MATH 416: ADVANCED APPLIED LINEAR ALGEBRA: FALL 2012
COMMENTS ON HW PROBLEMS

STEVEN J. MILLER (SJM1@WILLIAMS.EDU): MATH 416, FALL 2012

ABSTRACT. A key part of any math course is doing the homework. This ranges from reading the material in the book so that you can do the problems to thinking about the problem statement, how you might go about solving it, and why some approaches work and others don't. Another important part, which is often forgotten, is how the problem fits into math. Is this a cookbook problem with made up numbers and functions to test whether or not you've mastered the basic material, or does it have important applications throughout math and industry? Below I'll try and provide some comments to place the problems and their solutions in context.

1. HW #2: DUE SEPTEMBER 17, 2012

1.1. **Assignment.** First assignment: Section 2.2.3 of my notes: Exercises 2.3, 2.4, 2.5. Final problem: the diet problem with two products and two constraints led us to an infinite region, and then searching for the cheapest diet led us to a vertex point. Modify the diet problem by adding additional constraints so that, in general, we have a region of finite volume, and again show that the optimal point is at a vertex. Your constraints should be reasonable, and you should justify their inclusion.

1.2. **Solutions.** First assignment:

#1: Exercise 2.3: Find the optimal solution to the diet problem when the cost function is $\text{Cost}(x_1, x_2) = x_1 + x_2$.

Solution: Unfortunately I made a mistake in describing the Diet Problem in the notes. In the text I had one unit of cereal contributing 30 units of iron and 5 units of protein; however, when I wrote the equations in (1) I transposed things, and had one unit of cereal giving 30 units of iron and 15 units of protein. I'll thus solve the problem both ways.

Using the numbers in the book, we have the following system of equations:

$$\begin{aligned} 30x_1 + 15x_2 &\geq 60 \text{ (iron)} \\ 5x_1 + 10x_2 &\geq 70 \text{ (protein)} \\ x_1, x_2 &\geq 0, \end{aligned} \tag{1.1}$$

and now we want to minimize $\text{Cost}(x_1, x_2) = x_1 + x_2$. It isn't immediately clear what the optimal solution is, as both products have the same cost per unit, but one delivers more iron and the other more protein. We give a plot in Figure 1.

Here is the Mathematica code to generate the plot.

```
line1[x_] := If[-2 x + 4 > 0, -2 x + 4, 0]
Cost[x_, c_] := If[ -x + c > 0, -x + c, 0]
Plot[{line1[x], -.5 x + 7, Cost[x,10.0], Cost[x,10.1],
  Cost[x,10.2], Cost[x,10.3], Cost[x,10.4], Cost[x,10.5]}, {x,0,14}]
```

The cost falls as we shift the cost lines down and to the left. Notice that whenever the protein constraint is satisfied then the iron constraint holds as well, and is thus extraneous. (To see this, note the coefficients from this equation are all larger than those of the one below, and the required amount is less!) The optimal diet will be entirely steak (i.e., only the second product). Thus $x_1 = 0$ and $x_2 = 7$.

Date: November 5, 2012.

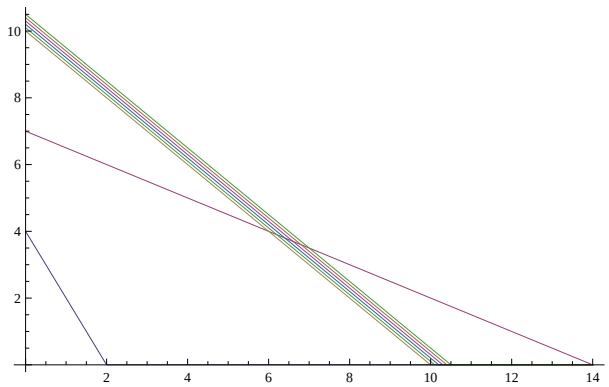

FIGURE 1. Diet Problem 1: Plot of the first diet problem, with several cost lines.

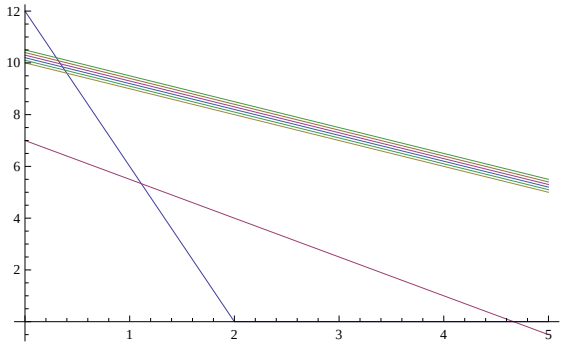

FIGURE 2. Diet Problem 1: Plot of the second diet problem, with several cost lines.

We now consider the other diet problem:

$$\begin{aligned} 30x_1 + 5x_2 &\geq 60 \ \text{(iron)} \\ 15x_1 + 10x_2 &\geq 70 \ \text{(protein)} \\ x_1, x_2 &\geq 0, \end{aligned} \tag{1.2}$$

and now we want to minimize $\text{Cost}(x_1, x_2) = x_1 + x_2$. We give a plot in Figure 2.

The Mathematica code is

```
line2[x_] := If[-6 x + 12 > 0, -6 x + 12, 0]
Cost[x_, c_] := If[ -x + c > 0, -x + c, 0]
Plot[{line2[x], -1.5 x + 7, Cost[x, 10.0], Cost[x, 10.1],
  Cost[x,10.2], Cost[x,10.3], Cost[x,10.4], Cost[x,10.5]}, {x,0,5}]
```

The cost is falling as the cost line moves down and to the left. We flow until we have none of the second product, only buying the first product (thus $x_1 = 4\frac{2}{3}$ and $x_2 = 0$).

#2: Exercise 2.4: There are three vertices on the boundary of the polygon (of feasible solutions); we have seen two choices of cost functions that lead to two of the three points being optimal solutions; find a linear cost function which has the third vertex as an optimal solution.

Solution: Based on the wording, we want the matrix formulation from the book (not the equations in the paragraphs in the text, but equation (1)):

$$\begin{aligned} 30x_1 + 5x_2 &\geq 60 \text{ (iron)} \\ 15x_1 + 10x_2 &\geq 70 \text{ (protein)} \\ x_1, x_2 &\geq 0. \end{aligned} \tag{1.3}$$

The two lines have slope -6 and -1.5; if we choose our cost function to have a slope between these two values, then the intersection of those two lines will be the unique optimal point. We can do this if we take a slope of -4, or equivalently if the cost function is $\text{Cost}(x_1, x_2) = 4x_1 + x_2$ (though we may replace the 4 with any number strictly between 1.5 and 6).

#3: Exercise 2.5: Generalize the diet problem to the case when there are three or four types of food, and each food contains one of three items a person needs daily to live (for example, calcium, iron, and protein). The region of feasible solutions will now be a subset of $\mathbb{R}^3$. Show that an optimal solution is again a point on the boundary.

Solution: If each food can contain exactly one item, then the only way we can have a solution is if each food contains a different item *or* we have more food choices than needed items. If we only have three food items, each food must contain a different nutrient, and then there is only one feasible diet: take the appropriate amount of each food. If instead we have four types of food, we need two of the food types to have the same nutrient, and the other two foods to have the remaining two nutrients. In this case, the only interesting aspect of the problem concerns the nutrient represented by two different foods. We simply take whichever food has a better price per unit of nutrient.

The problem is more interesting if the foods can contain all three items. In this case, if we have x_j units of food j, and food j delivers a_{ij} units of nutrient i then, assuming we need r_i units of nutrient i to stay alive, our constraints are

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &\geq r_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &\geq r_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &\geq r_3 \\ x_1, x_2, x_3 &\geq 0. \end{aligned} \tag{1.4}$$

The cost function is $\text{Cost}(x_1, x_2, x_3) = c_1x_1 + c_2x_2 + c_3x_3$.

