1. (a) Suppose Bo somehow learns Alex's strategy vector a. Describe a linear program whose solution is Bob's best possible strategy vector.

Solution: The only variables are Bo's probabilities b_j. The constraints ensure that b really is a probability distribution.

$$\begin{array}{ll} \text{minimize} & a^T M \cdot b \\ \text{subject to} & \sum_j b_j = 1 \\ & b_j \geq 0 \quad \text{for all } j \end{array}$$

The objective vector is the vector-matrix product $c = a^T M$. Each coefficient $c_j = (a^T M)_j = \sum_i M_{ij} a_i$ of the objective vector is Alex's expected score if Alex uses randomized strategy a and Bo deterministically chooses the number j. ■

Rubric: 2 points

(b) What is the dual of your linear program from part (b)?

Solution: Because the linear program in part (a) has only one non-sign constraint, the dual linear program has only one variable z, and the objective "vector" is the right side 1 of that primal constraint.

$$\begin{array}{ll} \text{maximize} & z \\ \text{subject to} & z \leq \sum_i M_{ij} a_i \quad \text{for all } j \end{array}$$

The variable z represents a lower bound on Alex's expected payoff, no matter what Bo chooses. The maximum value for that lower bound is the *minimum* coefficient of the vector $a^T M$. ■

Rubric: 2 points

(c) So what *is* Bo's optimal strategy, as a function of the vector a? And what is Alex's resulting expected score? (You should be able to answer this part even without answering parts (a) and (b).)

Solution: Bo's optimal strategy is to *deterministically* pick the index i that minimizes Alex's expected score $(a^T M)_j = \sum_i M_{ij} a_i$, and Alex's resulting expected score is exactly $\min_j (a^T M)_j$.

As a concrete example, suppose Alex chooses the uniform Undercut strategy $(\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6})$—roll a fair 6-sided die and subtract 1. Then Alex's expected-score vector is

$$a^T M = \begin{pmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \end{pmatrix} \begin{pmatrix} 0 & 1 & -2 & -3 & -4 & -5 \\ -1 & 0 & 3 & -2 & -3 & -4 \\ 2 & -3 & 0 & 5 & -2 & -3 \\ 3 & 2 & -5 & 0 & 7 & -2 \\ 4 & 3 & 2 & -7 & 0 & 9 \\ 5 & 4 & 3 & 2 & -9 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} \frac{13}{6} & \frac{7}{6} & \frac{1}{6} & -\frac{5}{6} & \mathbf{-\frac{11}{6}} & -\frac{5}{6} \end{pmatrix},$$

so Bo should *always* choose 4, which makes Alex's expected score $-11/6$. ■

Rubric: 2 points

(d) Now suppose that Alex knows that Bo will discover Alex's strategy vector before they actually start playing. Describe a linear program whose solution is Alex's best possible strategy vector.

Solution: The following linear program has $n+1$ variables: Alex's probabilities a_i and one additional variable z, which is a lower bound on Alex's expected score *no matter what Bo chooses*. Alex wants to make this lower bound as large as possible.

$$\begin{array}{lrl} \text{maximize} & z & \\ \text{subject to} & \sum_i M_{ij} a_i - z \geq 0 & \text{for all } j \\ & \sum_i a_i = 1 & \\ & a_i \geq 0 & \text{for all } i \end{array}$$

Again, the last two constraints guarantee that a is a probability distribution. ■

Rubric: 2 points

(e) What is the dual of your linear program from part (d)?

