Algorithmic Game Theory and Applications
Exercise 1

s0840449

1 Question 1

1.1 Part (a)

The definition of expected payoff, where x is a mixed strategy profile and u_i is the utility function, is:

$$U_i(x) := \sum_{s \in S} x(s) * u_i(s)$$

The probability values $x(s)$ can be represented as the matrix P:

$$P = x_1^T x_2 = \begin{pmatrix} \frac{1}{4} * 0 & \frac{1}{4} * \frac{2}{3} & \frac{1}{4} * \frac{1}{3} \\ \frac{3}{4} * 0 & \frac{3}{4} * \frac{2}{3} & \frac{3}{4} * \frac{1}{3} \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{6} & \frac{1}{12} \\ 0 & \frac{1}{2} & \frac{1}{4} \\ 0 & 0 & 0 \end{pmatrix}$$

Letting G_i be the game matrix for player i, $U_i(x)$ is given by the sum of the elements of the matrix formed by an element-wise multiplication of G_i and P.

$$\begin{aligned} U_1(x) &= sum(G_1 * P) \\ &= \frac{1}{6} * 4 + \frac{1}{12} * 6 + \frac{1}{2} * 7 + \frac{1}{4} * 7 \\ &= \frac{4}{6} + \frac{1}{2} + \frac{7}{2} + \frac{7}{4} \\ &= \frac{77}{12} \approx 6.42 \end{aligned}$$

$$\begin{aligned} U_2(x) &= sum(G_2 * P) \\ &= \frac{1}{6} * 8 + \frac{1}{12} * 4 + \frac{1}{2} * 4 + \frac{1}{4} * 5 \\ &= \frac{8}{6} + \frac{4}{12} + \frac{4}{2} + \frac{5}{4} \\ &= \frac{59}{12} \approx 4.92 \end{aligned}$$

1.2 Part (b)

I used an algebraic approach to compute the Nash Equilibrium. As I solved this before we discussed dominated strategies, I did not reduce the game matrix.

In the following, let the rows of the game matrix be labelled top (T), middle (M), and bottom (B), and the columns left (L), center (C), and right (R) (see Figure 1).

$$\begin{array}{c|ccc|} & \textbf{L} & \textbf{C} & \textbf{R} \\ \textbf{T} & (6,2) & (4,8) & (6,4) \\ \textbf{M} & (3,6) & (7,4) & (7,5) \\ \textbf{B} & (4,7) & (5,4) & (9,5) \end{array}$$

Figure 1: The labelled game matrix.

First, let Player 1 play the strategy $x = (t, m, 1 - t - m)$. This gives the following expected payoffs for Player 2 for each pure strategy:

$$U_2(L) = 2t + 6m + 7(1 - t - m) = 7 - 5t - m$$

$$U_2(C) = 8t + 4m + 4(1 - t - m) = 4 + 4t$$

$$U_2(R) = 4t + 5m + 5(1 - t - m) = 5 - t$$

In a Nash Equilibrium, the payoff for all pure strategies played with positive probability must be the same. Assuming $U_2(L) = U_2(C) = U_2(R)$ gives the linear equations:

$$7 - 5t - m = 4 + 4t$$

$$7 - 5t - m = 5 - t$$

Solving for t and m gives:

$$m = \frac{6}{5}, \quad t = \frac{1}{5}$$

As $m > \frac{4}{5}$, this is obviously not a solution. Therefore, one of the probabilities t, m, or $1 - m - t$ must be 0. Assume first that $t + m = 1$, i.e. that row B has zero probability in the Nash Equilibrium. This reduces the linear equations to:

$$7 - 5t - m = 4 + 4t$$

$$t + m = 1$$

Solving this gives a strategy of $x = (\frac{1}{4}, \frac{3}{4}, 0)$ for Player 1, which is a valid strategy. To double-check, compare the values of $U_2(L)$, $U_2(C)$ and $U_2(R)$. Substituting in x gives $U_2(L) = U_2(C) = 5$, which is correct as they should be equal, and $U_2(R) = 4.75$, which is correct as it should be less than the other two.

