Name: Vera Duong

ID: 109431166

**CSCI 3104, Algorithms** **Due March 19, 2021**

**Problem Set 8 (50 points)** **Spring 2021, CU-Boulder** Collaborators: Nathan Straub, Matt Hartnett

*Advice 1*: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.
*Advice 2*: Verbal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

**Instructions for submitting your solution**:

- The solutions **should be typed** and we cannot accept hand-written solutions. Here's a short intro to Latex.
- You should submit your work through **Gradescope** only.
- The easiest way to access Gradescope is through our Canvas page. There is a Gradescope button in the left menu.
- Gradescope will only accept **.pdf** files.
- It is vital that you match each problem part with your work. Skip to 1:40 to just see the matching info.

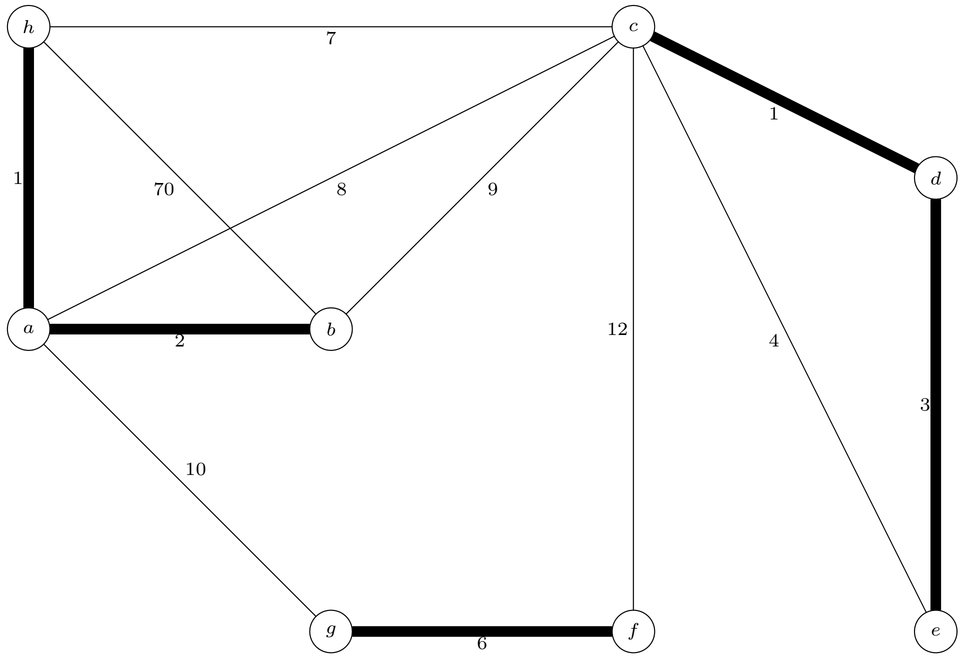
Name: Vera Duong

ID: 109431166

**CSCI 3104, Algorithms** **Due March 19, 2021**

**Problem Set 8 (50 points)** **Spring 2021, CU-Boulder** Collaborators: Nathan Straub, Matt Hartnett

1. We know it is important to find a safe edge in the generic minimum spanning tree algorithm. Consider the undirected, weighted graph $G$ shown below. The bold edges represent (the edges of) an intermediate spanning forest $F$, obtained by running the generic minimum spanning tree algorithm. The intermediate spanning forest $F$ has three components: $\{a, b, h\}$, $\{g, f\}$ and $\{c, d, e\}$. For each non-bold edge in the graph, determine whether the edge is **safe**, **useless** or **undecided**.



**Solution:**

$\{a, g\}$ is safe because it's the minimum edge that connects $\{a, b, h\}$ and $\{g, f\}$.

$\{h, b\}$ is useless because it connects its endpoints to $\{a, b, h\}$ which creates a cycle and not connect any forest components.

$\{h, c\}$ is safe because the forrest $\{a, b, h\}$ and $\{c, d, e\}$ are connected by the minimum edge of weight 7.

