Caleb Logemann
MATH 566 Discrete Optimization
Homework 3

1. Show that

$$A\mathbf{x} = \mathbf{b} \text{ has a nonnegative solution iff } \forall \mathbf{y} \in \mathbb{R}^m \text{ with } \mathbf{y}^T A \geq \mathbf{0}^T \text{ implies } \mathbf{y}^T \mathbf{b} \geq 0$$

implies

$$A\mathbf{x} \leq \mathbf{b} \text{ has a nonnegative solution iff } \forall \mathbf{y} \in \mathbb{R}^m, \mathbf{y} \geq \mathbf{0} \text{ with } \mathbf{y}^T A \geq \mathbf{0}^T \text{ implies } \mathbf{y}^T \mathbf{b} \geq 0.$$

2. Some university in Iowa was measuring the loudness of the fanâĂŹs screaming during the first touchdown of the local team. The measurements contain loudness in dB and the number of people at the stadium in thousands

# fans	53	55	59	61.5	61.5
dB	90	94	95	100	105

Find a line $y = ax + b$ best fitting the data. There are several different notions of best fitting. Commonly used is least squares that is minimizing $\sum_i (ax_i + b - y_i)^2$. But big outliers move the result a lot (and it is troublesome to do it using linear programming). Use the one that minimizes the sum of differences. That is

$$\sum_i (|ax_i + b - y_i|)$$

Write a linear program that solves the problem and solve if for the "measured" data.

The linear program's objective function should minimize the sum of the absolute values of the errors. However a linear program's objective function must be linear and therefore can't contain any absolute values funtions. Instead let $|ax_i + b - y_i| \leq e_i$. When $\sum_i e_i$ is minimized, then those inequalities become equalities. Furthermore, $|ax_i + b - y_i| \leq e_i$ can be transformed into two linear inequalities that can act as constraints for the linear program. Those two inequalties are

$$\begin{aligned} e_i &\geq ax_i + b - y_i \\ -e_i &\leq ax_i + b - y_i \end{aligned}$$

Rearranging this inequalites results in

$$\begin{aligned} -e_i + ax_i + b &\leq y_i \\ e_i + ax_i + b &\geq y_i \end{aligned}$$

Thus the linear program becomes

$$\begin{array}{ll} \min & \sum_i e_i \\ \text{s.t.} & -e_i + ax_i + b \leq y_i \quad \forall i \\ & e_i + ax_i + b \geq y_i \quad \forall i \end{array}$$

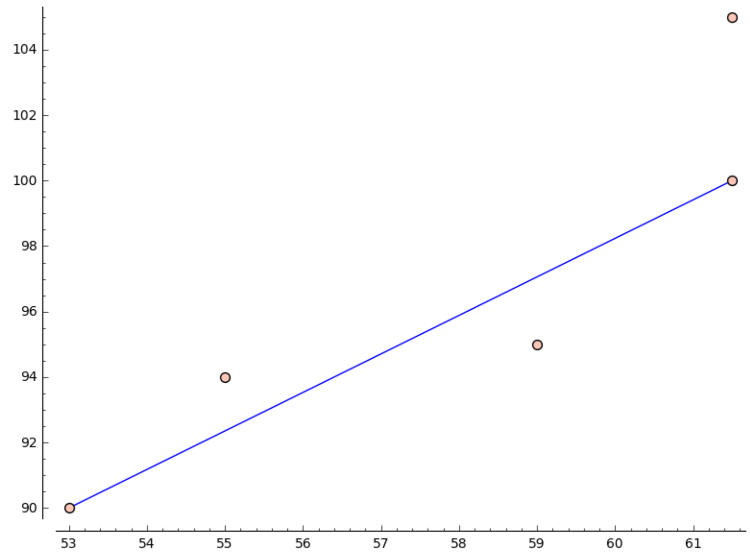
There are no restrictions on a or b they can be any real number. The values of e_i must be nonnegative, but we do not require a specific constraint for that as the other constraints implies this.