Since Player 1 will play the bottom row, Player 2 needs to equalize only $U_1(T)$ and $U_1(M)$. Again assuming that Player 2 is playing a three-way split strategy $x = (p, q, 1 - p - q)$ gives an invalid answer ($q = 3, p = \frac{7}{4}$), so assume that $q = p - 1$. Form the following equations (where $U_1(B)$ is **not** used, but included for later checking):

$$U_1(T) = 6p + 4(1 - p) = 4 + 2p$$

$$U_1(M) = 3p + 7(1 - p) = 7 - 4p$$

$$U_1(B) = 4p + 5(1 - p) = 5 - p$$

Solving for p by equating the first two gives:

$$p = \frac{1}{2}$$

So Player 2 will be playing the strategy $x = (\frac{1}{2}, \frac{1}{2}, 0)$. Again check the utilities $U_1(T)$, $U_1(M)$, and $U_1(B)$, finding that $U_1(T) = U_1(M) = 5$, and $U_1(B) = 4.5$.

To summarise, a Nash Equilibrium for the bimatrix is:

$$x^* = ((\frac{1}{4}, \frac{3}{4}, 0), (\frac{1}{2}, \frac{1}{2}, 0))$$

1.3 Part (c)

There is no pure Nash Equilibrium. This can be seen by considering what a pure Nash Equilibrium looks like for the game. Since Player 1 must choose one of the rows with probability 1, and Player 2 must choose a column with probability 1, the pure Nash Equilibrium state is a single cell of the matrix.

Therefore, to show that there is no pure Nash Equilibrium for this game, it suffices to exhaustively show that in each cell either Player 1 will want to change the row to get a more beneficial cell, or Player 2 will want to change the column to get a more beneficial cell.

In the following paragraphs, let $G_{i,j}$ denote the cell in the i-th row and j-th column of the game matrix.

$\boldsymbol{G}_{1,1}$: Player 2 can achieve a higher payoff by changing the column number to 2 or 3, and thus this is not an equilibrium cell.

$\boldsymbol{G}_{1,2}$: Player 1 can achieve a higher payoff by changing the row number to 2 or 3, and thus this is not an equilibrium cell.

$\boldsymbol{G}_{1,3}$: Both players can achieve higher payoffs by changing the row or column respectively (i.e. to $G_{2,3}$ or $G_{1,2}$), and thus this is not an equilibrium cell.

$\boldsymbol{G}_{2,1}$: Player 1 can achieve a higher payoff by changing the row number to 1 or 3, and thus this is not an equilibrium cell.

$\boldsymbol{G}_{2,2}$: Player 2 can achieve a higher payoff by changing the column number to 1 or 3, and thus this is not an equilibrium cell.

$\boldsymbol{G}_{2,3}$: Both players can achieve higher payoffs by changing the row or column respectively (i.e. to $G_{3,3}$ or $G_{2,1}$), and thus this is not an equilibrium cell.

$\boldsymbol{G}_{3,1}$: Player 1 can achieve a higher payoff by changing the row number to 1, and thus this is not an equilibrium cell.

$\boldsymbol{G}_{3,2}$: Both players can achieve higher payoffs by changing the row or column respectively (i.e. to $G_{2,2}$ or $G_{3,1}$, and thus this is not an equilibrium cell.

$\boldsymbol{G}_{3,3}$: Player 2 can achieve a higher payoff by changing the column number to 1, and thus this is not an equilibrium cell.

Since no cell is an equilibrium cell, there is no pure Nash Equilibrium for this game.

2 Question 2

Let $x^* = (x_1^*, x_2^*) \in X$, $x_1^* = \{p_1, ..., p_5\}$ and $x_2^* = \{q_1, ..., q_5\}$, be a mixed Nash Equilibrium for the given game.

