1. Describe and analyze an efficient algorithm that either solves Dumbledore's problem, or correctly reports that there is no valid assignment whose total cost is finite.

Solution: We modify the standard reduction from bipartite matching to maximum flows. Define a flow network $G = (V, E)$ as follows:

- V contains a source vertex s, one vertex p_i for each professor, and one vertex c_j for each committee.
- E contains two types of edges:
  - Edges $s \to p_i$ for each professor, each with capacity 3 and cost 0.
  - Edges $p_i \to c_j$ for each professor i and committee j such that professor i is both suitable and willing to serve on committee j. Each such edge has capacity 1 and cost equal to that professor's price for serving on that committee.
- For each committee j, the balance $b(c_j)$ is the number of instructors required for that committee. The source vertex s has balance $-\sum_j b(c_j)$, and all faculty vertices have balance 0.

Let f^* be the minimum-cost feasible flow in G. Because all capacities and balances are integers, we can assume that f^* is integral, so we can decompose f^* into $|f^*|$ paths of the form $s \to p_i \to c_j$. For each such path, assign professor i to committee j.

We compute the minimum-cost flow in G using the FastSuccessiveShortestPaths algorithm described in the lecture notes. The algorithm halts after at most $3n$ iterations, because the edges leaving s have total capacity $3n$, and each iteration runs in $O(E \log V)$ time. Thus, the overall running time is $O(nE \log V) = O(VE \log V) = \boldsymbol{O(N^2 \log N)}$ ***time***, where $N \le 2cn$ is the total size of the input lists. (Orlin's algorithm, cited in the same section of the notes, yields a running time of $O(N^2 \log^2 N)$.)

We can prove that the algorithm is correct by following the usual strategy for matchings.

- Suppose there is a valid committee assignment with cost k. Construct a flow $f : E \to \mathbb{R}$ as follows:

$$f(s \to p_i) = \text{the number of committees assigned to professor } i$$
$$f(p_i \to c_j) = 1 \text{ if professor } i \text{ assigned to committee } j, \text{ 0 otherwise}$$

  Routine definition-chasing implies that f is a feasible flow with cost k.
- On the other hand, given a feasible flow f with cost k, our algorithm constructs a valid committee assignment with cost k.

We conclude that the cost of the cheapest feasible flow in G is equal to the cost of the cheapest valid committee assignment; thus, our algorithm is correct. ■

Rubric: 10 points: standard graph-reduction rubric. No penalty for using a slower polynomial-time minimum-cost flow algorithm, or even claiming minimum cuts can be solved polynomial time without specifying an algorithm.

2. Suppose we are given a sequence of n linear inequalities of the form $a_i x + b_i y \le c_i$. Collectively, these n inequalities describe a convex polygon P in the plane.

   (a) Describe a linear program whose solution describes the largest square with horizontal and vertical sides that lies entirely inside P.

Solution: The following linear program has three variables: the coordinates (x, y) of the bottom left corner of the square, and the side length w of the square.

$$\begin{array}{lllll}
\text{maximize} & & w & & \\
\text{subject to} & a_i x & + b_i y & \le c_i & \text{for all } i \\
& a_i(x+w) & + b_i y & \le c_i & \text{for all } i \\
& a_i x & + b_i(y+w) & \le c_i & \text{for all } i \\
& a_i(x+w) & + b_i(y+w) & \le c_i & \text{for all } i
\end{array}$$

The four groups of inequalities respectively state that the bottom-left corner, the bottom-right corner, the top-left corner, and the top-right corner lie inside P. Every other point (x', y') in the square is a weighted average of these four corner points, so it also satisfies the inequality $a_i x' + b_i y' \le c_i$ for every index i, and therefore lies inside P. ■

Rubric: 5 points = 1 for objective + 2 for variables + 2 for constraints

(b) Describe a linear program whose solution describes the largest circle that lies entirely inside P.

Solution: The Euclidean distance from any point (p,q) to the line $ax + by = c$ is exactly

$$\frac{ap + bq - c}{\sqrt{a^2 + b^2}}.$$

For each index i, precompute the following values:

$$\hat{a}_i = \frac{a_i}{\sqrt{a_i^2 + b_i^2}} \qquad \hat{b}_i = \frac{b_i}{\sqrt{a_i^2 + b_i^2}} \qquad \hat{c}_i = \frac{c_i}{\sqrt{a_i^2 + b_i^2}}$$

Then the lines $\hat{a}_i x + \hat{b}_i y = \hat{c}_i$ and $a_i x + b_i y = c_i$ are identical, and for any point (p,q), we have $\hat{a}_i p + \hat{b}_i q \le \hat{c}_i$ if and only if $a_i p + b_i q \le c_i$.

