CSCI 3104 Algorithms Name: Jeremy M. Hein
Problem Set 8
Fall 2020, CU Boulder ID: 810952267

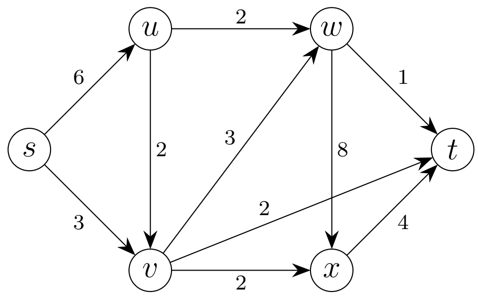
Instructions

- This problem set is **open book**: you may refer to the lectured material found on Canvas and the recommended books to help you answer the questions.
- This problem set is an **individual effort**. You must arrive at your answers independently and write them up in your own words. Your solutions should reflect your understanding of the content.
- **Posting questions to message boards or tutoring services including, but not limited to, Chegg, StackExchange, etc., is STRICTLY PROHIBITED. Doing so is a violation of the Honor Code.**
- Your solutions must be submitted typed in LaTeX, **handwritten work is not accepted**. If you want to include a diagram then we do accept photos or scans of hand-drawn diagrams included with an appropriate `\includegraphics` command. It is your responsibility to ensure that the photos you obtain are in a format that pdflatex understands, such as JPEG.
- The template tex file has carefully placed comments (% symbols) to help you find where to insert your answers. There is also a **STUDENT DATA** section in which you should input your name and ID, this will remove the warnings in the footer about commands which have not been edited. You may have to add additional packages to the preamble if you use advanced LaTeX constructs.
- You must CITE any outside sources you use, including websites and other people with whom you have collaborated. You do not need to cite a CA, TA, or course instructor.
- Take care with time, we do not usually accept problem sets submitted late.
- Take care to upload the correct pdf with the correct images inserted in the correct places (if applicable).
- Check your pdf before upload.
- **Check your pdf before upload.**
- **CHECK YOUR PDF BEFORE UPLOAD.**

Quicklinks: 1 (1a) (1b) 2

Problem 1

For both parts of Problem 1, use the following flow network.



(1a) Using the Ford–Fulkerson algorithm, compute the maximum flow that can be pushed from s to t, using $s \to u \to v \to t$ as your first augmenting path.

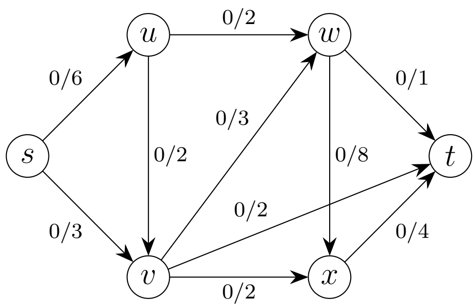
In order to be eligible for full credit you must include the following:

- The residual network for each iteration, including the residual capacity of each edge.
- The flow augmenting path for each iteration, including the amount of flow that is pushed through this path from $s \to t$.
- The updated flow network **after each iteration**, with flows for each directed edge clearly labeled.
- The maximum flow being pushed from $s \to t$ after the termination of the Ford-Fulkerson algorithm.

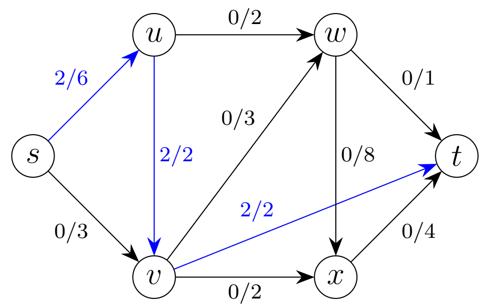
You may use `\includegraphics` to include hand-drawn flow networks or use the tikz picture below as inspiration for your own tikz diagrams. See Piazza for more details on tikz.