Solution: The dual linear program has $n+1$ variables: one variable d_j corresponding to each inequality constraint in the primal LP, plus one more variable y corresponding to the equality constraint.

$$\begin{array}{lrl} \text{minimize} & y & \\ \text{subject to} & \sum_j M_{ij} d_j + y \geq 0 & \text{for all } i \\ & \sum_j d_j = -1 & \\ & d_j \leq 0 & \text{for all } j \end{array}$$

Huh. The d_i's seem to represent *negations* of probabilities. Let's rewrite this in terms of new variables $b_i = -d_i$; notice the changes in red.

$$\begin{array}{lrl} \text{minimize} & y & \\ \text{subject to} & \sum_j M_{ij} b_j - y \leq 0 & \text{for all } i \\ & \sum_j b_j = \mathbf{1} & \\ & b_j \geq 0 & \text{for all } j \end{array}$$

This looks eerily similar to the LP from part (d)! In fact, the solution to this LP is *Bo's* optimal strategy vector b (and the resulting expected score y) if that strategy is known to *Alex* in advance. The variable y is an *upper* bound on Alex's expected score, no matter what *Alex* chooses; Bo wants to make this *upper* bound as *small* as possible.

The fundamental theorem of linear programming implies that the optimal objective values for these two linear programs are identical. So we've actually derived a proof of von Neumann's *minmax theorem*:

$$\max_a \min_b a^T M b = \min_b \max_a a^T M b$$

■

Rubric: 2 points. This is more detail than necessary for full credit.

(f) **Extra credit:** So what *is* Alex's optimal Undercut strategy, if Alex knows that Bo will know that strategy?

Solution: Solving the LP in part (d) gives us

$$\begin{aligned} a^* &= \left(0,\ \tfrac{10}{66},\ \tfrac{26}{66},\ \tfrac{13}{66},\ \tfrac{16}{66},\ \tfrac{1}{66}\right) \\ &\approx (0,\ 0.15152,\ 0.39394,\ 0.19697,\ 0.24242,\ 0.01515) \end{aligned}$$

with an objective value of 0.

I used `scipy.optimize.linprog` to compute a decimal approximation of the solution. Once we know that $a_0^* = 0$ and all other probabilities a_i^* are positive, the exact rational solution can be obtained by solving a system of linear equations via Cramer's rule: All relevant determinants are integers, because the constraint matrix of the LP is integral.

We can verify that this strategy is optimal as follows. The symmetry in the rules of Undercut implies that Alex and Bo's optimal strategies must be identical, so let $b^* = a^*$. It follows that the optimal expected score is exactly zero. Brute force computation implies that $a^*M = (\frac{25}{11}, 0, 0, 0, 0, 0)$ and (therefore!) $Mb^* = (-\frac{25}{11}, 0, 0, 0, 0, 0)$, so the solution vector $(a^*, 0) = (b^*, 0) = (0, \frac{10}{66}, \frac{26}{66}, \frac{13}{66}, \frac{16}{66}, \frac{1}{66}, 0)$ is feasible for both the LP in part (d) and its dual in part (e). Because its primal and dual objective values for are equal (to 0), this must be an optimal solution! ■

Rubric: 2 points

(g) **Extra credit:** If Bo knows that Alex is going to use their optimal strategy from part (f), what is Bo's optimal Undercut strategy?

Solution: Never choose 0.

If Alex uses their optimal undercut strategy a^*, Bo's expected score vector is $a^*M = (\frac{25}{11}, 0, 0, 0, 0, 0)$. So as long as Bo uses a strategy—deterministic or randomized—that never chooses 0, the resulting expected score is exactly 0.

Symmetrically, if Bo uses the same optimal strategy $b^* = (0, \frac{10}{66}, \frac{26}{66}, \frac{13}{66}, \frac{16}{66}, \frac{1}{66})$, then *no matter what strategy Alex uses*, the expected score is at most 0, and as long as *Alex* never chooses 0, the expected score is exactly 0.

Maybe this is why Hofstadter's original version of Undercut didn't allow choosing 0?