The same logic as before shows that an optimal solution must be on a boundary; the difference is now we need to use words like planes rather than lines. Instead of a region in the upper right quadrant we get a region in the positive octant. We now have planes of constant cost; we can decrease the cost by moving towards the origin, and thus if we're at an interior point we can lower the cost by shifting 'down'. Similarly, once we hit the boundary, we can continue to lower the cost by moving to a vertex (we might not be lowering the cost if the slopes align, but in that case we at least keep the cost constant).

It's a bit harder of course to visualize things in three-dimensions. We give a plot in Figure 3; the constraints are

$$\begin{aligned} 2x + y + z &\geq 4 \\ .6x + 2y + z &\geq 4 \\ 3x + 4y + z &\geq 6 \\ x, y, z &\geq 0. \end{aligned}$$

The Mathematica code is

```
plane1[x_, y_] := If[-2 x - y + 4 >= 0, -2 x - y + 4, 0];
plane2[x_, y_] := If[-.6 x - 2 y + 4 >= 0, -.6 x - 2 y + 4, 0];
plane3[x_, y_] := If[-3 x - 4 y + 6 >= 0, -3 x - 4 y + 6, 0];
Plot3D[{plane1[x,y], plane2[x,y], plane3[x,y]}, {x,0,2}, {y,0,2}]
```

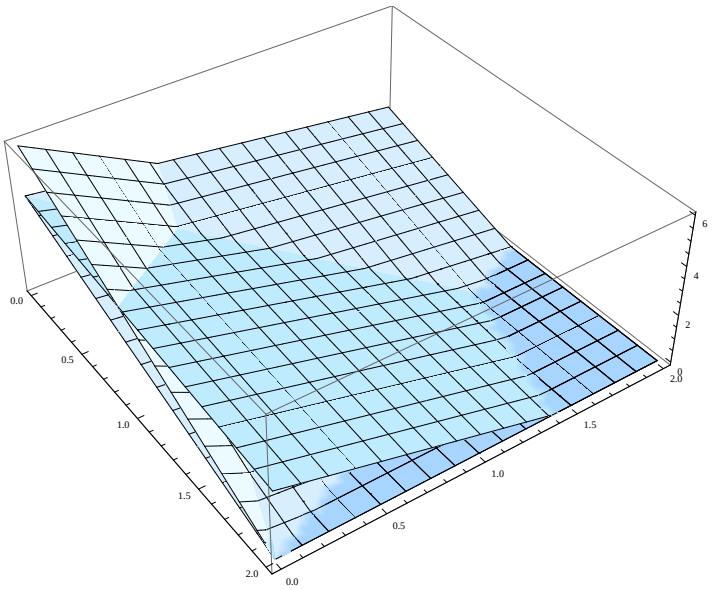

FIGURE 3. Diet Problem 3D: Plot of constraints in a 3-dimensional diet problem.

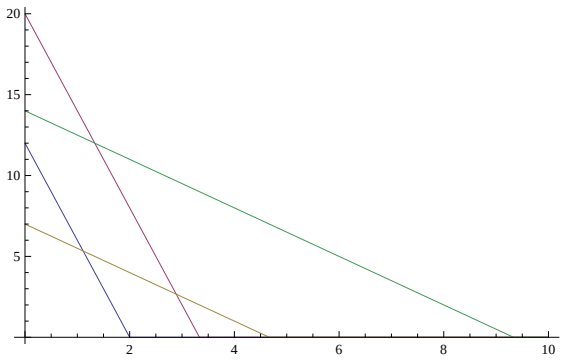

FIGURE 4. Diet Problem 3: Plot of the third diet problem, now with maximum daily allowances.

#4: The diet problem with two products and two constraints led us to an infinite region, and then searching for the cheapest diet led us to a vertex point. Modify the diet problem by adding additional constraints so that, in general, we have a region of finite volume, and again show that the optimal point is at a vertex. Your constraints should be reasonable, and you should justify their inclusion.

Solution: There are lots of ways to keep things finite. A 'fun' way is to prohibit you from eating too much of any nutrient (in other words, too much of a good thing *can* kill you!). Right now we said we need at least 60 units of iron and at least 70 units of protein; maybe we die if we eat more than 100 units of iron or 140 units of protein. We give a plot in Figure 4.

The Mathematica code is

```
line3[x_, c_] := If[-6 x + c/5 > 0, -6 x + c/5, 0]
line4[x_, c_] := If[-1.5 x + c/10 > 0, -1.5 x + c/10, 0]
Plot[{line3[x, 60], line3[x, 100], line4[x, 70], line4[x, 140]},
{x, 0, 10}]
```

There is a very nice consequence to our restrictions. We now have a closed and bounded subset of the plane. We know from real analysis that any continuous function on a closed and bounded set attains its maximum and its minimum. Thus, there *is* an optimal diet (i.e., a cheapest diet that will keep you alive).

The problem is we don't necessarily know how to find it. When we start studying the simplex method, we'll learn how to flow from a guess to a better guess. This is similar to some items you may have seen. For example, in Lagrange Multipliers we know candidates for a local extremum of f to the region with constraint function g satisfy $\nabla f = \lambda \nabla g$; if the two gradients are not aligned, we obtain information on which direction to flow. Of course, what's best locally might not be best globally – it might be better to take a small hit in the beginning to get to the global extremum; sadly this issue causes enormous complications in the subject. Another situation where you might have seen this is in contraction mappings, which give an iterative procedure to find fixed points (a nice application of this is in differential equations).

2. HW #3: DUE SEPTEMBER 24, 2012

2.1. **Assignment.** Section 2.3.1 of my notes: Exercise 2.7 (The notes might not have been clear: take as the original problem $A^T x \leq b$, x arbitrary, minimize $c^T x$, and take the dual problem to be $y^T A \geq c^T$, y arbitrary, minimize $y^T(-b)$). Problem #2: Formulate Sudoku as a linear programming problem (you can do either 4x4 or 9x9 Sudoku). Problem #3: Medical Residencies: Imagine there are P people who have just graduated from medical school and H hospitals. We are trying to match medical students with hospitals. Each student ranks the hospitals and each hospital ranks the students. Formulate this assignment problem as a linear programming problem; you may need to make some assumptions to finish the modeling. There are a lot of ways to do this; what do you want to maximize? Does a feasible solution always exist, and if so when? Does the existence of a feasible solution depend on the function you want to optimize? Problem #4: Exercise 2.10 from the notes. Note this is the A' from the text, and thus the k columns of A' are linearly independent. #5: Exercise 2.11 from the notes.

2.2. **Solutions.** #1: Exercise 2.7 (note there is an omission in the notes; the dual problem should ask you to minimize or maximize a given quantity; part of the homework assignment is to figure out exactly what should be minimized or maximized, and if we want a maximum or a minimum).

Solution: We'll consider the canonical problem $A\vec{x} \leq \vec{b}$, with $\vec{x}$ consisting of real numbers and with objective function $\vec{c}^T\vec{x}$ to minimize. The dual problem is $\vec{y}^T A \geq -\vec{c}^T$, and we wish to maximize the function $\vec{y}^T\vec{b}$. We may rewrite this as $-A^T\vec{y} \leq -\vec{c}$ with objective function $\vec{y}^T(-\vec{b})$ to minimize. Thus our original problem has matrix A, constraint vector $\vec{b}$ and objective vector $\vec{c}$, while the dual problem has matrix $-A^T$, constraint vector $-\vec{c}$ and objective vector $-\vec{b}$ (and both are minimization problems).

Thus taking the dual replaces the matrix with its negative transpose, and interchanges the constraint and objective vectors (we still have a minimization problem, but in interchanging we must add a minus sign). We thus have the map

$$\text{Dual}(A, \vec{b}, \vec{c}) \;=\; (-A^T, -\vec{c}, -\vec{b}).$$

If we apply this map again, we find

$$\text{Dual}(A, -\vec{c}, -\vec{b}) \;=\; \left((A^T)^T, \vec{b}, \vec{c}\right).$$

Since the transpose of the transpose of A is A, we have returned to our initial problem.