$\{b, c\}$ is undecided because it's not the minimum edge that connects $\{a, b, h\}$ and $\{c, d, e\}$ however it also does not create a cycle.

$\{c, a\}$ is undecided because it's not the minimum edge that connects $\{a, b, h\}$ and $\{c, d, e\}$ however it also does not create a cycle.

$\{c, f\}$ is undecided because it connects the forrest $\{g, f\}$ and $\{c, d, e\}$ but is not the minimum edge that connects to that componenet.

$\{c, e\}$ is useless because it connects its endpoints to $\{c, d, e\}$ which creates a cycle.

Name: Vera Duong

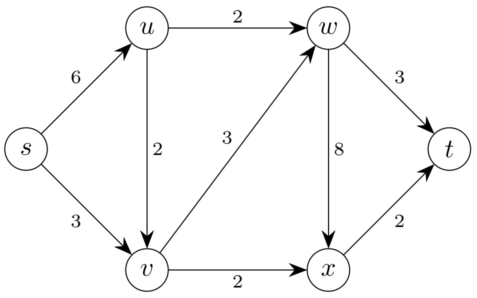
ID: 109431166

**CSCI 3104, Algorithms** **Due March 19, 2021**

**Problem Set 8 (50 points)** **Spring 2021, CU-Boulder** Collaborators: Nathan Straub, Matt Hartnett

2. **5 points extra credit to those who use Latex/Tikz to write up each step in their solution for Problems 2 and 3. You may hand draw your solution for this problem, however to receive the bonus points, you must have your solution nicely and neatly Latexed (we recommend Tikz).**

   For both parts of Problem 2, use the following flow network.



(2a) Using the Ford–Fulkerson algorithm, compute the maximum flow that can be pushed from $s$ to $t$, using $s \to u \to w \to t$ 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.

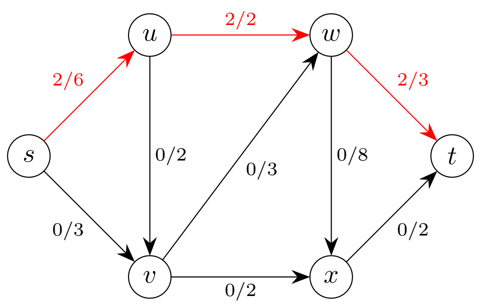
(2b) The Ford–Fulkerson algorithm terminates when there is no longer an augmenting path on the residual network. At this point, you can find a minimum cut of the form $(S, T)$ where $s \in S$ and $t \in T$. Indicate this cut and its value.

**Solution:**

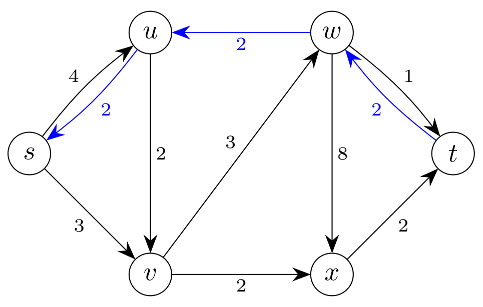
Go to next page for solution to problem 2!

(2a)