Solving first for Player 1, define the objective and constraints for player 1 to be a min-maximizer. The objective is, of course, to maximize a payoff value v. The constraints can be split into two types. The first type of constraints are 'probability constraints', which ensure that Player 1 is playing a valid strategy:

$$\forall i,\ p_i \geq 0,\ and \sum_{i=1}^{5} p_i = 1$$

The second set of constraints are 'value constraints', which ensure that the value that Player 1 is maximising, v, is a minimum. The constraints are therefore that each possible expected payoff must be at least v:

$$4p_1 + 8p_2 + 2p_3 + 4p_4 + 2p_5 \geq v$$
$$2p_1 + 3p_2 + 4p_3 + 9p_4 + 5p_5 \geq v$$
$$p_1 + 5p_2 + 6p_3 + 3p_4 + 6p_5 \geq v$$
$$9p_1 + 6p_2 + p_3 + 4p_4 + 4p_5 \geq v$$
$$5p_1 + 2p_2 + 7p_3 + 6p_4 + p_5 \geq v$$

Writing this all in linear programming style:

Find $x_1^* = \{p_1, p_2, p_3, p_4, p_5\}$ so as to:

Maximize: v

Subject to:

$$4p_1 + 8p_2 + 2p_3 + 4p_4 + 2p_5 \geq v,$$
$$2p_1 + 3p_2 + 4p_3 + 9p_4 + 5p_5 \geq v,$$
$$p_1 + 5p_2 + 6p_3 + 3p_4 + 6p_5 \geq v,$$
$$9p_1 + 6p_2 + p_3 + 4p_4 + 4p_5 \geq v,$$
$$5p_1 + 2p_2 + 7p_3 + 6p_4 + p_5 \geq v,$$
$$\forall i,\ p_i \geq 0,\ and$$
$$\sum_{i=1}^{5} p_i = 1$$

The solution given by maple's *simplex* solver is:

$$x_1^* = \{\frac{10}{113}, \frac{48}{113}.\frac{61}{226}, \frac{49}{226}, 0\},\ v = \frac{1013}{226}$$

Applying the same logic for Player 2, but instead aiming to *minimize* the payoff value v where v is a maximum (so that Player 2 is playing a max-minimizer), gives the linear program:

Find $x_2^* = \{q_1, q_2, q_3, q_4, q_5\}$ so as to:

Minimize: v

Subject to:

$$4q_1 + 2q_2 + q_3 + 9q_4 + 5q_5 \leq v,$$
$$8q_1 + 3q_2 + 5q_3 + 6q_4 + 2q_5 \leq v,$$
$$2q_1 + 4q_2 + 6q_3 + 1q_4 + 7q_5 \leq v,$$
$$4q_1 + 9q_2 + 3q_3 + 4q_4 + 6q_5 \leq v,$$
$$2q_1 + 5q_2 + 6q_3 + 4q_4 + q_5 \leq v,$$
$$\forall i,\ q_i \geq 0,\ \textit{and}$$
$$\sum_{i=1}^{5} q_i = 1$$

The solution to this is:

$$x_2^* = \{0, \frac{19}{226}, \frac{43}{113}.\frac{71}{226}, \frac{25}{113}\},\ v = \frac{1013}{226}$$

Therefore the minimax profiles for the game are as above, and the minimax value is $v = \frac{1013}{226} \approx 4.48$.

3 Question 3

3.1 Part (a)

The Nash Equilibrium $x^* \in X$ is:

$$x^* = (x_1^*, x_2^*),\ \ x_1^* = x_2^* = \{\frac{1}{2}, \frac{1}{2}\}$$

This can trivially be seen by solving algebraically as in Question 1. Assuming that Player 1 plays $(p, 1-p)$, Player 2 has the following payoffs (using the same column notation as before):

$$U_2(L) = -p + 1 - p = 1 - 2p$$
$$U_2(R) = p - 1(1-p) = -1 + 2p$$

Solving this gives $p = \frac{1}{2}$. A similar argument works for Player 2's probabilities, giving the above unique mixed strategy.