#2: Formulate Sudoku as a linear programming problem (you can do either 4x4 or 9x9 Sudoku).

Solution: Let x_{ijd} be the binary variable which is 1 if the cell in row i and column j is d, and zero otherwise. Let n be either 4 or 9. Then the constraints are

- For all $j \in \{1, \dots, n\}$ and for all $d \in \{1, \dots, n\}$: $\sum_{i=1}^n x_{ijd} = 1$. This means each column has each digit exactly once.
- For all $i \in \{1, \dots, n\}$ and for all $d \in \{1, \dots, n\}$: $\sum_{j=1}^n x_{ijd} = 1$. This means each row has each digit exactly once.
- Let $\mathcal{F} = \{(1,1), (1,2), \dots, (\sqrt{n}, \sqrt{n})\}$, and let $(a,b) + \mathcal{F}$ be the set of all pairs of the form $(a+x, b+y)$ for some $(x,y) \in \mathcal{F}$. Then For all $a, b \in \{0, 1, \dots, \sqrt{n}-1\}$ and all $d \in \{1, \dots, n\}$ we have $\sum_{(i,j)\in(a,b)+\mathcal{F}} x_{ijd} = 1$. This means that in each $\sqrt{n} \times \sqrt{n}$ box we have each digit.

We need an objective function. As all we care is for a feasible solution, we can take as our objective function $\sum_i \sum_j \sum_d x_{ijd}$.

Finally, often Sudokus have certain cells given to us; in that case, we simply add these as constraints: if $\mathcal{S}$ is the set of indices where we are given values, and v_{ij} is the given value, then for all $(i,j) \in \mathcal{S}$ we have $x_{ijd} = 1$ if $d - v_{ij}$ and 0 otherwise.

There are other ways to try and solve this. We could instead let $x_{ij} \in \{1,2,3,4\}$ and try to make that work. I know one group tried the constraint that each column, each row and each of the four blocks of four had to sum to 10, trying to use the only way to get 10 from these numbers is $1+2+3+4$.

Unfortunately, $2+3+2+3$ also works, but leads to an invalid Sudoku:

$$\begin{pmatrix} 2 & 3 & 2 & 3 \\ 3 & 2 & 3 & 2 \\ 2 & 3 & 2 & 3 \\ 3 & 2 & 3 & 2 \end{pmatrix}.$$

#3: Medical Residencies: Imagine there are P people who have just graduated from medical school and H hospitals. We are trying to match medical students with hospitals. Each student ranks the hospitals and each hospital ranks the students. Formulate this assignment problem as a linear programming problem; you may need to make some assumptions to finish the modeling. There are a lot of ways to do this; what do you want to maximize? Does a feasible solution always exist, and if so when? Does the existence of a feasible solution depend on the function you want to optimize?
Solution: First, the existence of a feasible solution is independent on whether or not an optimal solution exists. Let x_{ph} equal 1 if we assign student p to hospital h, and 0 otherwise. What are the constraints?

- No student can be assigned to more than one hospital: for all $p \in \{1,\dots,P\}$ we have $\sum_{h=1}^{H} x_{ph} \leq 1$. We write less than or equal to and not equal to as perhaps some students *will not be assigned to hospitals!*.
- Perhaps each hospital has a certain number of students needed, say d_i. Then for all $h \in \{1,\dots,H\}$ we have $\sum_{p=1}^{P} x_{ph} \geq d_i$. We might want equality here (no need to hire people you don't need, unless you want to keep them in the labor pool and have them gain experience for later).

The difficulty is in choosing an objective function. What do we want to minimize? A simple possibility is to have each student rank the H hospitals and each hospital rank each student, giving a 1 for first choice, 2 for second and so on. We then want to minimize the total score. Letting r_{ph} be the rank person p attaches to working at hospital H, and ρ_{ph} the rank hospital h attaches to having person p, we need to minimize $\sum_p \sum_h (r_{ph} + \rho_{ph}) x_{ph}$.

There are other rankings we can use. Perhaps each person gets 100 points and must assign them among the H hospitals. Or perhaps each person writes down how happy they would be working at each hospital, with 100 high and 0 low. There are lots of tweaks like this that we can do that will keep the objective function linear. Note something similar to this *is* used in assigning doctors to residency programs.

#4: Exercise 2.10 from the notes. Prove that if A' has M rows and k columns, with $M \geq k$, then $A'^T A'$ is invertible. Note this is the A' from the text, and thus the k columns of A' are linearly independent.
Solution: If z is any vector with k components, then $z^T A'^T A' z = ||A'z||^2$, where $||v||$ denotes the length of a vector v. Imagine $A'^T A'$ is not invertible. Then the columns of this matrix are dependent, and there is some non-zero vector v such that $A'^T A' v$ is the zero vector. Thus $v^T A'^T A' v = 0$, or $||A'v||^2 = 0$. The only way the length of the vector $A'v$ can be zero is if $A'v$ is zero. What does it mean for $A'v$ to be zero? If v is not the zero vector, it means the columns of A' are linearly dependent. As we know these columns are linearly dependent, we must have v the zero vector. This contradicts our assumption that v is not the zero vector, completing the proof.

#5: Exercise 2.11 from the notes. For fixed M, find some lower bounds for the size of $\sum_{k=1}^{M} \binom{N}{k}$. If $M = N = 1000$ (which can easily happen for real world problems), how many basic feasible solutions could there be? There are less than 10^{90} sub-atomic objects in the universal (quarks, photons, et cetera). Assume each such object is a supercomputer capable of checking 10^{20} basic solutions a second (this is much faster than current technology!). How many years would be required to check all the basic solutions?

Solution: The binomial coefficients are increasing to the middle, then decreasing. If $M \leq N/2$ a decent bound for the sum is $\binom{N}{M}$; if $N/2 \leq M \leq N$ a reasonable bound is $\binom{N}{N/2}$, though even better would be $\frac{1}{2}(1+1)^N$.

A basic feasible solution is a feasible solution where the columns corresponding to the non-zero entries are linearly independent. If we let c be the number of such columns, we find $1 \leq c \leq 1000$, and for each c the largest number of basic feasible solutions would be $\binom{1000}{c}$. We thus have $\sum_{c=1}^{1000}\binom{1000}{c}$. By the Binomial Theorem, this is $2^{1000}-1$ (we subtract 1 as we don't have $c=0$), which is approximately $1.07151 \cdot 10^{301}$. Under our assumptions, we can check 10^{110} possibilities a second, which means we need about $1.07151 \cdot 10^{191}$ seconds. As there are about $1.32016 \cdot 10^8$ seconds in a year, we would need approximately $8.11651 \cdot 10^{182}$ years, far longer than the 15 billion or so years we believe the universe has existed.

Preview: Week 4: Sept 24 to Sept 28, 2012:
HW: Due Monday, October 1: #1: Choose a tentative topic for your paper and class presentation, and a tentative group (of between 2 and 4 people). Have one person from the group email me a few paragraphs listing all group members, describing the project and stating what you want the class to get out of your write-up and talk. If you are having trouble coming up with topics, let me know. #2: Consider the 3×3 constraint matrix A where the first row is 1, 2, 3, the second row is 4, 5, 6 and the third row 7, 8, 9 (thus it's the numbers 1 through 3^2). Let the vector b equal $(1,1,1)^T$. Find all basic feasible solutions to $Ax = b$ with $x \geq 0$. #3: Prove $Mz = w$ has either 0, 1 or infinitely many solutions, and no other options can happen. #4: Let's revisit the chess problem from class. Consider an $n \times n$ chess board. We want to put down n queens and maximize the number of pawns that can be safely placed on the board. Set this up as a linear programming problem. #5: Do Exercise 2.14 from my notes.

Extra credit: Modify #2 so that we have an $n \times n$ matrix with the entries going from 1 to n^2, with $n \geq 3$. Let $b = (1,1,\ldots,1)^T$. Find all basic feasible solutions.