Iteration 1: $s \to u \to w \to t$

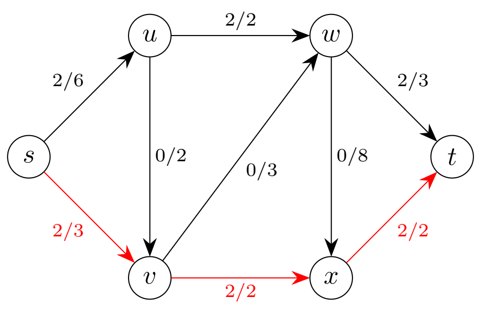


Bottleneck = 2

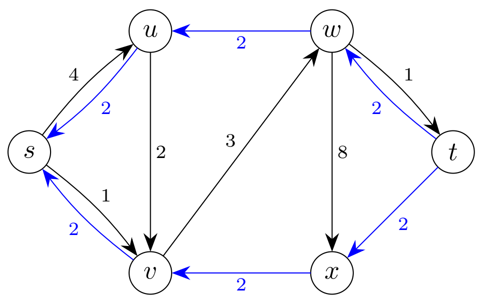


Residual Graph for 1st iteration

Iteration 2: $s \to v \to x \to t$

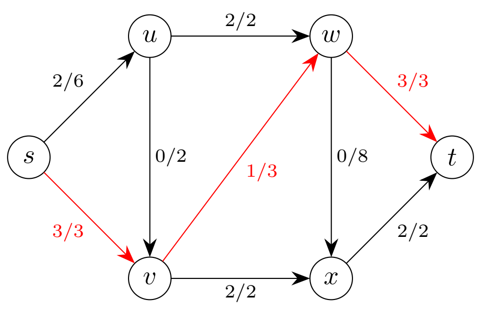


Bottleneck = 2

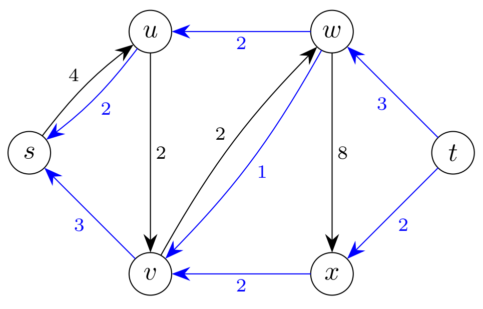


Residual Graph for 2nd iteration

Iteration 3: $s \to v \to w \to t$



Bottleneck = 1



Residual Graph for 3rd iteration

After iteration 3 we no longer have a simple path from $s \to t$ so we can calculate our max flow by adding all the bottleneck values: 2 + 2 + 1 = **5** is our max flow.

(2b) The max flow min cut theorem states that the max flow should also equal the value of the min cut. In this case, we have $S = \{s, u, v, w, x\}$ and $T = \{t\}$. This gives us a cut value of $3 + 2 = \mathbf{5}$ where the cut is between $(w, t)$ and $(x, t)$. We can also find the set $S$ by including the vertices $s \to t$ within the residual graph, and the vertex $t$ isn't included so we know where to make the cut.

Name: Vera Duong

ID: 109431166

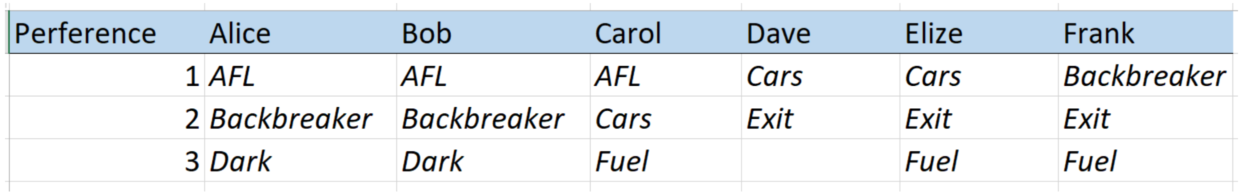
**CSCI 3104, Algorithms** **Due March 19, 2021**

**Problem Set 8 (50 points)** **Spring 2021, CU-Boulder** Collaborators: Nathan Straub, Matt Hartnett

3. **5 points extra credit to those who use Latex/Tikz to write up each step in their solution for Problems 2 and 3. You may hand draw your solution for this problem, however to receive the bonus points, you must have your solution nicely and neatly Latexed (we recommend Tikz).**

   A video game store is selling video games to different customers. Each of the customers only has the money to buy at most one video game. Due to the low storage of the video games, each of the different video games only has one copy available for sale. The video game store decides to run an algorithm to make sure the maximum number of customers can buy the game.
   Example - Following is one such preference of each customer. If the games all get sold according to customers' first preference, only 3 games will be sold. But a better sale strategy is Alice - *AFL*, Bob - *Dark*, Carol - *Cars*, Dave - *Exit*, Elize - *Fuel*, Frank - *Backbreaker* and this gets 6 game sold.

