Name: Chirag Telang

ID: 109410601

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

Problem Set 8 (50 points) **Spring 2021, CU-Boulder**

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.

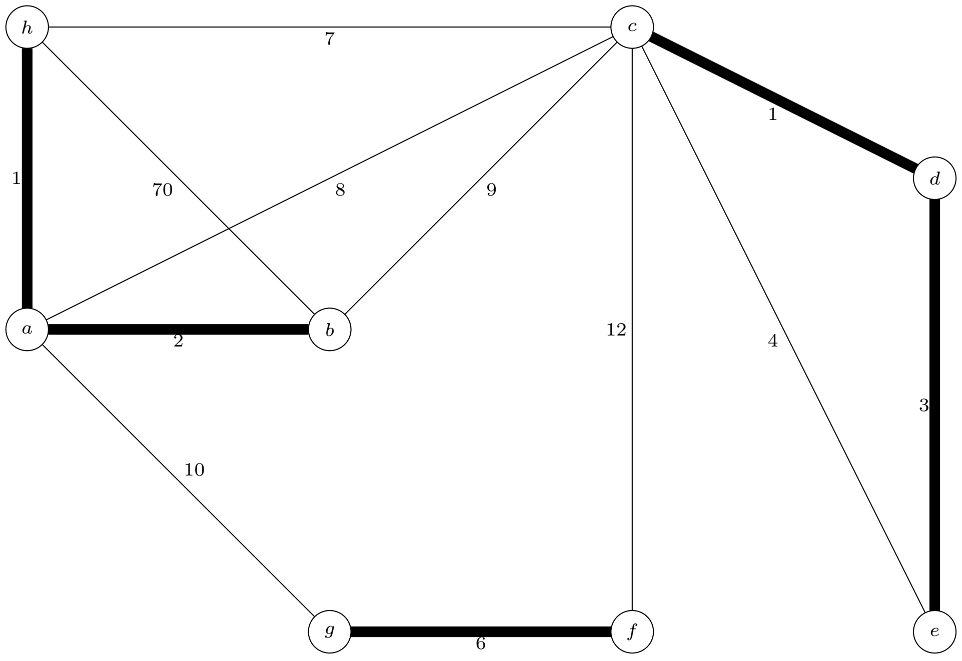
Name: Chirag Telang

ID: 109410601

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

Problem Set 8 (50 points) **Spring 2021, CU-Boulder**

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:

$\{h, c\}$ – safe

$\{a, g\}$ – safe

$\{c, e\}$ – useless

$\{b, h\}$ – useless

$\{c, f\}$ – undecided

$\{b, c\}$ – undecided

$\{a, c\}$ – undecided

Name: Chirag Telang

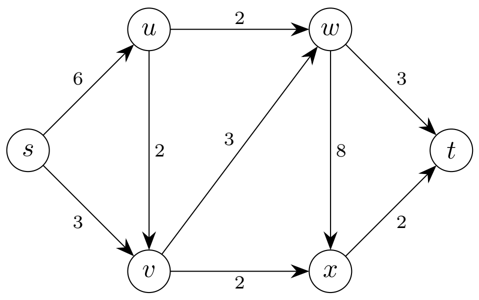
ID: 109410601

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

Problem Set 8 (50 points) **Spring 2021, CU-Boulder**

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.

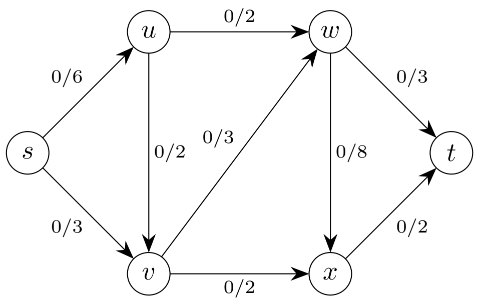
In order to be eligible for full credit you must include the following:

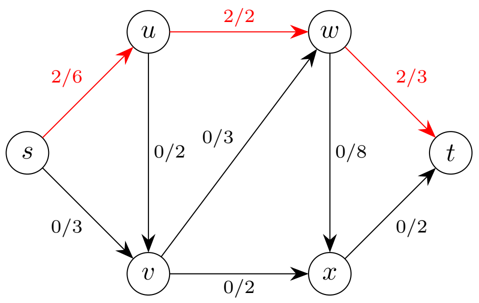
- The residual network for each iteration, including the residual capacity of each edge.
- The flow augmenting path for each iteration, including the amount of flow that is pushed through this path from $s \to t$.
- The updated flow network **after each iteration**, with flows for each directed edge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

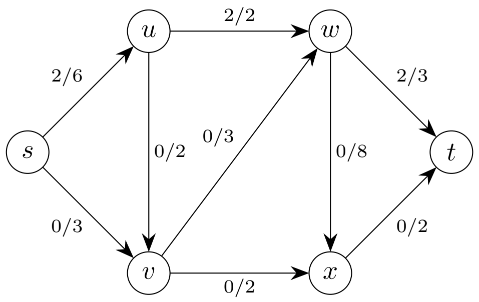
Solution:

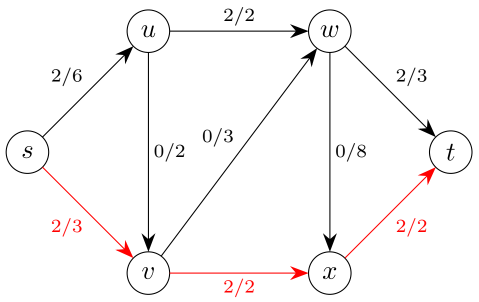
(a) Starting with flow $f = 0$, our first augmenting path is $s \to u \to w \to t$. The lowest capacity edge is 2, so we can push 2 units of flow down the selected path. The final residual network will have flow $f = 2$.



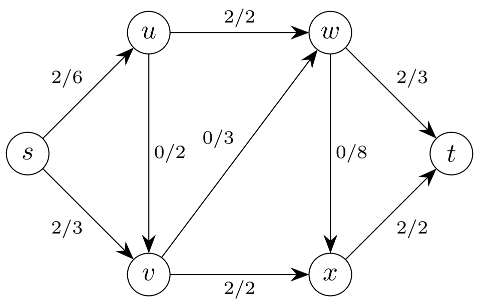


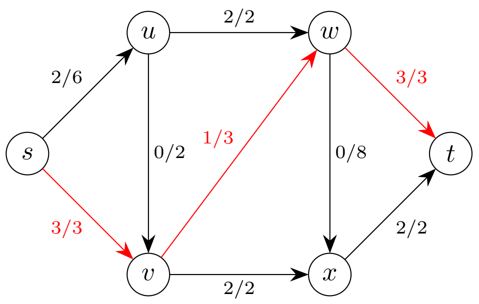
The current flow is $f = 2$, and our next augmenting path is $s \to v \to x \to t$. The lowest capacity edge is 2, so we can push 2 units of flow down the selected path. The final residual network will have flow $f = 4$.





The current flow is $f = 5$, and our next augmenting path is $s \to v \to w \to t$. The lowest capacity edge is 1, so we can push 1 unit of flow down the selected path. The final residual network will have flow $f = 5$.





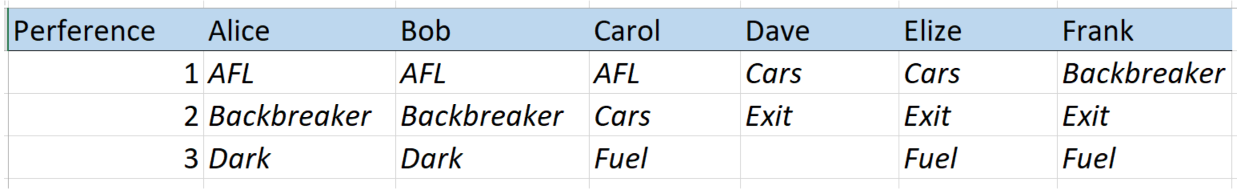
From here, there are no additional augmented paths we cane choose, since the only remaining capacity leaving s is through $u \to v \to w$. From w, there's no remaining capacity to move further so, therefore, we are finished. The maximum flow we can push from $s \to t$ is 5 units of flow.