In the column where he introduced Undercut, Hofstadter describes a competition between a program he wrote, which tries to guess the other player's next move, and another program written by Jon Peterson, which used this optimal randomized strategy. *"It was a humiliating and infuriating experience for me to watch my program, with all its 'intelligence', struggle in vain to overcome the blind randomness of Jon's program. But there was no way out. I was most disappointed to learn that, in some sense, the 'most intelligent' strategy of all not only was dumb-it even paid no attention whatsoever to the enemy's moves! Something about this seemed directly opposite to the original aim of Undercut, which was to have players trying to psych each other out to ever deeper levels."*

This problem would have been slightly more interesting without thumbs. If Alex and Bo can only name integers between 0 and 4, their optimal strategy vector is $(0, \frac{5}{10}, \frac{2}{10}, \frac{3}{10}, 0)$, and Alex's expected payoff vector is $(\frac{4}{5}, 0, 0, 0, \frac{1}{5})$. So in this variant, the smallest option 0 and the largest option 4 are both bad choices. It's only the presence of 5 that makes 4 a good choice! ■

Rubric: 2 points. 1 point for "Copy Alex's strategy".

2. A three-dimensional matching in an undirected graph G is a collection of vertex-disjoint triangles (cycles of length 3) in G. A three-dimensional matching is ***maximal*** if it is not a proper subgraph of a larger three-dimensional matching in the same graph.

   (a) Let M and M' be two arbitrary maximal three-dimensional matchings in the same underlying graph G. ***Prove*** that $|M| \leq 3 \cdot |M'|$.

   Solution: Consider an arbitrary triangle $\triangle$ in M. If $\triangle$ does not share a vertex with any triangle in M', then $M' + \triangle$ is a 3D matching, contradicting our assumption that M' is maximal. We conclude that every triangle in M shares at least one vertex with a triangle in M'.

   On the other hand, consider an arbitrary triangle $\triangle'$ in M'. Each vertex of $\triangle'$ is a vertex of *at most* one triangle in M, because the triangles in M are vertex-disjoint. Thus, the number of triangles in M is at most the number of *vertices* in M'. ■

   Rubric: 4 points = 2 for every triangle in M touches a triangle in M' + 2 for each triangle in M touches at most three triangles in M'.

   (b) Finding the *largest* three-dimensional matching in a given graph is NP-hard. Describe and analyze a fast 3-approximation algorithm for this problem.

   Solution: The following brute-force algorithm computes a maximal 3D matching in $O(VE)$ ***time***.

```
DUMB3DMATCHING(G):
  unmark every vertex of G
  M ← ∅
  for all vertices v
      for all edges uv
          for all edges vw
              if u ≠ w and uw ∈ E and u, v, w are all unmarked
                  mark u, v, w
                  add uvw to M
  return M
```

   Let M denote the 3D matching computed by our algorithm, and let OPT denote the largest 3D matching in G. Part (a) immediately implies $|OPT|/|M| \leq 3$. ■

   Rubric: 3 points = 1 for algorithm + 1 for time analysis + 1 for proof of approximation. No penalty for $O(V^3)$; we just want something that runs in polynomial time

(c) Finding the *smallest maximal* three-dimensional matching in a given graph is NP-hard. Describe and analyze a fast 3-approximation algorithm for this problem.

Solution: We use exactly the same algorithm as part (b)!

Let M denote the 3D matching computed by our algorithm, and let OPT denote the smallest maximal 3D matching in G. Part (a) immediately implies $|M|/|OPT| \leq 3$. ■

Rubric: 3 points = 1 for algorithm + 1 for time analysis + 1 for proof of approximation. "See part (b)" is worth exactly the same number of points as earned in part (b).

3. You are designing a digital circuit where each component can operate in one of two states: ***high signal*** (logical 1) or ***low signal*** (logical 0). The circuit includes several directed connections (edges), each with an associated non-negative weight representing the importance or strength of the connection. For a connection to function properly, the signal must flow from a component in the high signal state to a component in the low signal state, and the total weight of such valid connections should be maximized.

   You are given a directed, edge-weighted graph $G = (V, E, w)$ representing the circuit, where each vertex $i \in V$ corresponds to a circuit component, each directed edge $i \to j \in E$ corresponds to a connection between components, and the weight $w_{i\to j} \geq 0$ of each edge $i \to j$ indicates the importance of that edge.

