MATH-566 **HW 4**

Due **Sep 21** before class. Just bring it before the class and it will be collected there.

1: (*Basic Feasible Solution Solver*)
Write a program that will solve an instance of linear programming by enumerating basic feasible solutions and keeping the best one. Input to the program is matrix $A \in \mathbb{R}^{m \times n}$ and vectors $\mathbf{b} \in \mathbb{R}^m$ and $\mathbf{c} \in \mathbb{R}^m$. Solve

$$(P) \left\{ \begin{array}{lrcl} \text{minimize} & \mathbf{c}^T\mathbf{x} & & \\ \text{subject to} & A\mathbf{x} & = & \mathbf{b} \\ & \mathbf{x} & \geq & 0 \end{array} \right.$$

A template for sage follows. I suggest you copy it from the LaTeX source file....

```
# https://wiki.sagemath.org/quickref?action=AttachFile&do=get&target=quickref-linalg.pdf
# I also suggest to google itertools.combinations
#
# This HW is about writing a program that solves linear programming problems where
# the input is A,b,c and you want to find x >= 0 that minimizes c^Tx if Ax = b.
#
# You should write a solver that enumerates all basic feasible solutions and picks
# the best one. Recall that you get a basic feasible solution by picking m columns
# of A that are linearly independent and solve for x.
#
# The same is also implemented using sage solver so you can check if your program
# produced the correct result.
#
import itertools

# This is where you will write your program
def solve_linear_program(A,b,c):
    m = len(b)
    n = len(c)
    if A.nrows() != m or A.ncols() != n or m > n:
        print "Invalid input"
        return [None,None]

    min_value = None
    min_x     = None

    # Here comes your code where you compute min_value and min_x

    return [min_value,min_x]
```

```
# This is solver using sage
def solve_using_sage(A,b,c):
    m = len(b)
    n = len(c)
    p = MixedIntegerLinearProgram(maximization=False);
    x = p.new_variable(nonnegative=True);
    for i in range(m):
        p.add_constraint( b[i] == sum([A[i,j]*x[j] for j in range(n)]) )
    p.set_objective(sum( [ c[j]*x[j] for j in range(n)] ))
    return p.solve()

# This is running your and also sage solver and prints the results
def test_solver(A,b,c):
    min_value, min_x = solve_linear_program(A,b,c)
    min_value_sage = solve_using_sage(A,b,c)
    print "Optimal solution has value", min_value_sage," You computed",min_value," for x



######### Several test data

# Input
A = matrix(QQ,[[1,0,1,0,0],[1,1,0,1,0],[-1,1,0,0,1]])
b = vector(QQ,[4,5,1])
c = vector(QQ,[-1,-2,0,0,0])

test_solver(A,b,c)


# Input
A = matrix(QQ,[[1,-2,1,0,0],[1,1,0,1,0],[-1,1,0,0,1]])
b = vector(QQ,[4,2,4])
c = vector(QQ,[-1,2,0,0,0])

test_solver(A,b,c)


# Input
A = matrix(QQ,[[1,-2,6,6,1,1,0,0],[1,1,2,2,-2,0,1,0],[-1,1,3,3,6,0,0,1]])
b = vector(QQ,[4,2,5])
c = vector(QQ,[-1,2,-2,-2,1,0,0,0])

test_solver(A,b,c)
```

Solution:

2: (*Using simplex method*)
Convert the following program to equational form (add x_3, x_4, x_5) and solve it using the simplex method.

$$(P)\left\{\begin{array}{lrcrcl} \text{maximize} & x_1 & + & 2x_2 & & \\ \text{subject to} & x_1 & & & \leq & 4 \\ & x_1 & + & x_2 & \leq & 5 \\ & -x_1 & + & x_2 & \leq & 1 \\ & & & x_1, x_2 \geq 0 & & \end{array}\right.$$

Use Bland's rule for selecting pivots. That is, pivot on variable with lowest possible index.

Use the last tableau to argue that the solution is indeed optimal.

Plot the set of feasible solutions of (P) and mark the solutions obtained after each iteration of the simplex method.

Solution:
New program:

$$(P')\left\{\begin{array}{lrcrcccccl} \text{maximize} & x_1 & + & 2x_2 & & & & & & \\ \text{subject to} & x_1 & & & + & x_3 & & & = & 4 \\ & x_1 & + & x_2 & & & + \; x_4 & & = & 5 \\ & -x_1 & + & x_2 & & & & + \; x_5 & = & 1 \\ & & & & & & & x_1, x_2, x_3, x_4, x_5 \geq 0 & & \end{array}\right.$$

It corresponds to the first simplex tableau

$$\begin{array}{rcrcrcr} x_3 & = & 4 & - & x_1 & & \\ x_4 & = & 5 & - & x_1 & - & x_2 \\ x_5 & = & 1 & + & x_1 & - & x_2 \\ \hline z & = & 0 & + & x_1 & + & 2x_2 \end{array}$$

We pivot on x_1 and replace x_3 in the base.