3.2 Part (b)

The experimental code for this question was written in Python and can be seen in Appendix A. If run, you should get output like (logging prefixes removed for clarity):

```
Trying starting pair (T, T)
Initial setup:
Player 1's (pure) strategy: (0, 1)
Player 2's (pure) strategy: (0, 1)
Finished after 144494 rounds, with final mixed strategies:
Player 1's statistical mixed strategy: (72366/144493, 72127/144493)
Player 2's statistical mixed strategy: (72103/144493, 72390/144493)

Trying starting pair (T, H)
Initial setup:
Player 1's (pure) strategy: (0, 1)
Player 2's (pure) strategy: (1, 0)
Finished after 157290 rounds, with final mixed strategies:
Player 1's statistical mixed strategy: (78512/157289, 78777/157289)
Player 2's statistical mixed strategy: (78801/157289, 78488/157289)

Trying starting pair (H, T)
Initial setup:
Player 1's (pure) strategy: (1, 0)
Player 2's (pure) strategy: (0, 1)
Finished after 145771 rounds, with final mixed strategies:
Player 1's statistical mixed strategy: (7303/14577, 7274/14577)
Player 2's statistical mixed strategy: (36503/72885, 36382/72885)
```

```
Trying starting pair (H, H)
Initial setup:
Player 1's (pure) strategy: (1, 0)
Player 2's (pure) strategy: (1, 0)
Finished after 146114 rounds, with final mixed strategies:
Player 1's statistical mixed strategy: (72911/146113, 73202/146113)
Player 2's statistical mixed strategy: (72935/146113, 73178/146113)
Experiment took 32.60 seconds
```

This shows that over a large number of iterations, the value of the statistical mixed strategies for both players falls to $0.5 \pm \epsilon$, $\epsilon = 0.001$, for 50 consecutive iterations.

3.3 Part (c)

Not attempted.

4 Question 4

4.1 Part (a)

For simplicity, first convert all Opt_j to `Maximize` - this can be done by multiplying the objective function f_j by -1 for any j for which $Opt_j =$ `Minimize`. Additionally, let $x = \{x_1, x_2, ..., x_n\}$.

Next, repeat the following for $i = 1, ..., r$:

1. Form the linear program 'Maximize $f_i(x)$, subject to $C(x)$.
2. This will either return that there is no solution, an unbounded solution, or an optimal solution x^* given the constraints.
3. In the case of an unbounded or no solution, return failure.
4. Otherwise, substitute x^* into f_i, obtaining the optimal *value* v_i.
5. Add the following constraint to $C(x)$: $f_i(x) = v_i$[1].

The solution x^* from step 2 when $i = r$ is the solution to the whole problem.

[1]This can of course be re-written in simple form as $-f_i(x) \leq -v_i$. The $\geq$ case can be ignored as $f_i(x)$ cannot be higher than v_i so there is no need to bound it on that side.

The reason that this works can be seen by considering each iteration in turn. The initial iteration solves the basic linear program, finding an optimal solution for f_1 (or failing if no solution exists) and an optimal value v_1.

From there, an optimal solution for f_2 'among those vectors that [optimize f_1]' can be found by making it a requirement (adding a constraint) that any solution to f_2 gives the same value v_1 for f_1. This logic then repeats - once an optimal value for f_2 (given the new constraint) is found, it is added as a constraint for the next linear program, and so on.

Note that if there is any solution for f_1, there must be a solution for f_j, $j > 1$, but there may not be a bounded solution. For example, consider $f_1(x) = x_1 + x_2$, $f_2 = x_2$, and $C(x) = \{x_1 + x_2 \leq 6\}$. Then f_1 has bounded (although infinite) solutions, but among those infinite solutions f_2 does not have a bounded one - since x_1 can be negative then x_2 can be infinitely high!

