Caleb Logemann
MATH 566 Discrete Optimization
Homework 9

1. Implement Directed Minimum Mean Cycle in Sage. Before implementing the algorithm, show that it is possible to slightly modify the algorithm. Instead of adding an extra vertex s and edges from s to all other vertices, it is possible to simply assign $F_0(v) = 0$ for all $v \in V$ at the beginning. This avoids the hassle with adding an extra vertex. But it requires an argument that the algorithm is still correct.

   In the initial algorithm in the notes we consider a vertex s from which all other vertices can be can be reached. If instead we consider potentially starting at any vertex, we are not required to introduce a new vertex. This also implies that $F_0(v) = 0$ for all vertices because there always exists a walk of length zero from a vertex to itself, the empty walk. In this new setup $F_k(v)$ now represents the minimum cost walk of length k from any vertex to v. The rest of the algorithm works as intended, the vertex that realizes
$$\min_{v \in V} \max_{k, F_k(v) \neq \infty} \frac{F_n(v) - F_k(v)}{n-k}$$
   will be on the minimum mean cost cycle.

   The following function performs the minimum mean cost cycle algorithm as described in the notes with the modification described above.

```
def minimumMeanCostCycle(graph):
    n = graph.order()
    p = {v:[None]*(n+1) for v in graph.vertices()}
    F = {v:[0]+[oo]*n for v in graph.vertices()}
    for k in range(1,n+1):
        for e in graph.edges():
            u = e[0]
            v = e[1]
            c = e[2]
            if F[v][k] > F[u][k-1] + c:
                F[v][k] = F[u][k-1] + c
                p[v][k] = u

    # If all are infinite then min will be infinite
    if min([F[v][n] for v in graph.vertices()]) == oo:
        print 'This graph is acyclic'
        return (None, None)

    m = {v:oo for v in graph.vertices()}
    for v in graph.vertices():
        if F[v][n] != oo:
            m[v] = max([(F[v][n] - F[v][k])/(n-k) for k in range(n-1)])

    x = min(m, key=m.get)
    mu = m[x]

    # create cycle
    cycle = DiGraph([graph.vertices(), []])
    v = x
    for k in range(n):
        u = v
        v = p[v][n - k]
        cycle.add_edge(v, u, graph.edge_label(v, u))

    return (cycle, mu)
```

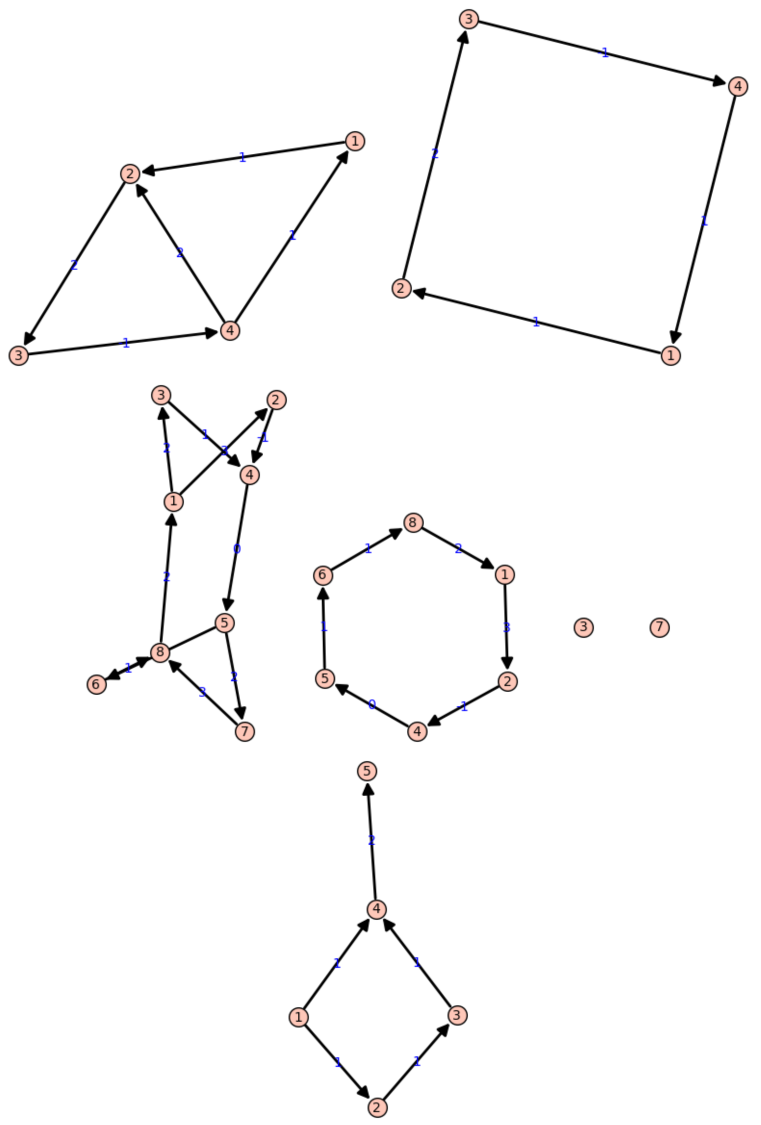
This script runs this algorithm on 3 test graphs.

```
load('minimumMeanCostCycle.sage')
g = DiGraph([(1,2,1), (2,3,2), (3,4,-1), (4,1,1), (4,2,2) ])
g.show(edge_labels=True)
(cycle,mu) = minimumMeanCostCycle(g)
if cycle != None:
    print "Mu=",mu
    cycle.show(edge_labels=True)


g = DiGraph([(1,2,3), (1,3,2), (2,4,-1), (3,4,1), (4,5,0), (5,6,1),
   ↪ (5,7,2), (6,8,1), (7,8,3), (8,1,2) ])
g.show(edge_labels=True)
(cycle,mu) = minimumMeanCostCycle(g)
if cycle != None:
    print "Mu=",mu
    cycle.show(edge_labels=True)


g = DiGraph([(1,2,1), (2,3,1), (3,4,1), (1,4,1), (4,5,2) ])
g.show(edge_labels=True)
(cycle,mu) = minimumMeanCostCycle(g)
if cycle != None:
    print "Mu=",mu
    cycle.show(edge_labels=True)
```