3. HW #4: DUE OCTOBER 1, 2012

3.1. **Assignment:** HW: Due Monday, October 1: #1: Choose a tentative topic for your paper and class presentation, and a tentative group (of between 2 and 4 people). Have one person from the group email me a few paragraphs listing all group members, describing the project and stating what you want the class to get out of your write-up and talk. If you are having trouble coming up with topics, let me know. #2: Consider the 3×3 constraint matrix A where the first row is 1, 2, 3, the second row is 4, 5, 6 and the third row 7, 8, 9 (thus it's the numbers 1 through 3^2). Let the vector b equal $(1, 1, 1)^T$. Find all basic feasible solutions to $Ax = b$ with $x \geq 0$. #3: Prove $Mz = w$ has either 0, 1 or infinitely many solutions, and no other options can happen. #4: Let's revisit the chess problem from class. Consider an $n \times n$ chess board. We want to put down n queens and maximize the number of pawns that can be safely placed on the board. Set this up as a linear programming problem. #5: Do Exercise 2.14 from my notes.

Extra credit: Modify #2 so that we have an $n \times n$ matrix with the entries going from 1 to n^2, with $n \geq 3$. Let $b = (1, 1, \ldots, 1)^T$. Find all basic feasible solutions.

3.2. **Solutions:** #1: Choose a tentative topic for your paper and class presentation, and a tentative group (of between 2 and 4 people). Have one person from the group email me a few paragraphs listing all group members, describing the project and stating what you want the class to get out of your write-up and talk. If you are having trouble coming up with topics, let me know.

#2: Consider the 3×3 constraint matrix A where the first row is 1, 2, 3, the second row is 4, 5, 6 and the third row 7, 8, 9 (thus it's the numbers 1 through 3^2). Let the vector b equal $(1, 1, 1)^T$. Find all basic feasible solutions to $Ax = b$ with $x \geq 0$.

Solution: We give a one-line solution at the end; as a large part of homework is to learn the methods and techniques, it is good to see the straightforward approach.

The matrix A is not invertible (the $n \times n$ matrix with entries going from 1 to n^2 is invertible only when $n \leq 2$); one way to see this is to note that the first plus third columns are twice the second. Note that any pair of columns are linearly independent, and any column is linearly independent. Thus there are 6 sub-matrices that generate basic feasible solutions, and each generates a unique candidate for a basic feasible solution. If A' is the reduced matrix, then the candidate for the basic feasible solution is found by solving $A'x' = b$. We multiply by A'^T on the left since $A'^T A'$ is invertible. This gives $A'^T A'x' = A'^T b$, or $x' = (A'^T A')^{-1} A^T b$. This gives us the non-zero entries of the candidate for the basic feasible solution; we finish by adding the zero entries.

- Using the first column, $(1, 4, 7)$, we get a non-zero element of $2/11$ and thus the candidate for the basic feasible solution is $(2/11, 0, 0)$.
- Using the second column, $(2, 5, 8)$, we get a non-zero element of $5/31$ and thus the candidate for the basic feasible solution is $(0, 5/31, 0)$.
- Using the third column, $(3, 6, 9)$, we get a non-zero element of $1/7$ and thus the candidate for the basic feasible solution is $(0, 0, 1/7)$.
- Using the first two columns we get non-zero elements $(-1, 1)$, and thus the candidate for the basic feasible solution is $(-1, 1, 0)$.
- Using the first and third columns we get non-zero elements $(-1/2, 1/2)$, and thus the candidate for the basic feasible solution is $(-1/2, 0, 1/2)$.
- Using the second and third columns we get non-zero elements $(-1, 1)$, and thus the candidate for the basic feasible solution is $(0, -1, 1)$.

Note we can check to make sure these are feasible solutions. When we check, however, the first three all ***fail*** to satisfy $Ax = b$, though the last three do. What went wrong? The problem is that b is not a linear combination of fewer than 2 columns of A, and when we try to take just one column it breaks down. This shouldn't be surprising. In that case $A'^T A$ is a 1×1 matrix and b is not in the column space of A'. While the last three solve the constraints, they are not basic feasible solutions as each has a negative entry. Thus, there are ***no*** basic feasible solutions.

One can do these calculations in a system such as Mathematica, though you have to be careful with the syntax. Here's the code for it.

```
A = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
Transpose[A];
b = Transpose[{{1, 1, 1}}];
A.b
Ap = Transpose[{{1, 4, 7}, {2, 5, 8}}];
Transpose[Ap] .Ap
Inverse[Transpose[Ap] .Ap]
Inverse[Transpose[Ap]. Ap]. (Transpose[Ap] . b)
```

Now, for the promised one-line solution. Imagine there is a basic feasible solution. Then we have $Ax = b$ with the entries of x non-negative and each entry of b is 1. Notice that the second row of A dominates the first row (each matrix element in the second row is larger than the corresponding entry in the first row), yet the constraints want the resulting dot products to be equal. In other words, $x_1 + 2x_2 + 3x_3 = 1$ and $4x_1 + 5x_2 + 6x_3 = 1$. This is impossible, as the second constraint can be written as

$$(x_1 + 2x_2 + 3x_3) + 3(x_1 + x_2 + x_3) \ = \ 1;$$

as $x_1 + 2x_2 + 3x_3 = 1$ this implies $3(x_1 + x_2 + x_3) = 0$, which implies each $x_i = 0$ (as they must be non-negative for a feasible solution), clearly violating the weighted sum equalling 1.

#3: Prove $Mz = w$ has either 0, 1 or infinitely many solutions, and no other options can happen.
Solution: If $Mz = w$ has exactly 0 or 1 solution we're done. Assume that it has at least two solutions, say $Mz_1 = w$ and $Mz_2 = w$ with $z_1 \neq z_2$; we have to show there are infinitely many solutions. Notice that $M(z_1 - z_2) = 0$; thus $z_1 - z_2$ is a non-zero vector in the nullspace of M. If we let $z_\lambda = z_1 + \lambda(z_1 - z_2)$ then we see $Mz_\lambda = w$, and thus there are infinitely many solutions.

#4: Let's revisit the chess problem from class. Consider an $n \times n$ chess board. We want to put down n queens and maximize the number of pawns that can be safely placed on the board. Set this up as a linear programming problem.
Solution: Let $x_{ij} = 1$ if we have a queen on the board in row i and column j, and zero otherwise. Our first constraint is

$$\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} \ = \ n.$$

This constraint says we place exactly n queens on the board. In fact, this is the only 'real' constraint; the other constraints come from helping to write the objective function.

For each point (i, j) on the chessboard, let $\mathcal{A}_{i,j}$ denote the squares that a queen placed at (i, j) can attack (plus the square (i, j)). For example, if $(i, j) = (1, 1)$ then $\mathcal{A}_{1,1}$ is the first row, the first column, and the diagonal of all pairs (d, d). We're going to introduce some new binary variables y_{ij}. We should think of these as being 1 if we can place a pawn safely at (i, j) and zero otherwise. Consider the constraints for all pairs (i, j) ***such that there is a queen at*** (i, j) we have

$$\sum_{(i,j) \in \mathcal{A}_{ij}} y_{ij} \ = \ 0.$$

This means we cannot place a pawn in the kill zone caused by a queen at (i, j); the difficulty, though, is we don't know where the queens are. One solution is to multiply this constraint by x_{ij} on the left, so it only comes into play if there is a queen at (i, j). In other words, consider

$$x_{ij} \sum_{(i,j) \in \mathcal{A}_{ij}} y_{ij} \ = \ 0;$$

	X	X	X
X	X	**X**	X
	X	X	X
X		X	

TABLE 1. The 12 squares that attack (2,3) on a 4×4 board.

if $x_{ij} = 0$ (so no queen at (i, j)) then the y_{ij}'s are free; if there is a queen there then each $y_{ij} = 0$ (i.e., cannot place a pawn there). Unfortunately, this is not linear. If it were, we'd be done, and we'd try to maximize the sum of the y_{ij}'s, as that would give us the most pawns placeable; technically, we need a minimization problem, so we minimize

$$-\sum_{i,j=1}^{n} y_{ij}.$$

This would give us a ***quadratic*** programming problem; the constraints are quadratic in places, although the objective function is still linear. It is possible to do this problem, however, with linear constraints.