The following linear program has three variables: the coordinates (p,q) of the center of the circle, and the radius r of the circle.

$$\boxed{\begin{array}{l}\text{maximize } r \\ \text{subject to } \hat{a}_i p + \hat{b}_i q \le \hat{c}_i - r \quad \text{for all } i\end{array}}$$

Equivalently, as described in Wikipedia:

$$\boxed{\begin{array}{l}\text{maximize } r \\ \text{subject to } a_i p + b_i q + \left(\sqrt{a_i^2 + b_i^2}\right) r \le c_i \quad \text{for all } i\end{array}}$$

Yes, this is still a linear program, even though it contains that square-root expression, because that expression involves only the *given data* a_i and b_i. Said differently, we can precompute the value $d_i = \sqrt{a_i^2 + b_i^2}$ for each index i, and then write the linear program as follows:

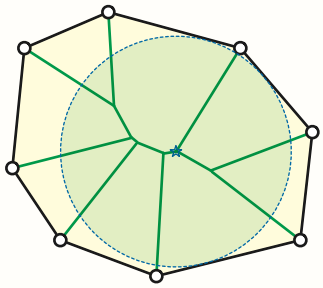
$$\boxed{\begin{array}{l}\text{maximize } r \\ \text{subject to } a_i p + b_i q + d_i r \le c_i \quad \text{for all } i\end{array}}$$

Each inequality specifies that the center point (p,q) not only satisfies the constraint $a_i p + b_i q \le c_i$ but also has Euclidean distance at least r from the line $a_i x + b_i y = c_i$. Thus, for any feasible solution (p,q,r), the entire circle of radius r centered at (p,q) lies inside each halfplane $a_i p + b_i q \le c_i$, and therefore inside the polygon P. ■

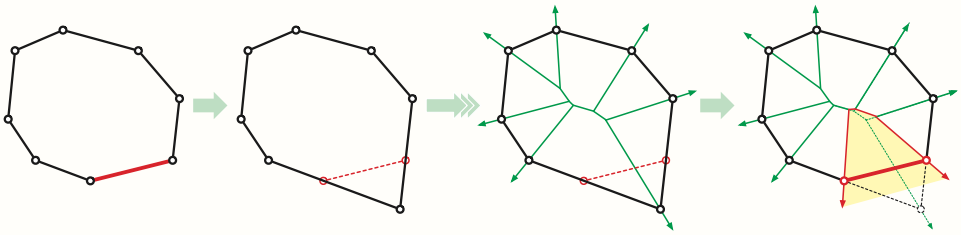
Rubric: 5 points = 1 for objective + 2 for variables + 2 for constraints. No credit if the objective function or the constraints involve non-linear functions *of the solution variables*.

Solution (computational geometry FTW): The largest circle inside P is centered at one of the $O(n)$ vertices of the *medial axis* of P. The medial axis is the set of all points in the interior of P that have more than one closest point on the boundary of P.

Because P is a convex polygon, the medial axis is a tree of line segments that partition the interior of P into smaller convex polygons ("faces"). Assuming P has m edges, the medial axis has at most $m-2$ interior vertices and $2m-3$ edges. Every vertex of the medial axis is the center of a circle that touches the boundary of P at three points.



The medial axis of P can be computed in $O(n)$ expected time using a simple randomized incremental algorithm. We remove one of the constraints that define P, chosen uniformly at random, recursively compute the medial axis of the larger polygon, add the removed constraint back in, and update the medial axis by carving out a new face. With careful bookkeeping, the time for the last update is proportional to the number of medial-axis edges on the new face. But because the constraint defining the new face is chosen uniformly at random, the *expected* number of medial-axis edges on that face is at most $(4m-6)/n < 4$.



Each vertex of the medial axis knows which edges of P are closest, so we can compute the largest circle centered at any medial-axis vertex in $O(1)$ time. We compute all such circles in $O(n)$ time and return the largest.

For more details, take Sariel's computational geometry course next semester!

[This solution is arguably equivalent to the previous one: The medial axis of P is actually the projection of the feasible polyhedron of the previous linear program into the (p,q)-plane!] ■

3. (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 which probabilities are positive and which are zero, 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?

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".