$$\begin{array}{rcrcrcr} x_1 & = & 4 & - & x_3 & & \\ x_4 & = & 1 & + & x_3 & - & x_2 \\ x_5 & = & 5 & - & x_3 & - & x_2 \\ \hline z & = & 4 & - & x_3 & + & 2x_2 \end{array}$$

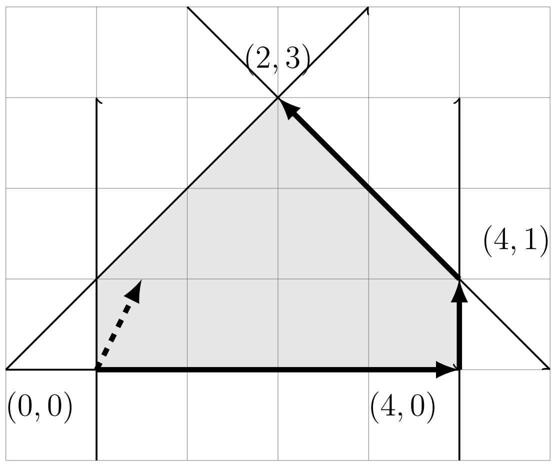
Now we pivot on x_2. The larges increase is bounded by x_4 and we obtain

$$\begin{array}{rcrcrcr} x_1 & = & 4 & - & x_3 & & \\ x_2 & = & 1 & + & x_3 & - & x_4 \\ x_5 & = & 4 & - & 2x_3 & - & x_4 \\ \hline z & = & 6 & & x_3 & - & 2x_4 \end{array}$$

Now we pivot on x_3. The larges increase is bounded by line with x_5 and we obtain

$$\begin{array}{rcrcrcr} x_1 & = & 2 & + & \frac{1}{2}x_5 & + & \frac{1}{2}x_4 \\ x_2 & = & 3 & - & \frac{1}{2}x_5 & - & \frac{3}{2}x_4 \\ x_3 & = & 2 & - & \frac{1}{2}x_5 & - & \frac{1}{2}x_4 \\ \hline z & = & 8 & - & \frac{1}{2}x_5 & - & \frac{5}{2}x_4 \end{array}$$

This gives optimal solution $x_1 = 2$ and $x_2 = 3$. The solution is optimal since any solution $\mathbf{x}$ must satisfy the last equation. If the solution contains x_4 or x_5 non-zero, then its value is less then ours. If $x_4 = x_5 = 0$, then the solution is determined uniquely.



3: (*Simplex method test*)
Use simplex method on the following program:

$$(P)\left\{\begin{array}{lrcrcl} \text{maximize} & x_1 & & & & \\ \text{subject to} & x_1 & - & x_2 & \leq & 1 \\ & -x_1 & + & x_2 & \leq & 2 \\ & & & x_1, x_2 \geq 0 & & \end{array}\right.$$

What is happening in the computation?

Solution:
The program is unbounded. It should be possible to discover this from the simplex tableau since it is possible to increase one variable beyond any bound.

Here are the simplex tableaus:

$$\begin{array}{rcrcrcr} x_3 & = & 1 & - & x_1 & + & x_2 \\ x_4 & = & 2 & + & x_1 & - & x_2 \\ \hline z & = & 0 & + & x_1 & & \end{array} \sim \begin{array}{rcrcrcr} x_1 & = & 1 & - & x_3 & + & x_2 \\ x_4 & = & 2 & - & x_3 & & \\ \hline z & = & 1 & - & x_3 & + & x_2 \end{array}$$

After one step of the simple method we see we can increase x_2 without any upped bound. This means that (P) is unbounded.

4: (*Ellipsoid method for solving linear programs*)
How would you solve a program $(P) =$ maximize $\mathbf{c}^T\mathbf{x}$ s.t. $A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ using the ellipsoid method with an $\varepsilon > 0$ error?
(Suppose both (P) and its dual (D) are superconsistent.)
(Hint: How to formulate the linear program as finding a point in a polytope? Use dual program and ε to guarantee full dimension.)

Solution:
Solve primal and dual at the same time, add equation $\mathbf{c}^T\mathbf{x} = \mathbf{b}^T\mathbf{y}$. Relax all constraints by ε to get a full dimension.
Suppose we create a polytope

$$\{(\mathbf{x}, \mathbf{y}) : A\mathbf{x} \leq \mathbf{b}, A^T\mathbf{y} \leq \mathbf{c}, \mathbf{c}^T\mathbf{x} = \mathbf{b}^T\mathbf{y}\}$$

Notice that by the duality theorem, every point in the polytope gives an optimal solution to P as well as to D since we added the strong duality constraint $\mathbf{c}^T\mathbf{x} = \mathbf{b}^T\mathbf{y}$. Unfortunately, the polytope does not have a full dimension. To get the full dimension, we notice that it is enougt to relax $\mathbf{c}^T\mathbf{x} = \mathbf{b}^T\mathbf{y}$, since (P) and (D) are already superconsistent. The relaxation may look like $\mathbf{c}^T\mathbf{x} + \epsilon \geq \mathbf{b}^T\mathbf{y}$.