The following implements this linear program for the given data.

```
import numpy as np
# initialize data
x = vector([53, 55, 59, 61.5, 61.5])
y = vector([90, 94, 95, 100, 105])
n = x.length()

# create linear program
milp = MixedIntegerLinearProgram(maximization=False)
u = milp.new_variable(nonnegative=True)
a = milp.new_variable(nonnegative=False)
b = milp.new_variable(nonnegative=False)

# set objective and constraints
ones = matrix(np.full((1,n), 1))
milp.set_objective((ones*u)[0])
milp.add_constraint(identity_matrix(n)*u + b[0] + a[0]*x >= y)
milp.add_constraint(-identity_matrix(n)*u + b[0] + a[0]*x <= y)

print('Objective Value: {}'.format(milp.solve()))
a1 = milp.get_values(a[0])
b1 = milp.get_values(b[0])
print('a = {}'.format(a1))
print('b = {}'.format(b1))
print('Errors')
for i, v in milp.get_values(u).iteritems():
    print('u_%s = %s' % (i, v))

sp = scatter_plot(zip(x, y))
z = var('z')
lp = plot(a1*z + b1, (z, min(x), max(x)))
show(sp + lp)
```

The out put of the previous code is

```
Objective Value: 8.70588235294
a = 1.17647058824
b = 27.6470588235
Errors
u_0 = 0.0
u_1 = 1.64705882353
u_2 = 2.05882352941
u_3 = 0.0
u_4 = 5.0
```



3. Suppose you are preparing a schedule for classes. You have fixed number of classes and students. Every student told you which classes (s)he wants to attend. However, you do not have enough time slots to run all classes sequentially so you need to make some classes run in parallel. Create a schedule and argue why it is the best schedule in the sense that people as few conflicts as possible. You should be able to justify the optimality of the schedule in some sense. Make schedule with 3 and with 4 timeslots. Assume that there is no limit on how many classes can run in one timeslot.

   Use the following data

```
[[0,0,0,1,0,1,0,1,0,0,1,0,1,0,0,1,1,1,1,1,1,1,1,0,1,1,1,1,0,0,1,0,0,0,1,1,1,1,1,1,0,1,1,1,1,1,1,0,0,1,0,1,0,0],
[1,1,1,1,0,1,0,1,0,0,0,1,1,1,0,0,0,0,0,0,1,0,1,0,0,0,0,0,1,1,0,1,0,0,1,0,0,0,1,0,1,0,1,0,0,0,0,1,0,0,1,1,1,0],
[1,0,1,0,1,0,1,0,1,1,1,1,0,0,1,1,1,1,1,1,0,0,0,1,1,1,0,1,0,0,0,1,0,0,1,1,0,1,1,1,0,1,1,1,1,1,0,0,0,1,1,0,0,0],
[0,0,1,0,0,0,1,0,0,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,1,0,0,1,0,0,0,1,0,0,0,0,0,0,0,1,1,1,0,1,1,0,0,0,0],
[1,0,0,1,0,1,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,1,0,0,1,0,1,0,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0],
[0,1,1,1,1,0,0,0,1,1,1,1,1,0,1,0,1,1,0,1,0,0,0,0,0,1,0,0,0,1,1,0,0,1,1,0,1,1,1,1,0,0,0,1,1,1,1,0,0,1,0,1,0,1],
[0,0,1,1,0,1,1,0,0,1,1,1,0,1,1,0,1,1,0,1,0,1,1,0,0,1,1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1,1,0,1,1,0,0,1,0,1,1,0],
[1,0,0,0,0,1,1,0,1,0,0,0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,1,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0,1,0,1,0,1,0,1,0,0,0,0,0],
[0,1,0,0,0,1,1,0,1,0,1,0,0,0,0,1,1,1,0,1,0,1,1,1,1,1,0,0,0,0,1,0,1,0,0,0,1,0,1,1,0,0,1,1,1,0,1,0,0,0,0,1,0,1],
[1,1,1,1,0,1,0,1,0,0,0,1,1,1,0,0,0,0,0,0,1,0,1,0,0,0,0,0,1,1,0,1,0,0,1,0,0,0,1,0,1,0,1,0,0,0,0,1,0,0,1,1,1,0],
[1,0,1,0,1,0,1,0,1,1,1,1,0,0,1,1,1,1,1,1,0,0,0,0,1,1,0,1,0,0,0,1,0,0,1,1,0,1,1,1,0,1,1,1,1,1,0,0,0,1,1,0,0,0],
[0,0,1,0,0,0,1,0,0,1,1,0,0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,1,1,1,0,1,1,0,0,0,0],
[1,0,0,1,0,1,0,0,0,1,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0,1,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,1,0]]
```

   Row corresponds to one class, columns corresponds to one students. 1 means the student wants to attend the class.