow $M$;
• For every edge $e$:
•       decrease $c_e$ by 1 and find the new maximum flow $m'$;
•       if $m' < m$ output $e$;
•       restore the original value of $c_e$;
• Endloop;
```

6. (20 points) Is it true that for every flow network with at most m edges, there exist a sequence of at most m augmentations that lead to a maximum flow?

   Solution: To prove this, we apply the following algorithm to decompose a maximum flow f into a set of flow-paths:

   - Repeat the following two steps until there is no such s, t-path:
   - Find an s, t-path P with positive flow, i.e., $f(e) > 0$ for all $e \in P$.
   - Let Δ be the minimum value of the flow f on edges of P. Decrease the flow f on each edge $e \in P$ by Δ.

   In every iteration the value of f on at least one edge becomes 0. Hence the number of iterations is at most m. Using these paths as augmenting paths (with augmenting value Δ) leads to the desired result.