4.2 Part (b)

To compute the dual of this linear program, first represent it in 'Primal' form:

Maximize: $c^T x$

Subject to:

$$Ax \leq b,$$

$$\forall i,\ x_i \geq 0$$

For this problem:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} = \begin{pmatrix} 2 & 1 & 0 \\ 3 & 0 & 4 \\ 0 & 4 & 5 \end{pmatrix},$$

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix},\ b = \begin{pmatrix} 9 \\ 4 \\ 10 \end{pmatrix},\ c = \begin{pmatrix} 3 \\ 2 \\ 4 \end{pmatrix}$$

To now take the dual of the linear program, form the new linear program (in which $y = \{y_1, y_2, y_3\}$):

Minimize: $b^T y$

Subject to:

$$A^T y \geq c,$$
$$\forall i,\ y_i \geq 0$$

Or, in non-primal form:

Minimize: $9y_1 + 4y_2 + 10y_3$

Subject to:

$$\begin{gathered} 2y_1 + 3y_2 \geq 3, \\ y_1 + 4y_3 \geq 3, \\ 4y_2 + 5y_3 \geq 4, \\ \forall i,\ y_i \geq 0 \end{gathered}$$

Solving these linear programs (using xmaple) gives the solutions:

$$x_1 = \frac{4}{3},\ x_2 = \frac{5}{2},\ x_3 = 0$$
$$y_1 = 0,\ y_2 = 1,\ y_3 = \frac{1}{2}$$

5 Question 5

Firstly I will show that if an x exists such that $Ax \leq b$, there is no vector y such that $y \geq 0$, $y^T A = \mathbf{0}$, and $y^T b \leq 0$. This can be seem by simple algebra. Assume that an x exists that solves the linear equation above. Now if such a y exists:

$$\begin{aligned} Ax &\leq b \\ y^T Ax &\leq y^T b \\ y^T Ax &< 0 \text{ (Since } y^T b < 0) \\ \mathbf{0}x &< 0 \text{ (Since } y^T A = \mathbf{0}) \\ 0 &< 0 \end{aligned}$$

This is an obvious contradiction, as a number cannot be less than itself. Therefore, in the case where an x exists that solves $Ax \leq b$, there cannot be a y satisfying $y \geq 0$ and $y^T A = \mathbf{0}$ and $y^T b \leq 0$.

Next, it must be shown that if there does **not** exist a solution x for the system $Ax \leq b$, there **must** exist a y satisfying $y \geq 0$ and $y^T A = \mathbf{0}$ and $y^T b \leq 0$. This will be done via induction on the number of columns in A, using Fourier-Motzkin elimination.

Base case. The base case occurs when there is only one column in A. As such, $x = \{x_1\}$, and $A = \{a_1, ..., a_m\}^T$.

Consider the variables $a_i \in A$. If any a_i is 0, then an unsolveable system can be created by setting $b_i < 0$ (so that the inequality is $0 \leq b_i < 0$). A suitable y for such a system can be found by letting $y_i = 1$, and $y_j = 0$ $(j \neq i)$. Then $y^T A = 0$ since $a_i = 0$, and $y^T b < 0$ since $b_i < 0$.