Let $\mathcal{Q}_{ij}$ be the set of all pairs on the $n \times n$ chessboard that can attack square (i, j) **and** the square (i, j) as well. We're using a script Q to emphasize that these are the places to put a queen to eliminate the possibility of a pawn being safely placed at (i, j). For example, if $n = 4$ then

$$\mathcal{Q}_{2,3} \ = \ \{(2,1),(2,2),(2,3),(2,4),(1,3),(3,3),(4,3),(1,2),(3,4),(1,4),(3,2),(4,1)\}$$

(see Table 1 for a visualization).

Our objective function is the same as before:

$$-\sum_{i,j=1}^{n} y_{ij}.$$

We want this to be as small as possible, which means we want the sum of the y_{ij}'s to be as large as possible. In other words, we want to have as many squares as possible not under attack by queens.

As we are placing n queens on the board, at most n queens can make the square (i, j) unsafe for a pawn. Consider the constraint: for all $(i, j) \in \{1, \ldots, n\}^2$ we have

$$2n(1 - y_{ij}) \ \geq \sum_{(i',j') \in \mathcal{Q}_{ij}} x_{i'j'}.$$

What does this do?

- If there are no queens on the board attacking the square (i, j) then the right hand side is zero and there is no effect on y_{ij}, as the left hand side is always non-negative. We thus have complete freedom in choosing y_{ij} in this case. As we are trying to minimize the negative of the sum of the y_{ij}'s (or, equivalently, maximize the sum of the y_{ij}'s), we the program will take $y_{ij} = 1$ and place a pawn safely there.
- What if there is at least one queen attacking the square (i, j)? Then the sum on the right hand side is positive. Further, **it is at most n as there are only n queens**. If $y_{ij} = 1$ then the left hand side is 0, which is smaller than n and contradicts the inequality! Thus we cannot take $y_{ij} = 1$, and this case forces y_{ij} to be zero. This is exactly what we want, as it now tells us we cannot have a pawn safely placed at (i, j).

As we took a long path to the answer, it's worth writing down the constraints cleanly:

- Parameters: $\mathcal{Q}_{ij}$: all the pairs (i, j) on an $n \times n$ chessboard that can attacked a pawn located at (i, j), including (i, j); equivalently, these are all the squares where a queen placed there would attack a pawn at (i, j).

- Variables: $x_{ij} = 1$ if a queen is at (i,j) and 0 otherwise; $y_{ij} \in \{0,1\}$ (constraints chosen later will force y_{ij} to be 0 if the location of the queens prevents a pawn from being placed safely at (i,j)).
- Constraint: Location of Queens: $\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} = n$. This forces exactly n queens to be placed on the $n \times n$ board.
- Constraint: Location of Pawns: $2n(1-y_{ij}) \geq \sum_{(i',j')\in\mathcal{Q}_{ij}} x_{i'j'}$. We may rewrite this in more standard form as
$$2ny_{ij} + \sum_{(i',j')\in\mathcal{Q}_{ij}} x_{i'j'} \;\leq\; 2n.$$
If a queen is placed and attacks (i,j) then y_{ij} must be zero (as otherwise the left hand side exceeds the right hand side). If no queen is placed that attacks square (i,j) then y_{ij} is free.
- Objective function: Minimize $-\sum_{i=1}^{n}\sum_{j=1}^{n} y_{ij}$. This is the negative of the number of pawns that may safely be placed on the board.

Note that our choice of objective function will make us set y_{ij} to 1 whenever possible. If we wanted to truly make y_{ij} indicate whether or not a pawn ***is*** safely placed at (i,j), all we need to do is ***force*** ourselves to place a pawn at (i,j) if possible. We can do this by adding the constraint: for all (i,j):
$$-ny_{ij} + \sum_{(i,j)\in\mathcal{Q}_{ij}} x_{ij}) \;\leq\; 1/2.$$
Why does this work? If there are no queens placed that attack (i,j) then y_{ij} is free. If, however, at least one queen is there then we must have $y_{ij} = 1$ as otherwise the inequality fails (note the sum is at most n, so taking $y_{ij} = 1$ will ensure it is satisfied).

#5: Do Exercise 2.14 from my notes: Consider the following Linear Programming problem: $x_j \geq 0$,
$$\begin{pmatrix} 1 & 4 & 5 & 8 & 1 \\ 2 & 2 & 3 & 8 & 0 \\ 3 & 2 & 1 & 6 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 311 \\ 389 \\ 989 \end{pmatrix}, \tag{3.1}$$
and we want to minimize
$$5x_1 + 8x_2 + 9x_3 + 2x_4 + 11x_5. \tag{3.2}$$
Find (or prove one does not exist) an optimal solution.

Solution: There are several ways to go. We give a one-line solution from the TA at the end; as a large part of homework is to learn the methods and techniques, it is good to see the straightforward approach.

We have 5 columns, and a basic optimal solution (if it exists) must come from a basic feasible solution. There are $\binom{5}{3} = 10$ ways to choose 3 columns from 5 to find a basic feasible solution, and the basic feasible solution must have exactly 3 non-zero entries. We could look at all of these candidates and see which is the optimal solution; we know an optimal solution must exist, as the objective function is a positive linear combination of our variables. Thus the function is bounded, as each x_i satisfies $0 \leq x_i \leq 989$. Using standard results from analysis (a continuous function on a compact set attains its maximum and minimum).

We need to find a basic feasible solution. If we try the first three columns of A, we get $A'x = b$. As A is a 3×3 matrix with linearly independent columns it is invertible, and we get $x = A'^{-1}b$. Unfortunately $A'^{-1}b$ has a negative entry, and thus cannot be a basic feasible solution. Remember our method only generates ***candidates*** for basic feasible solutions; it cannot ensure that they ***are*** basic feasible.

Undaunted, we continue. We find that there are no basic feasible solutions – all of the candidates have a negative entry, and thus there are no solutions. Here is code to generate the matrices:

```
B = {{1, 4, 5}, {2, 2, 3}, {3, 2, 1}};
B = {{1, 4, 8}, {2, 2, 8}, {3, 2, 6}};
B = {{1, 4, 1}, {2, 2, 0}, {3, 2, 0}};
B = {{1, 5, 8}, {2, 3, 8}, {3, 1, 6}};
```

```
B = {{1, 5, 1}, {1, 3, 0}, {3, 1, 0}};
B = {{1, 8, 1}, {2, 8, 0}, {3, 6, 0}};
B = {{4, 5, 8}, {2, 3, 8}, {2, 1, 7}};
B = {{4, 5, 1}, {2, 3, 0}, {2, 1, 0}};
B = {{4, 8, 1}, {2, 8, 0}, {2, 6, 0}};
B = {{5, 8, 1}, {3, 8, 0}, {1, 6, 0}};
```

Here is code to check one of the cases:

```
B = {{4, 8, 1}, {2, 8, 0}, {2, 6, 0}};
Print["Our pruned matrix is ", MatrixForm[B]];
b = Transpose[{{311, 389, 989}}];
MatrixForm[b];
basicsoln = Inverse[B].b;
Print["Candidate for basic feasible is ", MatrixForm[basicsoln]];
```

We can try and solve this directly:

```
Clear[x1]; Clear[x2]; Clear[x3]; Clear[x4]; Clear[x5];
Solve[x1 + 4 x2 + 5 x3 + 8 x4 + x5 == 311 &&
  2 x1 + 2 x2 + 3 x3 + 8 x4 == 389
  && 3 x1 + 2 x2 + x3 + 6 x4 == 989, {x1, x2, x3, x4, x5}]
```

The output is $x1, x2$ free and

```
{{x3 -> -(2789/5) + (6 x1)/5 + (2 x2)/5,
  x4 -> 1289/5 - (7 x1)/10 - (2 x2)/5,
  x5 -> 5188/5 - (7 x1)/5 - (14 x2)/5}}
```

If we plot the three lines that arise from forcing x_3, x_4 and x_5 to be non-negative, we see that there is no solution to these inequalities that has all five variables positive. The Mathematica code is

```
Plot[{-x1 + 5188/14, (-7/4) x1 + 1289/2, -3 x1 + 2789/3}, {x1,0,400}]
```

and we give the plot in Figure 5.

