Homework 4

Sahil Palnitkar
CSCE 411

24-3)

a)

We can use the Bellman Ford Algorithm to sole this. It returns a Boolean which is true if there is not a negative weighted cycle and false if there is a negative weighted cycle. We can use product rule of log such that the log of the product is the sum of the log of the individual numbers. There is one vertex V in the directed graph G = (V,E) for each currency. For each pair, there are directed edges (vi, vj) and (vj and vi).

Using log rules:

R[i1, i2] x R[i2, i3] x … x R[ik-1, ik] x R[ik, i1] > 1
log(R[i1, i2] x R[i2, i3] x … x R[ik-1, ik] x R[ik, i1]) > log(1)
log(R[i1, i2]) + log(R[i2, i3]) + … + log(R[ik-1, ik]) + log(R[ik, i1]) > 0

Since we're looking for negative weight, we multiple both sides by -1.
- log(R[i1, i2]) - log(R[i2, i3]) - … - log(R[ik-1, ik]) - log(R[ik, i1]) < 0

```
Algorithm to check if there is a negative weighted cycle()
n = card(V)
distance = array[0,n][0,n]
for i = 0 to n-1
        for j = 0 to n-1
                if (i,j) in E
                        distance[i][j] = c(i,j)
                else
                        distance[i][j] =+ inf

for k = 0 to n-1
        for I = 0 to n-1
                for j = 0 to n-1
                        if distance[i][j] > distance [i][k] + distance[k,j]
                                distance[i][j] = distance[i][k] + distance[k][j]
for i = 0 to n-1
        if distance[i][i] < 0
                return true
return false
```

Bellman-Ford takes O(VE) time. Here |V| = n and |E| = O(n^2). Therefore the total runtime is O(n^3) b)

```
Shortest path algorithm()
n = card(V)
distance = array[0,n][0,n]
nextNode = array[0,n-1][0,n-1]
for i = 0 to n-1
        for j = 0 to n-1
                if (i,j) in E
                        distance[i][j] = c(i,j)
                        nextNode[i][j] = j
                else
                        distance[i][j] =+ inf
                        nextNode[i][j] = null

for k = 0 to n-1
        for I = 0 to n-1
                for j = 0 to n-1
                        if distance[i][j] > distance [i][k] + distance[k,j]
                                distance[i][j] = distance[i][k] + distance[k][j]
                                nextNode[i][j] = nextNode[i][j]

finallist = []
for i = 0 to n-1
        if distance[i][j] < 0
                shortestpath = <i>
                runnode = i
                do
                        runnode = nextNode[runnode][i]
                        shortestpath.pushback(runnode)
                until runnode ==i
                        finalist.pushfront(shortestpath)
return finallist
```

The overall runtime is stil O(n^3) as we have 3 nested for loops.

29.1-5)

First we convert to standard form.

maximize

$2x_1 - 6x_3$

$x_1 + x_2 - x_3 \leq 7$

$-3x_1 + x_2 \leq -8$

$x_1 - 2x_2 - 2x_3 \leq 0$

$x_1, x_2, x_3 \geq 0$

Introducing slack variables x_4, x_5, and x_6.

Slack form:

$z = 2x_1 - 6x_3$
$x_4 = 7 - x_1 - x_2 + x_3$
$x_5 = -8 + 3x_1 - x_2$
$x_6 = -x_1 + 2x_2 + 2x_3$
Those are the basic variables. The nonbasic variables are x_1, x_2, and x_3.

29.3-5)
Convert to slack form:
$z = 18x_1 + 12.5x_2$
$x_3 = 20 - x_1 - x_2$
$x_4 = 12 - x_1$
$x_5 = 16 - x_2$

All coefficients are negative. Therefore the solution is (12,8,0,0,8) and has a value of 316.
With $x_1 = 12$ and $x_2 = 8$ and disregarding x_3 and x_5 we get a objective value of 316

29.3-6)
Convert to slack form:
$z = 5x_1 - 3x_2$
$x_3 = 1 - x_1 + x_2$
$x_4 = 1 - x_1 + x_2$
$x_4 = 2 - 2x_1 - x_2$

x_1 has a positive coefficient. Substitute x_1 for $x_1 = 1 - x_3 + x_2$
$z = 5 - 5x_3 + 2x_2$
$x_1 = 1 - x_3 + x_2$
$x_4 = 2x_3 - 2x_2$

x_2 has a positive coefficient. Substitute x_2 for $x_2 = x_3 - 0.5x_4$
$z = 5 - 3x_3 - x_4$
$x_1 = 1 - 0.5x_4$
$x_2 = x_3 - 0.5x_4$

All variables now have a negative coefficient. Solution is (1,0,0,0) with objective value 5.