(b) A minimum cut includes the edges (w,t) and (x,t). This cut satisfies the notion that $s \in S$ and $t \in T$, $S \cup T = V$, and $S \cap T = \emptyset$. The cost of this cut is 5, which is equal to the flow found in part A.

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.



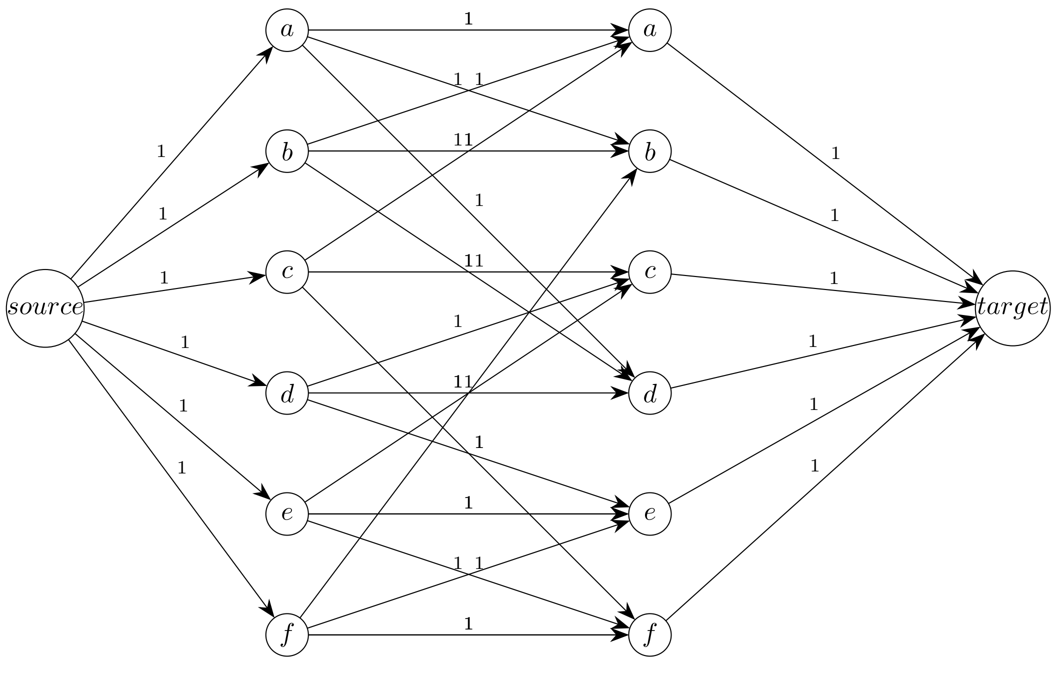
Perference	Alice	Bob	Carol	Dave	Elize	Frank
1	*AFL*	*AFL*	*AFL*	*Cars*	*Cars*	*Backbreaker*
2	*Backbreaker*	*Backbreaker*	*Cars*	*Exit*	*Exit*	*Exit*
3	*Dark*	*Dark*	*Fuel*		*Fuel*	*Fuel*

   (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:

   (a) Below is a network G where the left-side nodes are the **names** and the right-side nodes are the **games**.



(b) Using the Ford-Fulkerson (G) subroutine, we know that the max flow is 6 – since all the edges going towards the target and going out from the source have a capacity of 1. In regards to the vertices between the name and game nodes, only 6 vertices can have a flow of 1 at any given time, meaning that the rest of the vertices will have a flow of 0. Therefore, the maximum of games sold is 6.