Next, consider what happens if all a_i are non-zero. Denote by a_j any positive element of A, and a_k any negative element. All inequalities then take one of two forms:

$$a_j x_1 \leq b_j, \text{ or}$$

$$a_k x_1 \leq b_k$$

Dividing through by the element of A:

$$x_1 \leq \frac{b_j}{a_j}$$

$$x_1 \geq \frac{b_k}{a_k}$$

For there to be any solution to $Ax \leq b$, the smallest 'j' fraction must be greater than or equal to the largest 'k' fraction:

$$max \frac{b_k}{a_k} \leq min \frac{b_j}{a_j}$$

Therefore to form an unsolveable system, let some $\frac{b_{k'}}{a_{k'}}$ be greater than some $\frac{b_{j'}}{a_{j'}}$. A suitable y must now be found. Let $y_{k'} = a_{j'}$, $y_{j'} = -a_{k'}$, and all other $y_i = 0$. Solving for $y^T A$:

$$y^T A = y_{k'} a_{k'} + y_{j'} a_{j'} = a_{j'} a_{k'} + (-a_{k'}) a_{j'} = 0$$

Next, solving for $y^T b$ (recalling that $a_{k'} < 0$ and $|a_{j'} b_{k'}| < |a_{k'} b_{j'}|$):

$$y^T b = y_{k'} b_{k'} + y_{j'} b_{j'} = a_{j'} b_{k'} + a_{k'} b_{j'} < 0$$

Therefore, for any formation of a one column matrix A where there is no solution x, there exists a y such that $y \geq 0$ and $y^T A = \mathbf{0}$ and $y^T b \leq 0$. This means that the base case holds.

Inductive Step. Assume that for less than n columns, the claim that 'if there does **not** exist a solution x for the system $Ax \leq b$, there **must** exist a y satisfying $y \geq 0$ and $y^T A = \mathbf{0}$ and $y^T b \leq 0$' holds. I will show that based on this assumption, it also holds for n columns, which will complete the inductive proof.

For a n-column matrix A, the inequality $Ax \leq b$ expands to:

$$\sum_{j=0}^{n} a_{i,j} x_j \leq b_i \text{, for i = 1, ..., m}$$

Or, row-wise:

$$\begin{aligned} a_{1,1}x_1 + ... + a_{1,n}x_n &\leq b_1 \\ &... \\ a_{m,1}x_1 + ... + a_{m,n}x_n &\leq b_m \end{aligned}$$

To remove x_n using Fourier-Motzkin elimination, the following steps are performed:

- Re-write all constraints for which $a_{i,n}x_n \neq 0$ as either $x_n \leq \frac{...}{a_{i,n}}$ or $x_n \geq \frac{...}{a_{i,n}}$. Notice that the second case occurs when $a_{i,n} < 0$. Call the first set of indices P, and the second set N.
- Generate all pairs of P and N (i.e. the cartesian product), such that $\frac{...}{a_{i,n}} \leq \frac{...}{a_{j,n}}$ for $i \in N$, $j \in P$. Add these constraints to the system.

Consider now what the inequalities introduced by Fourier-Motzkin will look like:

$$\frac{b_i - a_{i,1}x_1 - ... - a_{i,n-1}x_{n-1}}{a_{i,n}} \leq \frac{b_j - a_{j,1}x_1 - ... - a_{j,n-1}x_{n-1}}{a_{j,n}}, \text{ for } i \in N, j \in P$$

Multiplying through by the denominators, and remembering that $a_{i,n} < 0$:

$$\begin{aligned} a_{j,n}(b_i - a_{i,1}x_1 - ... - a_{i,n-1}x_{n-1}) &\geq a_{i,n}(b_j - a_{j,1}x_1 - ... - a_{j,n-1}x_{n-1}) \\ a_{j,n}b_i - a_{j,n}a_{i,1}x_1 - ... - a_{j,n}a_{i,n-1}x_{n-1} &\geq a_{i,n}b_j - a_{i,n}a_{j,1}x_1 - ... - a_{i,n}a_{j,n-1}x_{n-1} \end{aligned}$$

Re-arranging to look more like the other linear inequalities:

$$a_{j,n}a_{i,1}x_1+...+a_{j,n}a_{i,n-1}x_{n-1}-a_{i,n}a_{j,1}x_1-...-a_{i,n}a_{j,n-1}x_{n-1} \leq a_{j,n}b_i-a_{i,n}b_j$$

A useful trick can now make this simpler. Multiplying any inequality in a system of linear inequalities by a positive value will not change whether or not there is a solution, and will not change the existence of y - a new y' can be created by multiplying the corresponding entry of y by the value. Therefore, it can be assumed without loss of generality that all $a_{i,n} < 0$ are -1, and all $a_{j,n} > 0$ are 1.

