Caleb Logemann
MATH 566 Discrete Optimization
Midterm II

1. *Girth* of a graph $G = (V, E)$ is the length of the shortest cycle. Notice cycle in undirected graph has at least 3 vertices. Formulate an integer program that is solving the problem of shortest cycle in a graph.

   In order to create this integer program, I will create a binary variable for each vertex in the graph G. Let $x_v \in [0, 1]$, denote whether or not vertex v is in the shortest cycle. Note that if v is in the shortest cycle then at least two of v's neighbors must also be in the cycle. If only one neighbor was in the cycle, then there would be a path into v but not out of v, so it could not be a cycle. If there weren't any neighbors in the cycle, then v would be disconnected from the cycle. This reasoning can be expressed in the following constraint.

   $$\sum_{u \in N(v)} (x_u) \geq 2$$

   where $N(v)$ is the set of vectors that are neighbors of v. This is also assuming that v is in the cycle so $x_v = 1$. If v is not in the cycle then $x_v = 0$ and there may or may not be neighbors of v in the cycle. This does not require a constraint but a trivial constraint can be used to describe this situation.

   $$\sum_{u \in N(v)} (x_u) \geq 0$$

   This is clearly true because all x_u are binary variables. Now it is necessary to be able to describe both of these situations depending on the value of x_v. Note that if $x_v = 0$, then $2x_v = 0$ and if $x_v = 1$, then $2x_v = 2$. Using this fact both constraints can be expressed as

   $$\sum_{u \in N(v)} (x_u) \geq 2x_v$$

   This constraint forces there to be two neighbors in the cycle if v is in the cycle, but doesn't enforce anything if v is not in the cycle.

   Now in order to find the girth of a graph, we wish to detect the smallest cycle. Therefore the integer program should minimize the number of vertices in the cycle or equivalently the objective function should be

   $$\text{minimize} \sum_{v \in V} x_v$$

   Just using the constraints given above and this objective function the integer program will simply set all x_v equal to zero. In order to force the program to find a nontrivial cycle we must force that at least 3 vertices are in the cycle. No cycle and can created with 0, 1, or 2 vertices. This constraint can be written as

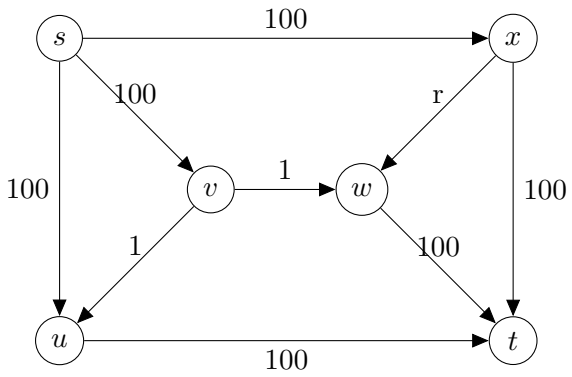
   $$\sum_{v \in V} x_v \geq 3$$

   Therefore the full integer program is

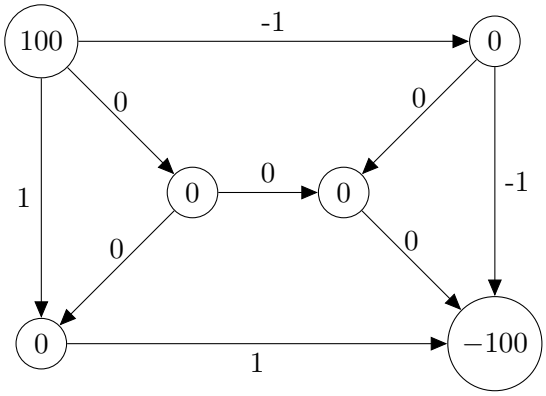
   $$(P) = \begin{cases} \text{minimize} & \sum_{v \in V} x_v \\ \text{subject to} & \sum_{v \in V} (x_v) \geq 3 \\ & \sum_{u \in N(v)} (x_u) \geq 2x_v \quad \forall v \in V \\ & x_v \geq 0 \quad \forall v \in V \\ & x_v \leq 1 \quad \forall v \in V \\ & x_v \in \mathbb{Z} \quad \forall v \in V \end{cases}$$

2. Recall that in Minimum Cost Flow, we were augmenting on minimum mean cycle. Show that if we allow augmentation on any augmenting cycle, the algorithm will not work right (bad convergence).