Now, the one-line solution. These numbers were not randomly chosen (though I forgot when initially looking at this problem). I wanted something without any feasible solutions. If $(x_1, x_2, x_3, x_4, x_5) \geq (0, 0, 0, 0, 0)$ then there cannot be a solution to $Ax = b$. To see this, note that the sum of the entries in the j^{th} column in the first and second rows exceeds the value in the j^{th} column in the third row, ***but*** the sum of the first two entries of b is less than the third. There cannot be a solution. More mathematically, adding the first two constraints gives

$$3x_1 + 6x_2 + 8x_3 + 16x_4 + x_5 \ = \ 700,$$

while the third row is

$$3x_1 + 2x_2 + x_3 + 6x_4 \ = \ 989.$$

Subtracting yields

$$4x_2 + 7x_3 + 10x_4 + x_5 \ = \ -289,$$

which is impossible as all the x_i are supposed to be non-negative.

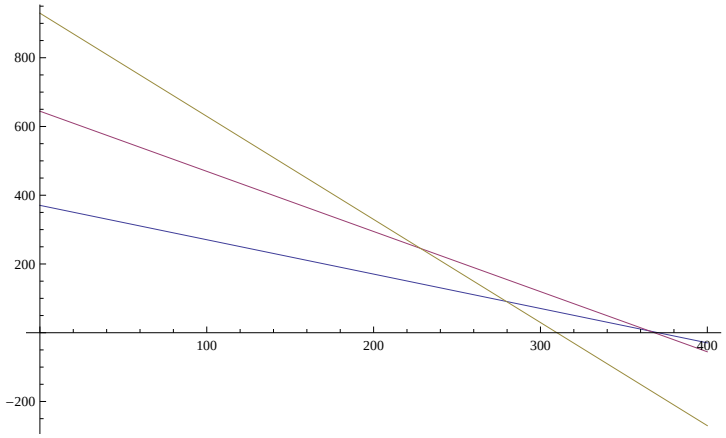

FIGURE 5. Plot of the three inequalities. The valid points are *below* the first and third lines (gold and blue) and *above* the middle (purple) line.

Remark: This (and the earlier problem with the 3×3 matrix) indicate the value of really looking at a problem and its algebra first before ploughing away. Often we can make our lives much easier by studying the problem, looking at symmetries, finding something to exploit. We ***can*** plug away, but we can save time. I consider Henry David Thoreau the patron saint of mathematics for his sage advice of `Simplify, simplify`. (Of course, this should be simplified to **Simplify**, but I'll grant him this as he has a point to make.) Look for savings first before doing calculations; this is in line with the spirit of duality and the savings available there.

Homework #5: Due Monday, October 8: #1: Submit to me (separate from the rest of your homework) an outline for your paper topic and class presentation. This should include a summary of what you want to discuss, what you want the class to get out of your presentation / writeup, what sources you believe you will use. #2: Write down linear constraints for the event A or B or C must happen. #3: Find as good of a function f as you can such that you can find infinitely many pairs of integer $x < y$ with the run-time of the Euclidean algorithm at least $f(x)$. For example, what we did in class shows you can't take $f(x) = 4\log_2(x)$; can you take $f(x) = c\log_2 x$ for some $c < 1$? #4: Consider an $n \times n \times n$ chesscube. Write down a linear programming problem to figure out how many hyperpawns can safely be placed given that n hyperqueens are placed in the chesscube. Note the hyperqueens can attack diagonally, horizontally, vertically, and forward-backly.

Extra Credit: for a couple of values of n, figure out the maximum number of pawns that can safely be placed on an $n \times n$ chessboard given that there are n queens that must be placed. Is this sequence in the OEIS (`http://oeis.org/`)?

4. HW #5: DUE OCTOBER 8, 2012

4.1. **Assignment:** Homework #5: Due Monday, October 8: #1: Submit to me (separate from the rest of your homework) an outline for your paper topic and class presentation. This should include a summary of what you want to discuss, what you want the class to get out of your presentation / writeup, what sources you believe you will use. #2: Write down linear constraints for the event A or B or C must happen. #3: Find as good of a function f as you can such that you can find infinitely many pairs of integer $x < y$ with the run-time of the Euclidean algorithm at least $f(x)$. For example, what we did in class shows you can't take $f(x) = 4\log_2(x)$; can you take $f(x) = c\log_2 x$ for some $c < 1$? #4: Consider an $n \times n \times n$ chesscube. Write down a linear programming problem to figure out how many hyperpawns can safely be placed given that n hyperqueens are placed in the chesscube. Note the hyperqueens can attack diagonally, horizontally, vertically, and forward-backly.

4.2. **Solutions:**

#2: Write down linear constraints for the event A or B or C must happen.
Solution: We start with decision variables x_A, x_B, x_C where $x_E = 1$ if event E happens and 0 if event E does not occur. We have the inclusive or; thus our constraint is simply $x_A + x_B + x_C \geq 1$. The only way this constraint fails is if $x_A = x_B = x_C = 0$, in other words, if none of the events happen.

#3: Find as good of a function f as you can such that you can find infinitely many pairs of integer $x < y$ with the run-time of the Euclidean algorithm at least $f(x)$. For example, what we did in class shows you can't take $f(x) = 4\log_2(x)$; can you take $f(x) = c\log_2 x$ for some $c < 1$?
Solution: One way is to start at the 'end' of the Euclidean algorithm and work backwards. If we end with the pair (1, 2), then the smallest pair we can have the step before is (2, 3) (as applying the Euclidean algorithm to that pair yields (1, 2)). Continuing to move back, we see the smallest pair leading to (2, 3) is the pair (3, 5) (as $5 = 1 \cdot 3 + 2$). Similarly, the smallest pair that goes to (3, 5) is (5, 8) (as $8 = 1 \cdot 5 + 3$). A pattern is emerging; we want the a's in the Euclidean algorithm to be 1 each time, and this leads the Fibonacci numbers as the remainders. It suggests we try $x = F_n$ and $y = F_{n+1} = F_n + F_{n-1}$. Using properties of the Fibonacci numbers ($F_{m+1} = F_m + F_{m-1}$ and $F_2 = 2$ and $F_1 = 1$), we see it takes $n - 1$ steps to get down to the pair (1, 2). To finish the problem, we need to know how big $x = F_n$ is (as a function of n). We use Binet's formula, which says

$$F_n \;=\; \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n \;\approx\; \frac{\phi^n}{\sqrt{5}},$$

where $\phi = (1+\sqrt{5})/2$ is the golden mean. See

`http://en.wikipedia.org/wiki/Fibonacci_number#Relation_to_the_golden_ratio`

for a proof.

All that's left is to express $n - 1$ as a function of $x = F_n$. Since $F_n \approx \phi^n/\sqrt{5}$, we get $F_n\sqrt{5} \approx \phi^n$ or $n \approx \log(F_n\sqrt{5})/\log(\phi)$; actually, since we want to express our answer in terms of logarithms base 2, it's better to use the base 2 logarithm and obtain $n \approx \log_2(F_n\sqrt{5})/\log_2(\phi)$, and thus

$$f(F_n) \;=\; n-1 \;\approx\; \frac{\log_2(F_n\sqrt{5})}{\log_2(\phi)} - 1 \;\approx\; \frac{1}{\log_2(\phi)}\log_2(F_n).$$

If we don't use the base 2 logarithm, then later we need to use the change of base formula $\log_b u/\log_b v = \log_v u$).