Making this assumption simplifies the inequality:

$$\begin{aligned} a_{i,1}x_1 + ... + a_{i,n-1}x_{n-1} + a_{j,1}x_1 + ... + a_{j,n-1}x_{n-1} &\leq b_i + b_j \\ (a_{i,1} + a_{j,1})x_1 + ... + (a_{i,n-1} + a_{j,n-1})x_{n-1} &\leq b_i + b_j \end{aligned}$$

So, the new constraints, in the system will have right-hand sides of the form $b_i + b_j$, for $i \in N$, $j \in P$. Call the modified constraint vector b'.

The new system, A', has $n-1$ columns and no solution (since the original n-column system did not have one). Therefore, the induction hypothesis holds for it - there is a y' such that $y' \geq 0$, $y'^T A' = \mathbf{0}$, and $y'^T b' < 0$.

Expand $y'^T b' < 0$, remembering that b' consists of the unchanged indices U, and the new rows $b_i + b_j$ ($i \in N$, $j \in P$):

$$\begin{aligned} y'^T b' &< 0 \\ \sum_{u \in U} y'_u b_u + \sum_{i \in N, j \in P} y'_{ij}(b_j + b_i) &< 0 \\ \sum_{u \in U} y'_u b_u + \sum_{i \in N, j \in P} y'_{ij} b_j + y'_{ij} b_i &< 0 \\ \sum_{u \in U} y'_u b_u + \sum_{i \in N, j \in P} y'_{ij} b_j + \sum_{i \in N, j \in P} y'_{ij} b_i &< 0 \\ \sum_{u \in U} y'_u b_u + \sum_{j \in P} (\sum_{i \in N} y'_{ij}) b_j + \sum_{i \in N} (\sum_{j \in P} y'_{ij}) b_i &< 0 \end{aligned}$$

Next, expand $y'^T A' = 0$, noting that A' consists of the unchanged rows A_i, $i \in U$, and new rows $A'_{ij} = A_i + A_j$ (where ij is a single index with $i \in N$, $j \in P$).

$$y'^T A' = \mathbf{0}$$

$$\sum_{u \in U} y'_u A_{u,c} + \sum_{i \in N, j \in P} y'_{ij}(A_{j,c} + A_{i,c}) = 0 \text{ for columns c = 1, ..., n-1}$$

$$\sum_{u \in U} y'_u A_{u,c} + \sum_{i \in N, j \in P} y'_{ij} A_{j,c} + y'_{ij} A_{i,c} = 0$$

$$\sum_{u \in U} y'_u A_{u,c} + \sum_{i \in N, j \in P} y'_{ij} A_{j,c} + \sum_{i \in N, j \in P} y'_{ij} A_{i,c} = 0$$

$$\sum_{u \in U} y'_u A_{u,c} + \sum_{j \in P} (\sum_{i \in N} y'_{ij}) A_{j,c} + \sum_{i \in N} (\sum_{j \in P} y'_{ij}) A_{i,c} = 0$$

These two expansions give a y for the original n column system: let $y_i = y'_i$ for $i \in U$, $y_j = \sum_{i \in N} y'_{ij}$ for $j \in P$, and $y_i = \sum_{j \in P} y'_{ij}$ for $i \in N$. Such a y causes the above expansions to turn into:

$$\sum_{u \in U} y'_u b_u + \sum_{j \in P} (\sum_{i \in N} y'_{ij}) b_j + \sum_{i \in N} (\sum_{j \in P} y'_{ij}) b_i < 0$$

$$\sum_{i=0}^{m} y_i b_i < 0$$

$$y^T b < 0$$

(Similarly for $y^T A = \mathbf{0}$.)