Before we start, note that I am using fractional notation to depict the capacities of edges in the flow and residual networks. The capacity of each edge must be between 0 and the capacity of the edge, and if an augmenting path utilizes a reverse flow, it deducts from the amount of flow along that edge. For example, 2/6 along edge $s \to u$ would indicate there are 4 units of flow available from s to u and 2 units of reverse flow available from u to s. I find this notation significantly easier to understand, and Professor Davies indicated we are free to use it as long as we make note of it for the graders.

We start with flow $f = 0$, and our first augmenting path is $s \to u \to v \to t$. We see that our lowest capacity edge along the path is two, so we can push two units of flow down this path:

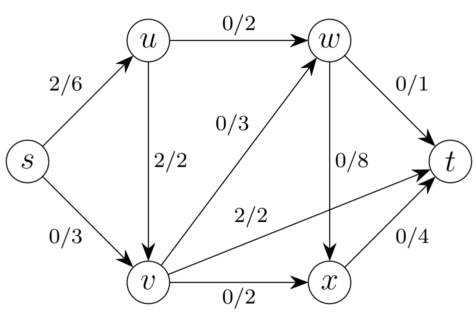


Start: $f = 0$

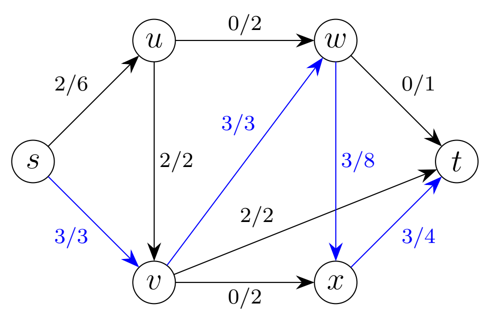


Residual Network: $f = 2$

Our current flow f is 2. The next augmenting path we will choose is $s \to v \to w \to x \to t$. Our lowest capacity edge along this path is 3, so we can push three units of flow along this path:

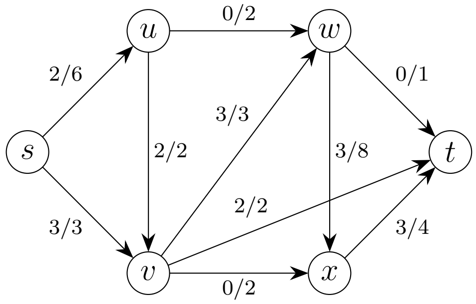


Flow Network: $f = 2$

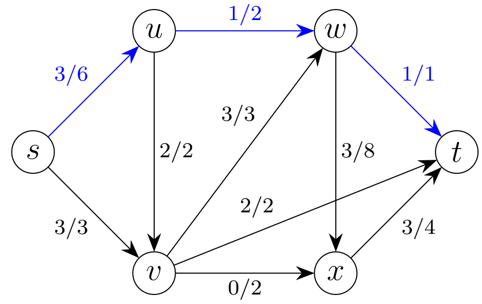


Residual Network: $f = 5$

Our current flow f is 5. The next augmenting path we will choose is $s \to u \to w \to t$. Our lowest capacity edge along this path is 1, so we can push one unit of flow along this path:

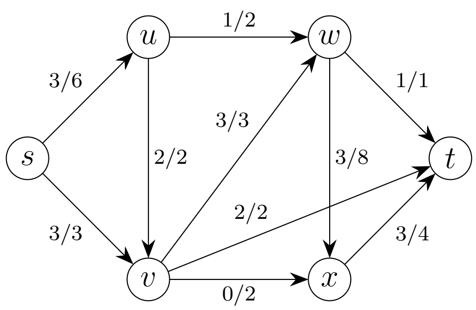


Flow Network: $f = 5$

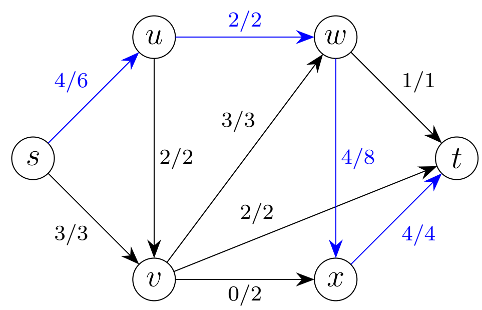


Residual Network: $f = 6$

Our current flow f is 6. The next augmenting path we will choose is $s \to u \to w \to x \to t$. Our lowest capacity edge along this path is 1, so we can push one unit of flow along this path:



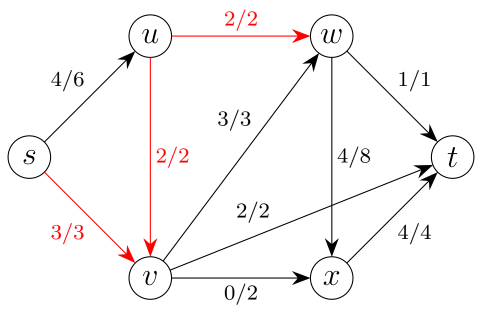
Flow Network: $f = 6$



Residual Network: $f = 7$

We note there are no additional augmenting paths we can choose. The only remaining capacity leaving s is via the edge su. From u, there is no capacity to move further towards sink t, so we are done. The maximum flow we can push from s to t is 7 units.

(1b) The Ford–Fulkerson algorithm terminates when there is no longer an augmenting path on the residual network. At this point, you can find a minimum cut of the form (S, T) where $s \in S$ and $t \in T$. Indicate this cut and its value.



Flow Network: $f = 7$

A minimum cut is indicated above in red, including edges $\{sv, uv, uw\}$. We can see from this cut that it satisfies the criteria that $s \in S$ and $t \in T$, $S \cap T = \emptyset$, and $S \cup T = V$. The cost of this cut is 7, which is equal to the flow we found in part 1(a), as desired.

Problem 2

In this problem we will consider a generalization of the Maximum (s, t)-Flow problem where we allow multiple sources and multiple sinks.

We define an (S, T)-flow network as $G = (V, E, c, S, T)$ where V is a set of vertices, E is a set of edges, c is a non-negative integer-valued function that assigns a capacity to each edge in E, S is a subset of the vertex set V and T is a subset of the vertex set V. In this problem, we may assume that all edges in E are directed and $S \cap T = \emptyset$. For convenience, we may also assume that f and c are defined for all $V \times V$ and not just E. That is, we assume $f(u, v) = c(u, v) = 0$ for all $(u, v) \notin E$. Our goal is the find the maximum amount of flow that can be sent from the vertices in S to the vertices in T. We define an (S, T)-flow as a non-negative integer-valued function f such that for all $v \in V \setminus S \setminus T$,

$$\sum_{u \in V} (f(u, v) - f(v, u)) = 0.$$

We also define the value of an (S, T)-flow f as

$$|f| = \sum_{s \in S} \sum_{u \in V} (f(s, u) - f(u, s)).$$

We still say that an (S, T)-flow is feasible if for all $e \in E$,

$$f(e) \leq c(e).$$

The Maximum (S, T)-Flow problem asks, given some (S, T)-flow network $G = (V, E, c, S, T)$, to output a feasible (S, T)-flow f with maximum value, $|f|$. Your goal in this problem is to solve this the Maximum (S, T)-Flow problem by *reducing* to the Maximum (s, t)-Flow problem we already know how to solve. Recall from lecture that the normal Maximum (s, t)-Flow problem assumes that $S = \{s\}$ for some $s \in S$ and $T = \{t\}$ for some $t \in V$.

In order to be eligible for full credit you must include the following:

- Describe a process that takes an (S, T)-flow network $G = (V, E, c, S, T)$ as input and outputs an (s, t)-flow network $G' = (V', E', c', s, t)$. Your process should describe how to construct V', E', c' and which vertices in V' are s and t.
- Argue that if there exists a feasible flow f with respect to G such that $|f| = k$ for some non-negative integer k, then there exists a feasible flow f' with respect to G' such that $|f'| = k$.

- Likewise, argue that if there exists a feasible flow f' with respect to G' such that $|f'| = k$ for some non-negative integer k, then there exists a feasible flow f with respect to G such that $|f| = k$.

- Finally, conclude by clearly explaining how you can use the above process and observations to solve the Maximum (S, T)-Flow problem.