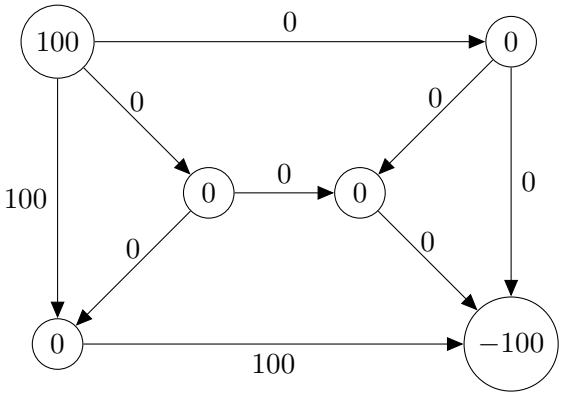
   Consider the following directed graph where $r = \frac{\sqrt{5}-1}{2}$ with capacities shown



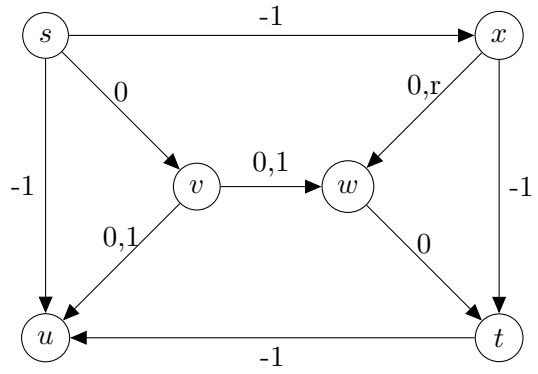
   The costs are shown in the following graph, along with the sources and sinks of each vertex



   I will start with the following initial flow which has cost 200.

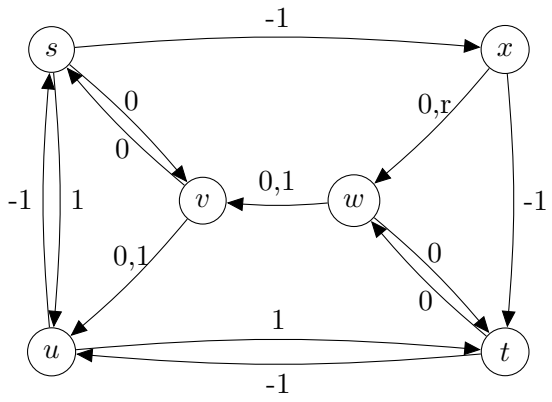


   The initial residual graph with (costs, residual capacity) shown is given below.

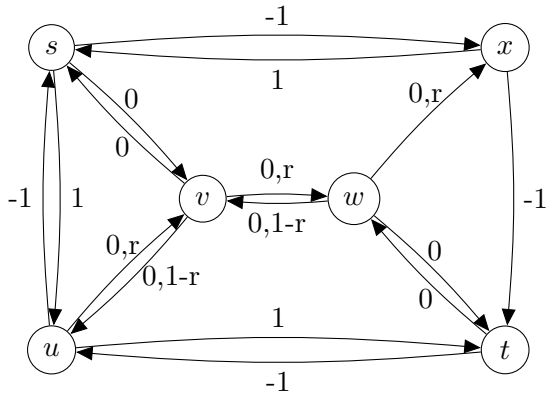


Clearly the minimum mean cost cycle is $s \to x \to t \to u \to s$ which if augmented on would find the minimum cost flow with value -200. However if the minimum mean cost cycle is selected, there exists a sequence of augmenting cycles that can be selected to never approach the minimum cost flow.

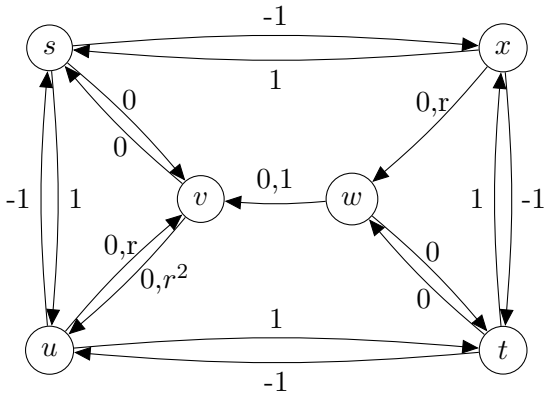
Consider the negative cycle, $c_1 = s \to v \to w \to t \to u \to s$. This cycle has mean cost $-2/5$ and minimum capacity 1. Augmenting on this path gives a new residual graph shown below.