Therefore, there exists a y for the n-column system such that $y \geq 0$, $y^T A = \mathbf{0}$, and $y^T b \leq 0$, which completes our inductive proof!

A Python Code for Question 3b

```
from fractions import Fraction
import logging
import random
import time


logging.basicConfig()
_logger = logging.getLogger("question_3b")
_logger.setLevel(logging.INFO)
```

```
def run_experiment(player_1_starting, player_2_starting):
  # Only the value for heads is stored. Tails is '1 - player_i'.
  player_1_heads = player_1_starting
  player_2_heads = player_2_starting
  number_rounds = 1

  # Used to track the error. The results are taken to be true once
  # the error has been below 0.001 for both players for 50 rounds.
  # (Those numbers are of course chosen rather arbitrarily, since this
  # is all experimental rather than mathematical.)
  consecutive_errors = 0

  _logger.info("Initial setup:")
  _logger.info("Player 1's (pure) strategy: (%s, %s)", player_1_heads,
      1 - player_1_heads)
  _logger.info("Player 2's (pure) strategy: (%s, %s)", player_2_heads,
      1 - player_2_heads)

  while consecutive_errors < 50:
    expected_player_1 = Fraction(player_1_heads, number_rounds)
    expected_player_2 = Fraction(player_2_heads, number_rounds)

    _logger.debug("Round %s:", number_rounds)
    _logger.debug("Player 1's statistical mixed strategy: (%s, %s)",
        expected_player_1, 1 - expected_player_1)
    _logger.debug("Player 2's statistical mixed strategy: (%s, %s)",
        expected_player_2, 1 - expected_player_2)

    # Find player 1's pure best response.
    if expected_player_2 > Fraction(1, 2):
      # Player 2 is more likely to play heads, so player 1 should play heads
      # as well.
      player_1_move = 1
    elif expected_player_2 < Fraction(1, 2):
      # Player 2 is more likely to play tails, so player 1 should play tails
      # as well.
      player_1_move = 0
    else:
      # The ratio is exactly 50:50, so guess.
      player_1_move = random.randint(0, 1)

    # Find player 2's pure best response.
    if expected_player_1 > Fraction(1, 2):
      # Player 1 is more likely to play heads, so player 2 should play tails.
      player_2_move = 0
    elif expected_player_1 < Fraction(1, 2):
      # Player 1 is more likely to play tails, so player 2 should play heads.
      player_2_move = 1
    else:
      # The ratio is exactly 50:50, so guess.
      player_2_move = random.randint(0, 1)

    # Update the error stats.
    player_1_error = abs(0.5 - float(expected_player_1))
    player_2_error = abs(0.5 - float(expected_player_2))
    _logger.debug("Errors: %s, %s", player_1_error, player_2_error)
    if player_1_error < 0.001 and player_2_error < 0.001:
      consecutive_errors += 1
    else:
      consecutive_errors = 0

    number_rounds += 1
    player_1_heads += player_1_move
    player_2_heads += player_2_move

  _logger.info("Finished after %s rounds, with final mixed strategies:",
      number_rounds)
  _logger.info("Player 1's statistical mixed strategy: (%s, %s)",
      expected_player_1, 1 - expected_player_1)
  _logger.info("Player 2's statistical mixed strategy: (%s, %s)",
      expected_player_2, 1 - expected_player_2)
  print


def convert(num):
  return 'H' if num == 1 else 'T'
```

```
def main():
  before = time.time()
  for i in range(2):
    for j in range(2):
      _logger.info("Trying starting pair (%s, %s)", convert(i), convert(j))
      run_experiment(i, j)
  after = time.time()
  _logger.info("Experiment took %.2f seconds", after - before)


if __name__ == '__main__':
  main()
```