The output of this script is shown below.

```
Launched png viewer for Graphics object consisting of 15 graphics primitives
Mu= 3/4
Launched png viewer for Graphics object consisting of 13 graphics primitives
Launched png viewer for Graphics object consisting of 29 graphics primitives
Mu= 1
Launched png viewer for Graphics object consisting of 19 graphics primitives
Launched png viewer for Graphics object consisting of 16 graphics primitives
This graph is acyclic
```

The scipt also produces the following images.



2. Show that in integer program, it is possible to express the following constraint:

$$x \in [100, 200] \cup [300, 400]$$

in other words

$$100 \leq x \leq 200 \text{ or } 300 \leq x \leq 400$$

How to express the constraint *without* using *or*?

First of all it can be see that $x \geq 100$ and $x \leq 400$ unconditionally as both intervals satisfy these conditions. Assuming these inequalities the system can be simplified to either $x \leq 200$ or $x \geq 300$. In order to express this or statement as linear constraints consider $z \in \mathbb{Z}$ such that $0 \leq z \leq 1$. Now the variable z is a binary variable that will turn on only one of the previous constraints. Consider the system

$$\begin{aligned} x &\leq 200 + Mz \\ x &\geq 300 - M(1 - z) \end{aligned}$$

where M is sufficiently large. In this case $M = 200$ is sufficient. Consider when $z = 0$, in this case these constraints become

$$\begin{aligned} x &\leq 200 \\ x &\geq 100 \end{aligned}$$

This corresponds to $x \in [100, 200]$ Consider the case when $z = 1$, then the constraints become

$$\begin{aligned} x &\leq 400 \\ x &\geq 300 \end{aligned}$$

This corresponds to $x \in [300, 400]$. In fact when $M = 200$, these constraints don't even require the upper and lower bounds stated earlier. This constraint can be fully expressed as

$$\begin{aligned} x &\leq 200 + 200z \\ x &\geq 300 - 200(1 - z) \\ z &\geq 0 \\ z &\leq 1 \\ x, z &\in \mathbb{Z} \end{aligned}$$

This system is identical to the statement $x \in [100, 200] \cup [300, 400]$ and $x \in \mathbb{Z}$.

3. Determine which of the matrices below are (i) unimodular, (ii) totally unimodular, or (iii) neither. Be sure to explain your answer.

$$\begin{pmatrix} 1 & -1 & -1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

a. **b.** **c.**

(i) Since these are all square matrices, they will be unimodular if their determinant is ± 1. So I compute the determinants of these 3 matrices. First I will compute the determinant of (a).

$$\det(a) = \begin{vmatrix} 1 & -1 & -1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{vmatrix}$$

Expand along first column

$$\det(a) = 1 \times \begin{vmatrix} 0 & 0 & 1 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \end{vmatrix} - (-1) \times \begin{vmatrix} -1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \end{vmatrix}$$

Expand along the first column for the first determinant and along the last column for the second determinant

$$\begin{aligned} &= 1 \times \left(-1 \times \begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix} \right) - (-1) \times \left(1 \times \begin{vmatrix} -1 & -1 \\ 0 & 1 \end{vmatrix} \right) \\ &= 1 \times (-1 \times -1) - (-1) \times (1 \times -1) \\ &= 1 - 1 \\ &= 0 \end{aligned}$$

Since the determinant of (a) is 0 this matrix is not unimodular.
Second I will compute the determinant of (b).

$$\det(b) = \begin{vmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{vmatrix}$$

I will first expand along the first row.

$$\det(b) = 1 \times \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{vmatrix} + 1 \times \begin{vmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{vmatrix}$$

I will expand the first determinant along the first row and the second determinant along the first column

$$\begin{aligned} \det(b) &= 1 \times \left(1 \times \begin{vmatrix} 1 & 1 \\ 0 & 1 \end{vmatrix}\right) + 1 \times \left(1 \times \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix}\right) \\ &= 1 \times (1 \times 1) + 1 \times (1 \times 1) \\ &= 1 + 1 \\ &= 2 \end{aligned}$$

