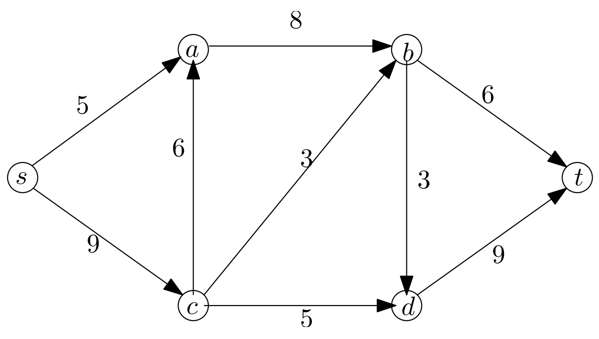
COMP 360 - Winter 2014 - Assignment 1

Due: 6pm Jan 31st.

General rules: In solving these questions you may consult books but you may not consult with each other. There are in total 120 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.

1. (15 Points) Consider the following maximum flow problem, and apply the fattest path algorithm to find a maximum flow. List the augmenting paths used by the algorithm and their bottlenecks in the right order (For example $P_1 = (s, c, a, b, t)$ and Bottleneck(P_1) = 6).



   Solution:

   - $P_1 = (s, c, a, b, t)$ and Bottleneck(P_1) = 6.
   - $P_2 = (s, a, c, d, t)$ and Bottleneck(P_2) = 5.
   - $P_3 = (s, c, b, d, t)$ and Bottleneck(P_3) = 3.
   - The max flow is 14.

2. (10 Points) Consider the following three 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:
• 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 the fattest $s,t$-path $P$ in $G_f$ has Bottleneck$(P, f) \geq \Delta$:
•         Augment the flow using $P$ and update $G_f$.
•     Endwhile.
•     Set $\Delta := \Delta/2$.
• Endwhile.
• Output $f$.
```

```
Algorithm 3: 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 $f$.
```

(a) 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.

(b) Explain why Algorithm 2 and Algorithm 3 are essentially the same algorithms (Hence the the number of augmentations in the fattest path algorithm is also at most $2m\lceil \log_2 K \rceil$).

Solution (sketch): Both algorithms always augment the fattest existing path.

3. (15 Points) Consider a minimum cut (A, B) in a flow network with exactly two edges e_1 and e_2 going from A to B.

(a) Prove: Decreasing the capacity of e_1 by 1 and meanwhile increasing the capacity of e_2 by 1 cannot increase the value of the maximum flow.

Solution: Decreasing the capacity of e_1 by 1 and increasing the capacity of e_2 by 1 does not change the capacity of the cut (A, B) which is also equal to the maximum flow. Hence the new max flow cannot exceed this (unchanged) capacity.

(b) Disprove: Decreasing the capacity of e_1 by 1 and meanwhile increasing the capacity of e_2 by 1 cannot decrease the value of the maximum flow.

Solution: You just need to give an example, and any example where there is a second min-cut which includes e_1 but not e_2 works.

4. (10 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.

Solution: Take $B = \{t\}$ and $A = V - \{t\}$. Then by the above fact

$$\text{val}(f) = f^{out}(A) - f^{in}(A) = f^{out}(A) = f^{in}(t).$$

5. (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 $f(e) = c_e$.

Solution: We have

$$\begin{aligned}\text{val}(f) &= f^{out}(A) - f^{in}(A) \le f^{out}(A) \le \\ &\sum_{e \text{ from } A \text{ to } B} f(e) \le \sum_{e \text{ from } A \text{ to } B} c_e = \text{cap}(A,B).\end{aligned}$$

Since (A,B) is a minimum cut and f is a maximum flow, we know $\text{val}(f) = \text{cap}(A,B)$. So all the above inequalities must be equalities. In particular $f(e) = c_e$ for all e from A to B.

6. (20 points) Let $(G, s, t, \{c_e\})$ be a flow network that satisfies the following property. For every edge e in G, there exists at least one minimum cut (A,B) such that e goes from A to B. Prove that G does not contain any directed cycles.

   Solution: Let f be a maximum flow. By the previous question and the assumption for all the edges e we have $f(e) = c_e$. Now suppose that G has a directed cycle. Decrease one unit of flow from every edge on this cycle and call this new flow f'. Note that since we did this to the edges on the cycle, f' is valid flow and furthermore $\text{val}(f') = \text{val}(f)$. So f' is also a maximum flow but this contradicts the previous question as there are edges for which $f'(e) < c_e$ (for every edge e on the cycle).

7. (20 points) Let $(G, s, t, \{c_e\})$ be a flow network, and let F be the set of all edges e for which 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.

   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;

8. (Bonus: 20 Points) Consider the variant of the maximum flow problem where every node v also has an integer capacity $c_v \ge 0$. 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}.