   Help them come up with an algorithm to find a sale strategy that gets the maximum games sold using Ford-Fulkerson.



| Perference | Alice | Bob | Carol | Dave | Elize | Frank |
|---|---|---|---|---|---|---|
| 1 | *AFL* | *AFL* | *AFL* | *Cars* | *Cars* | *Backbreaker* |
| 2 | *Backbreaker* | *Backbreaker* | *Cars* | *Exit* | *Exit* | *Exit* |
| 3 | *Dark* | *Dark* | *Fuel* | | *Fuel* | *Fuel* |

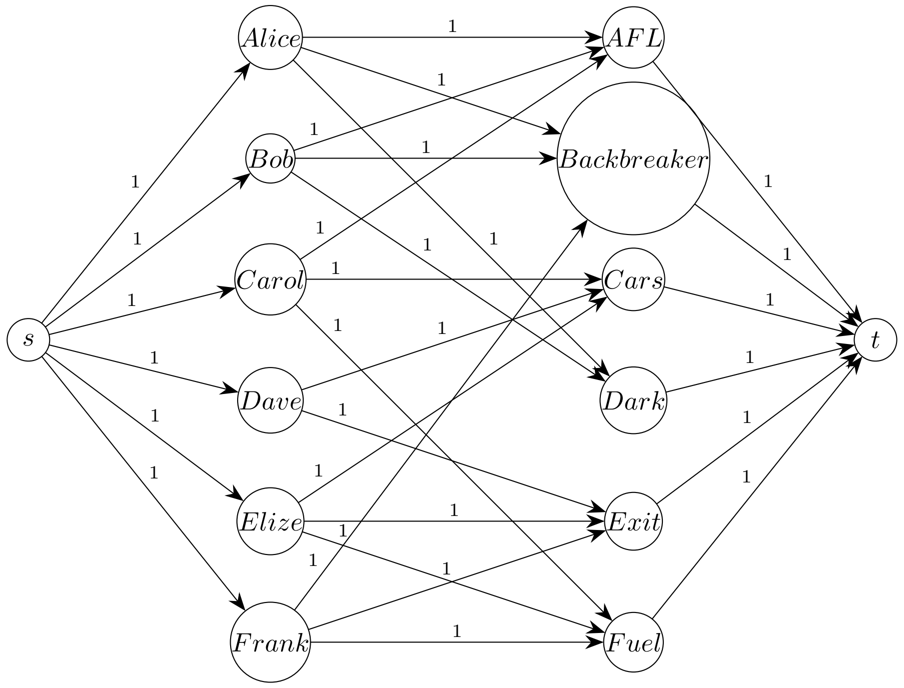
   (a) Draw a network $G$ to represent this problem as a flow maximization problem for the example given above. Clearly indicate the source, the edge directions, the sink/target, and the capacities, and label the vertices.

   (b) Assume that you have access to Fork-Fulkerson sub-routine called **Ford-Fulkerson(G)** that takes a network and gives out max-flow in terms of f(e) for all the edges. How will you use this sub-routine to find the maximum games sold. Clearly explain your solution.

**Solution**:

Go to next page for solution to problem 3!

(3a)



(3b) In the graph, every edge is 1 since a person can only buy at most one game. We assume that Ford-Fulkerson(G) was done correctly and gives out the max-flow for all the edges. We can see the flow/capacity for each edge and can then determine the maximum games sold by seeing whether or not there is flow in the given edge. If an edge has a flow/capacity of 0/1, we know that edge was not included in the paths to find the max-flow and can eliminate the edge between the person and game in the max flow calculation. For example if the edge (Alice, Dark) had 0/1 for flow/capacity, we don't count that in for max games sold. If we had the edge (Alice, AFL) = 1/1, this would count for our calculation in max games sold since that path was taken. We would then sum up the flows of all edges from every person where their flow/capacity is 1/1. In this case, we would have max flow of 6 where it's the maximum number of games sold. Furthermore, another method is that we can also sum up the flows going out of $s$ since they connect to one person, and one person can buy at most one game which maximizes the games sold to 6, and the flows going out of $s$ should equal the flows going into $t$.