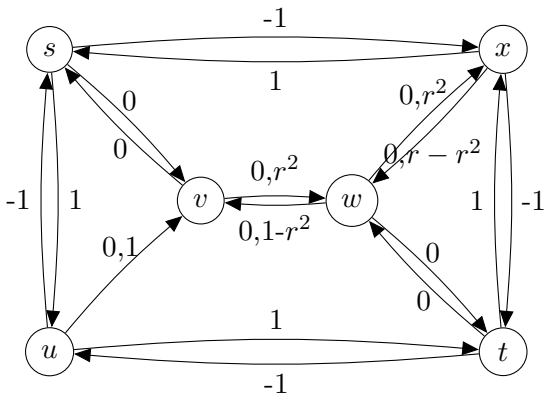
Next augment on $c_2 = s \to x \to w \to v \to u \to s$, with mean cost $-2/5$ and min capacity r. The new residual graph is



Now augment on $c_3 = s \to v \to w \to x \to t \to u \to s$, with mean cost $-1/2$ and min capacity r. The new residual graph is now shown below.



Next augment on c_2 again with mean cost $-2/5$ and min capacity r^2. The residual graph is now



Finally augment on $c_4 = u \to v \to w \to t \to u$ with mean cost $-1/4$ and min capacity r^2. If this list of cycles, c_2, c_3, c_2, c_4, is repeated the flow can be augmented indefinitely. This is shown in more detail with the table below.

cycle	mean cost	min cap	(v,u)	(v,w)	(x,w)	(s,x)
			1	1	r	100
$c_1 = svwtus$	$-2/5$	1	1	0	r	100
$c_2 = sxwvus$	$-2/5$	r	1 - r	r	0	100 - r
$c_3 = svwxtus$	$-1/2$	r	r^2	0	r	100 - r
$c_2 = sxwvus$	$-2/5$	r^2	0	r^2	$r - r^2$	$100 - r - r^2$
$c_4 = uvwtu$	$-1/4$	r^2	r^2	0	r^3	$100 - r - r^2$
c_2	$-2/5$	r^3	$r^2 - r^3$	r^3	0	$100 - r - r^2 - r^3$
c_3	$-1/2$	r^3	r^4	0	r^3	$100 - r - r^2 - r^3$
c_2	$-2/5$	r^4	0	r^4	$r^3 - r^4$	$100 - r - r^2 - r^3 - r^4$
c_4	$-1/4$	r^4	r^4	0	r^5	$100 - r - r^2 - r^3 - r^4$

As this table shows this repitition of augmentation cycles will continue indefinitely. Also this set of augmentation cycles does not approach the optimal flow. The table shows that the residual on (s,x) approaches $100 - \sum_{n=1}^{\infty}(r^n)$. This value can be computed using geometric series, but it suffices to notice that this value is greater than zero. It was shown earlier that the optimal flow on this network uses all of the capacity of this edge. Since this residual is not zero, the limit of this process is not the optimal flow.

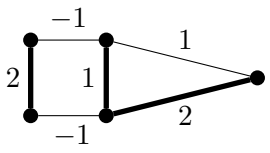
Therefore we can conclude that if the minimum mean cost cycle is not selected the min cost flow algorithm may not terminate and may not approach the optimal solution.

3. Suppose you have a graph $G = (V, E)$. Every edge is assigned a cost $c : E \to \mathbb{R}$. How to pick edges, such that the sum of costs of the picked edges is maximized and the picked edges do not contain a cycle (i.e., the set of picked edges forms a forest)? More formally,

$$\text{maximize}\left\{\sum_{e\in X} c(e) : X \subseteq E, X \text{ has no cycle}\right\}.$$

Describe how to use one/some of the algorithms we had in class to solve this problem.
(I mean, how to modify the input to this problem such that it can be solved by some other algorithm? Alternatively, you can also describe an algorithm.)

Example: Notice that the edges can have negative weight. The thick edges in the following graph show one of the optimal solutions. (There are two optimal solutions. Your algorithm should find one.)



In order to solve this problem, consider the minimum spanning tree on the graph G with cost function $-c$, that is the cost of all the edges have their sign flipped. If this problem is solved using any of the known minimum spanning tree algorithms, then it has already been shown that this tree has the smallest cost. This is equivalent to the maximum cost spanning tree in the original problem. However the original problem didn't specify that a single tree was needed, only that there were no cycles. It is known that adding any edge to a spanning tree will introduce a cycle, so no edge can be added to this spanning tree to increase this cost. However removing edges will not introduce a cycle, therefore any edges that decrease the overall value of the spanning tree should be removed. In other words all edges with negative cost in the original problem should be removed from the set X. As we have already seen no edge can be added to this set and this remains true after removing the negative cost edges. If a positive cost edge could have been used instead of a negative cost edge, then the spanning tree algorithm would have added it to the tree.

In summary, let Y be the set of edges in the minimum spanning tree of $G = (V, E)$ with cost function $-c : E \to \mathbb{R}$. Also let Z be the set of edges with negative cost that is

$$Z = \{e : c(e) < 0\}$$

Thus the solution to the problem is

$$X = Y - Z$$

that is the edges of Y with any edges in Z removed.