Since the determinant of (b) is 2 this matrix is not unimodular.
Lastly I will compute the determinant of (c).

$$\det(c) = \begin{vmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{vmatrix}$$

First I will expand along the first column

$$\det(c) = -1 \times \begin{vmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{vmatrix}$$

Next I will expand along the first row

$$\det(c) = -1 \times \left(1 \times \begin{vmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{vmatrix}\right)$$

Expanding along the first column gives

$$\begin{aligned} \det(c) &= -1 \times \left(1 \times \left(-1 \times \begin{vmatrix} 1 & 1 \\ 1 & 0 \end{vmatrix} + 1 \times \begin{vmatrix} 1 & 1 \\ 0 & 1 \end{vmatrix}\right)\right) \\ &= -1 \times (1 \times (-1 \times -1 + 1 \times 1)) \\ &= -1 \times (1 \times (1 + 1)) \\ &= -1 \times (1 \times 2) \\ &= -1 \times 2 \\ &= -2 \end{aligned}$$

Since the determinant of (c) is -2 this matrix is not unimodular.

In summary none of the matrices are unimodular.

(ii) In order for a matrix to be totally unimodular every square submatrix must have determinant -1, 0, or 1. Note that since matrices (b) and (c) don't have a determinant -1, 0, or 1 when considered as a whole matrix they cannot be totally unimodular. Matrix (a) which has determinant 0 can potentially be totally unimodular. In fact we see that each column has exactly one 1 and one -1, so by a theorem in the notes (a) is totally unimodular.

(iii) We have shown that (a) is totally unimodular but not unimodular. However (b) and (c) are neither unimodular nor totally unimodular.

4. Show that $A \in \mathbb{Z}^{m\times n}$ is totally unimodular iff $[A\ I]$ is unimodular (where I is $m \times m$ unit matrix).

Proof. Let $A \in \mathbb{Z}^{m\times n}$ and let I be the $m \times m$ identity matrix. Let A be totally unimodular. This implies that any square submatrix has determinant 0, 1, or -1. Consider a $m \times m$ basis of $[A\ I]$. Clearly any basis that contains only columns of A has determinant either 1 or -1, as A is totally unimodular. Now let this basis contain some columns of I. Let A_b be the columns of A in the basis and let I_b be the columns of I in the basis. Now this basis can be thought of as the square matrix $[A_b\ I_b]$. Let $K = \{k : e_k \in I_b\}$, which is the set of indices of vector in I that are contained in I_b. Thus if $1 \in K$, then $e_1 = [1, 0, 0, \cdots]^T \in I_b$. Also let K^c be the set of indices that are not in K. Note that $|K^c|$ is equal to the number of columns of A_b. When taking the determinant of $[A_b\ I_b]$, it is possible to expand along the columns of I_b, so that

$$\det([A_b\ I_b]) = \pm 1 \det(A_b[K^c])$$

where $A_b[K^c]$ is the submatrix created by taking the rows of A_b indexed by K^c. Note that this is a square submatrix of A. As A is totally unimodular, this implies that $\det(A_b[K^c]) = \pm 1, 0$. Since this is a basis, the determinant can't be zero or $\det([A_b\ I_b]) \neq 0$. Therefore $\det(A_b[K^c]) \neq 0$, and this implies that

$$\det([A_b\ I_b]) = \pm 1 \times \pm 1 = \pm 1$$

Also note that if the basis of $[A\ I]$ is I, then $\det(I) = 1$. Thus any basis of $[A\ I]$ has a determinant ± 1, so $[A\ I]$ is unimodular.

Now assume $[A\ I]$ is unimodular. Consider any square submatrix of A determined by rows $M = \{i_k\}$ and columns $N = \{j_k\}$. Consider a subset of the columns of I determined by M^c, that is let $I_b = [e_i : i \in M^c]$. Also let $A_b = [a_j : jinN]$, be the columns of A determined by N. Then the square matric $[A_b\ I_b]$ is either a basis of $[A\ I]$ in which case the determinant is ± 1 or $[A_b\ I_b]$ is not a basis of $[A\ I]$ in which case the determinant is 0. It was shown in the last argument that the

$$\det([A_b\ I_n]) = \pm 1 \det(A[M, N])$$

where $A[M, N]$ is the square submatrix. Equivalently

$$\det(A[M, N]) = \pm 1 \det([A_b\ I_n])$$

Given the three possibilities of the deteminant of $[A_b\ I_b]$, it can be seen that $\det(A[M, N]) = 0, \pm 1$. Thus A is totally unimodular as any square submatrix has determinant 0 or ± 1. □

5. Find a unimodular matrix A, that is not totally unimodular.

Consider the matrix

$$A = \begin{matrix} 9 & 7 \\ 5 & 4 \end{matrix}$$

The matrix A is unimodular because $A \in \mathbb{Z}^{2\times 2}$ and $\det(A) = 9 \times 4 - 5 \times 7 = 36 - 35 = 1$. However A is not totally unimodular because not all of the entries of A are -1, 0, 1.