Thus, there are some inputs where the Euclidean algorithm will take on the order of the logarithm of x steps to run. We see

$$f(F_n) \;\approx\; \frac{1}{\log_2(\phi)}\log_2(F_n) \;=\; \log_\phi(2)\log_2(F_n)$$

(where we used a log-law to rewrite the constant in a nice way). Therefore the answer to the posed question is 'yes' in that we do get an answer of size logarithm of x, but our c is not less than 1 as $c \approx \log_\phi(2) \approx 1.44042$. This is the worst case scenario for the Euclidean algorithm, as it takes the most steps to get back to our initial pair.

Remark: it's worth noting how important the change of base formula is. Though often forgotten, it's one of the most important of the log laws. The reason is that if we can compute logarithms in one base, we can use this to get logarithms in any other. Thus, we only need `one` *table! Ah, efficiency!*

#4: Consider an $n \times n \times n$ chesscube. Write down a linear programming problem to figure out how many hyperpawns can safely be placed given that n hyperqueens are placed in the chesscube. Note the hyperqueens can attack diagonally, horizontally, vertically, and forward-backly.

Solution: We need to slightly generalize our arguments from the last assignment. Let $x_{ijk} = 1$ if we place a queen at (i, j, k) and 0 otherwise, and let $y_{ijk} = 1$ if there is a pawn at (i, j, k) and 0 otherwise. Let $\mathcal{Q}_{ijk}$ be the set of all locations that can attack (i, j, k).

Our first constraint is

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} x_{ijk} \ = \ 1;$$

this ensures we place exactly n queens on the board.

The second constraint is for the location of the pawns: for $1 \le i, j, k \le n$:

$$2n(1 - y_{ijk}) \ \ge \sum_{(i',j',k') \in \mathcal{Q}_{ijk}} x_{i'j'k'}.$$

If no queens attack (i, j, k) then the sum on the right is zero and there is no effect on y_{ijk}. If however there is at least one queen attacking the location (i, j, k) then the only way the inequality is satisfied is to have $y_{ijk} = 0$ (note in this case the sum on the right is non-zero, and is at most n as there are only n queens on the board).

The objective function to minimize is $-\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} y_{ij}$. This is the negative of the number of pawns that may safely be placed on the board. Note now that if we *can* place a pawn at (i, j, k) we will.

Homework #6: Due Monday, October 15: #1: Write an introduction to your problem / topic. Clearly state what you are going to tackle. At the very least, enumerate what additional material you'll need to discuss your topic that has not been covered in class. If there are items you want me to lecture on, let me know. Summarize some of the literature. This *must* be in TeX. It should be at least 3 full pages (this includes the bibliography, but not a title page!). This is meant to be the nucleus of your write-up. Presentations will start as soon as people are ready. You have anywhere from 20 minutes to 50 minutes. Take the time to do a good, thorough job. Your write-up should also be complete. You may assume your audience has the knowledge base of our class; anything that we haven't covered must be explained (though not necessarily proved).

5. HW #6: DUE OCTOBER 15, 2012

5.1. **Assignment:** Homework #6: Due Monday, October 15: #1: Write an introduction to your problem / topic. Clearly state what you are going to tackle. At the very least, enumerate what additional material you'll need to discuss your topic that has not been covered in class. If there are items you want me to lecture on, let me know. Summarize some of the literature. This *must* be in TeX. It should be at least 3 full pages (this includes the bibliography, but not a title page!). This is meant to be the nucleus of your write-up. Presentations will start as soon as people are ready. You have anywhere from 20 minutes to 50 minutes. Take the time to do a good, thorough job. Your write-up should also be complete. You may assume your audience has the knowledge base of our class; anything that we haven't covered must be explained (though not necessarily proved).

6. HW #8: DUE OCTOBER 29, 2012

6.1. **Assignment:** Due Monday October 29: #1: Give an example of a square matrix A such that there is no orthogonal matrix Q with $Q^T AQ$ a diagonal matrix. #2: Let Q be an orthogonal matrix. Must Q^3 be orthogonal? What about $Q + Q + Q$? Prove your claims. #3: Consider $N \times N$ real symmetric matrices such that each matrix element is at most B. Find as good as you can upper bound for the absolute values of the eigenvalues in terms of B and N. #4: A unitary matrix U is such that $U^H U = UU^H = I$, where H stands for the Hermitian of the matrix (this means taking the complex conjugate of the transpose). In class we proved the eigenvalues of real symmetric and complex Hermitian matrices are real. Discover and prove as much as you can about the eigenvalues of unitary matrices. What can you say about them? What about the eigenvalues of orthogonal matrices?

6.2. **Solutions:**

#1: Give an example of a square matrix A such that there is no orthogonal matrix Q with $Q^T AQ$ a diagonal matrix.

Solution: The standard example is $A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$. Note the eigenvalues of this matrix are 0, 0 but there is only one eigenvector direction, $\overrightarrow{e_1} = (1, 0)$. If this matrix could be diagonalized then we would have

$$Q^T AQ \ = \ \Lambda \ = \ \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix},$$

as the two eigenvalues are zero. By multiplying we see this implies $A = Q\Lambda Q^T$, which in this case is the zero matrix as Λ is the zero matrix. Thus A is the zero matrix, a contradiction. Hence there is a square matrix which cannot be diagonalized.

#2: Let Q be an orthogonal matrix. Must Q^3 be orthogonal? What about $Q + Q + Q$? Prove your claims.

Solution: If $(Q^3)(Q^3)^T = (Q^3)^T(Q^3) = I$ then Q^3 is orthogonal. We show the first claim holds as the second is similar. We have

$$(Q^3)(Q^3)^T \ = \ QQQQ^TQ^TQ^T \ = \ QQ(QQ^T)Q^TQ^T \ = \ QQIQ^TQ^T = QQQ^TQ^T.$$

We continue to argue like this, and find

$$(Q^3)(Q^3)^T \ = \ Q(QQ^T)Q^T \ = \ QIQ^T \ = \ QQ^T \ = \ I.$$

While Q orthogonal implies Q^3 is orthogonal, $Q + Q + Q$ is never orthogonal. The simplest way to see this is to note it equals $3Q$, and now

$$(3Q)(3Q)^T \ = \ 9QQ^T \ = \ 9I,$$

and as this is not I the matrix $3Q$ cannot be orthogonal.

#3: Consider $N \times N$ real symmetric matrices such that each matrix element is at most B. Find as good as you can upper bound for the absolute values of the eigenvalues in terms of B and N.

Solution: Consider $A\overrightarrow{v} \ = \ \lambda\overrightarrow{v}$. We may assume each entry of $\overrightarrow{v}$ is at most 1 in absolute value, and since we do not have the zero vector we may assume some element is exactly 1; without loss of generality let's assume it's the first component. This is similar to some of the normalizations we've done in linear programming (i.e., putting things into canonical form). Sometimes it's convenient to normalize the eigenvector to have *length* one. To figure out a bound for $|\lambda|$ let's look at each row of $A\overrightarrow{v}$. If we look at the first row, from $A\overrightarrow{v} = \lambda\overrightarrow{v}$ we get

$$a_{11}v_1 + a_{12}v_2 + \cdots + a_{1N}v_N \ = \ \lambda v_N \ = \ \lambda.$$

As each $|a_{ij}| \leq B$ and each $|v_k| \leq 1$, we get

$$|\lambda| \ = \ |a_{11}v_1 + a_{12}v_2 + \cdots + a_{1N}v_N| \ \leq \ NB.$$

Thus the eigenvalues are all at most NB, and this narrows down where in the plane we must search.