   Now suppose we assign some vertices to state ***high*** and the rest to state ***low***. The ***weight*** of this high/low assignment is the sum of the weights of all edges $i \to j$ such that vertex i is high and vertex j is low. Your task is to find a high/low assignment with the maximum possible weight.

   (a) Describe a *simple*, self-contained, and efficient randomized algorithm for this problem that finds a high/low assignment whose weight is within a factor of 4 of optimal. No LPs are necessary!

Solution: For each vertex $i \in V$, we independently assign it a state **high** with probability $1/2$, otherwise we assign it a state of **low**. Let H (resp. L) denote the set of vertices that are assigned high (resp. low). The expected total weight of edges going from high to low is

$$\begin{aligned}
\mathrm{E}\left[\sum_{i\to j\in E} w_{ij} \cdot [i \in H \text{ and } j \in L]\right] &= \sum_{i\to j\in E} w_{ij} \cdot \mathrm{E}[[i \in H \text{ and } j \in L]] \\
&= \sum_{i\to j\in E} w_{ij} \cdot \Pr[i \in H \text{ and } j \in L] \\
&= \sum_{i\to j\in E} w_{ij} \cdot \Pr[i \in H] \cdot \Pr[j \in L] \\
&= \frac{1}{4} \sum_{i\to j\in E} w_{ij}.
\end{aligned}$$

Since the total weight of *any* assignment is at most $\sum_{i\to j\in E} w_{ij}$, the above is within a factor of 4 of the optimal value. ■

Rubric: 2 points.

(b) Prove that the following integer linear program (ILP) is an exact formulation of our assignment problem: every high/low assignment to the vertices gives an ILP solution whose objective value is at least the weight of the assignment, and every ILP solution gives an assignment whose weight is at least the objective value.

$$\begin{aligned}
\text{maximize} \quad & \sum_{i\to j\in E} w_{i\to j} z_{i\to j} \\
\text{subject to} \quad & z_{i\to j} \le x_i && \text{for each edge } i\to j \in E \\
& z_{i\to j} \le 1 - x_j && \text{for each edge } i\to j \in E \\
& z_{i\to j} \in \{0,1\} && \text{for each edge } i\to j \in E \\
& x_i \in \{0,1\} && \text{for each vertex } i \in V
\end{aligned}$$

Solution: Suppose we have integer vectors $x \in \{0,1\}^V$ and $z \in \{0,1\}^E$ that satisfies the constraints of the ILP, and let $W = \sum_{i\to j\in E} w_{i\to j} z_{i\to j}$ be the corresponding objective value. We construct an assignment with weight W as follows. For each vertex $i \in V$, if $x_i = 1$, we assign i a state of **high** and if $x_i = 0$, we assign i a state of **low**. For each edge $i\to j$, we have $z_{i\to j} = 1$ iff both $x_i = 1$ and $x_j = 0$. Thus, $z_{i\to j} = 1$ if and only if i is high and j is low. It follows that the weight of this assignment is exactly W.

No suppose we are given a high/low assignment with weight W. Define $x_i = 1$ for all vertices i that are assigned **high** and $x_j = 0$ for all vertices j that are assigned **low**. We also define $z_{i\to j} = 1$ if edge $i\to j$ goes from **high** to **low** — that is, if i is **high** and j is **low** — and zero otherwise. The resulting integer vectors $x \in \{0,1\}^V$ and $z \in \{0,1\}^E$ satisfy all the constraints of our ILP, and the value of the ILP on this solution is W. ■

Rubric: 3 points = 1½ for each direction

(c) The LP relaxation of the above ILP is obtained by replacing the integer constraints with $0 \le z_{i\to j} \le 1$ for each edge $i\to j$ and $0 \le x_i \le 1$ for each vertex i. Consider the following randomized rounding algorithm:

i. First solve the LP relaxation of the ILP from part (b). Let (z^*, x^*) denote the solution to this LP.

ii. Then independently assign each vertex i a state of **high** with probability $\frac{1}{4} + \frac{x_i^*}{2}$ and **low** otherwise.