With a lot more work, more can be proved. A great result is the Gershgorin circle theorem; see for example

`http://en.wikipedia.org/wiki/Gershgorin_circle_theorem`

Another useful fact about eigenvalues of matrices (this time ones with either all positive entries, or at the very least no negative ones) is the Perron-Frobenius theorem:

`http://en.wikipedia.org/wiki/Perron%E2%80%93Frobenius_theorem`

#4: A unitary matrix U is such that $U^H U = UU^H = I$, where H stands for the Hermitian of the matrix (this means taking the complex conjugate of the transpose). In class we proved the eigenvalues of real symmetric and complex Hermitian matrices are real. Discover and prove as much as you can about the eigenvalues of unitary matrices. What can you say about them? What about the eigenvalues of orthogonal matrices?

Solution: Let $\overrightarrow{v}$ be an eigenvector of the unitary matrix U with eigenvalue λ. Then $U\overrightarrow{v} = \lambda\overrightarrow{v}$, and we have

$$||U\overrightarrow{v}||^2 \ = \ (U\overrightarrow{v})^H(U\overrightarrow{v}) \ = \ (\lambda\overrightarrow{v})^H(\lambda\overrightarrow{v}) \ = \ \overline{\lambda}\lambda\overrightarrow{v}^H\overrightarrow{v} \ = \ |\lambda|^2||\overrightarrow{v}||.$$

We now compute $||U\overrightarrow{v}||$ another way:

$$||U\overrightarrow{v}||^2 \ = \ (U\overrightarrow{v})^H(U\overrightarrow{v}) \ = \ \overrightarrow{v}^H U^H U\overrightarrow{v} \ = \ \overrightarrow{v}^H I\overrightarrow{v} \ = \ \overrightarrow{v}^H\overrightarrow{v} \ = \ ||\overrightarrow{v}||^2.$$

Thus

$$|\lambda|^2||\overrightarrow{v}|| \ = \ ||\overrightarrow{v}||;$$

as $||\overrightarrow{v}|| \neq 0$ we find $|\lambda| = 1$. Thus the eigenvalues of unitary matrices have absolute value one.

As orthogonal matrices are special cases of unitary matrices, their eigenvalues must be one in absolute value as well. The eigenvalues of real orthogonal matrices, however, do not need to be real. Consider for example a rotation by 90 degrees. There is no way this can have a real eigenvector. Its eigenvectors and eigenvalues are all complex. In matrix form, a rotation by θ radians is

$$R(\theta) \ = \ \begin{pmatrix} \cos\theta & 0\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}.$$

Thus

$$R(\pi/2) \ = \ \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}.$$

The sum of the eigenvalues is 0 and the product is 1. Thus we're solving $\lambda_1 + \lambda_2 = 0$ and $\lambda_1\lambda_2 = 1$. Using the first equation to eliminate λ_2 gives $-\lambda_1^2 = 1$, so $\lambda_1 = \pm i$ (and $\lambda_2 = \mp i$). Thus, while the eigenvalues of real orthogonal matrices are one in absolute value, they can have non-zero imaginary parts.

Homework due Monday, November 5: #1: Let $A = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$. Find the eigenvalues and eigenvectors *of unit length* of A. If $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ are these eigenvectors, let Q be the matrix where the first column is $\overrightarrow{v_1}$ and the second column is $\overrightarrow{v_2}$. Compute $Q^T AQ$. #2: Let $T_{n+1} = T_n + T_{n-1} + T_{n-2}$ with $T_0 = 0$, $T_1 = 0$ and $T_2 = 1$. Find the generating function for this sequence.

7. HW #9: DUE NOVEMBER 5, 2012

7.1. **Assignment:** Homework due Monday, November 5: #1: Let $A = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$. Find the eigenvalues and eigenvectors *of unit length* of A. If $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ are these eigenvectors, let Q be the matrix where the first column is $\overrightarrow{v_1}$ and the second column is $\overrightarrow{v_2}$. Compute $Q^T AQ$. #2: Let $T_{n+1} = T_n + T_{n-1} + T_{n-2}$ with $T_0 = 0$, $T_1 = 0$ and $T_2 = 1$. Find the generating function for this sequence.

7.2. **Solutions:**

#1: Let $A = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$. Find the eigenvalues and eigenvectors *of unit length* of A. If $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ are these eigenvectors, let Q be the matrix where the first column is $\overrightarrow{v_1}$ and the second column is $\overrightarrow{v_2}$. Compute $Q^T AQ$.

Solution: To find the eigenvalues, one way is to use $\det(A - \lambda I) = 0$. As

$$A \ = \ \begin{pmatrix} 1-\lambda & 2 \\ 2 & 1-\lambda \end{pmatrix},$$

we have

$$\det(A - \lambda I) \ = \ (1-\lambda)^2 - 4,$$

or $1 - \lambda = \pm 2$ so $\lambda = 3$ or -1. We now solve $(A - \lambda I)\overrightarrow{v} = \overrightarrow{0}$. Thus for $\lambda = 3$ we find

$$\begin{pmatrix} -2 & 2 \\ 2 & -2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \ = \ \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

which implies $x = y$. As we want the vector to have length 1 (its length-squared is $x^2 + y^2$), we can't take $x = y = 1$ but instead need $x^2 + y^2 = 2x^2 = 1$, so we take $x = y = 1/\sqrt{2}$. The second eigenvalue, $\lambda = -1$, gives us

$$\begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \ = \ \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

so $x = -y$. As we want the length to be one, we take $x = -y = 1/\sqrt{2}$.

Our matrix Q is thus

$$Q \ = \ \begin{pmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix} \ = \ \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix};$$

notice that $Q^T Q = QQ^T = I$, and standard matrix multiplication gives $Q^T AQ = \begin{pmatrix} 3 & 0 \\ 0 & -1 \end{pmatrix}$, the diagonal matrix of eigenvalues.

Some comments are in order. If we were to switch the two eigenvectors, we would switch the two diagonal entries. Also, there are faster ways to find the eigenvalues and eigenvectors. I call it the Boola-Boola Theorem. If each row of a matrix sums to c then c is an eigenvalue with corresponding eigenvector all 1s. For our matrix, as each row sums to 3 then 3 is an eigenvalue with corresponding eigenvector (1,1) or, if we want unit length, $(1/\sqrt{2}, 1/\sqrt{2})$. As the sum of the eigenvalues is the trace (which is 2), the other eigenvalue is -1. Further, the eigenvectors associated to distinct eigenvalues of a real symmetric matrix are perpendicular, and thus the other eigenvector is in the direction $(-1, 1)$, or as a unit vector $(1/\sqrt{2}, 1/\sqrt{2})$.

#2: Let $T_{n+1} = T_n + T_{n-1} + T_{n-2}$ with $T_0 = 0$, $T_1 = 0$ and $T_2 = 1$. Find the generating function for this sequence.

Solution: We discussed generating functions when studying the Fibonaccis. The proof is similar here. Set

$$g(x) \ = \ \sum_{n=0}^{\infty} T_n x^n \ = \ \sum_{n=2}^{\infty} T_n x^n$$

as $T_0 = T_1 = 0$. We want to use the recurrence relation, so let's pull out the $n = 2$ term:

$$\begin{aligned} g(x) &= T_2x^2 + \sum_{n=3}^{\infty} T_n x^n \\ &= x^2 + \sum_{m=2}^{\infty} T_{m+1}x^{m+1} \\ &= x^2 + \sum_{m=2}^{\infty} (T_m + T_{m-1} + T_{m-2})x^{m+1} \\ &= x^2 + x\sum_{m=2}^{\infty} T_m x^m + x^2 \sum_{m=2}^{\infty} T_{m-1}x^{m-1} + x^3 \sum_{m=2}^{\infty} T_{m-2}x^{m-2} \\ &= x^2 + xg(x) + x^2 \sum_{n=1}^{\infty} T_n x^n + x^3 \sum_{n=0}^{\infty} T_n x^n \\ &= x^2 + xg(x) + x^2 g(x) + x^3 g(x). \end{aligned}$$

Thus

$$(1 - x - x^2 - x^3)g(x) \;=\; x^2,$$

or

$$g(x) \;=\; \frac{x^2}{1 - x - x^2 - x^3}.$$