Prove that in expectation, this algorithm yields a 2-approximation to the optimal value for our assignment problem.

Solution: Let (z^*, x^*) be the optimal *fractional* solution to the LP, and let $OPT^* = \sum_{i\to j\in E} w_{i\to j} z^*_{i\to j}$ denote its optimal objective value. Similarly, let OPT be the optimal objective value for the *integer* linear program, or equivalently, the weight of the optimal high/low assignment. We immediately have $OPT^* \ge OPT$.

Let (H, L) be the random assignment obtained from x^* and z^* by our randomized rounding algorithm, where H is the set of vertices assigned **high** and $L = V \setminus H$ is the set of vertices assigned **low**. Let $W(H, L)$ denote the weight of this assignment.

Define $X \in \{0,1\}^E$ and $Z \in \{0,1\}^E$ to be the **integer** vectors derived from the assignment (H, L) as in part (b):

- For each vertex i, we define $X_i = [i \in H]$
- For each edge $i\to j$, we define $Z_{i\to j} = [X_i = 1 \text{ and } X_j = 0]$

Note that the vectors X and Z are random variables. Part (b) implies that $W(H, L)$ is exactly the objective value $\sum_{i\to j\in E} w_{i\to j} Z_{i\to j}$ of the ILP. Thus, its expected value is

$$\begin{aligned}
\mathrm{E}[W(H,L)] &= \mathrm{E}\left[\sum_{i\to j\in E} w_{i\to j} Z_{i\to j}\right] \\
&= \sum_{i\to j\in E} w_{i\to j} \cdot \mathrm{E}[Z_{i\to j}] \\
&= \sum_{i\to j\in E} w_{i\to j} \cdot \Pr[X_i = 1 \text{ and } X_j = 0] \\
&= \sum_{i\to j\in E} w_{i\to j} \cdot \Pr[X_i = 1] \cdot \Pr[X_j = 0] \\
&= \sum_{i\to j\in E} w_{i\to j} \cdot \left(\frac{1}{4} + \frac{x_i^*}{2}\right) \cdot \left(\frac{1}{4} + \frac{1 - x_j^*}{2}\right) \\
&= \frac{1}{4} \sum_{i\to j\in E} w_{i\to j} \cdot \left(\frac{1}{2} + x_i^*\right) \cdot \left(\frac{1}{2} + (1 - x_j^*)\right).
\end{aligned}$$

Note that for any real numbers α and β we have

$$\left(\frac{1}{2}+\alpha\right)\left(\frac{1}{2}+\beta\right) \geq 2 \cdot \min\{\alpha, \beta\}.$$

To see this, we may assume without loss of generality that $\alpha \leq \beta$, in which case the desired inequality reduces to

$$\left(\frac{1}{2}+\alpha\right)^2 \geq 2 \cdot \alpha \iff \left(\frac{1}{2}-\alpha\right)^2 \geq 0.$$

This inequality implies

$$\begin{aligned}
\mathrm{E}[W(H,L)] &= \frac{1}{4}\sum_{i\to j\in E} w_{i\to j} \cdot \left(\frac{1}{2}+x_i^*\right) \cdot \left(\frac{1}{2}+(1-x_j^*)\right) \\
&\geq \frac{1}{2}\sum_{i\to j\in E} w_{i\to j} \cdot \min\{x_i^*, 1-x_j^*\} \\
&\geq \frac{1}{2}\sum_{i\to j\in E} w_{i\to j} z_{i\to j}^* \\
&= \frac{1}{2} OPT^* \\
&\geq \frac{1}{2} OPT.
\end{aligned}$$

■

Rubric: 5 points.