02-713 Homework #10: Linear Programming
Due: Apr. 28 by 9:30am

Don't look up solutions in the web. The point of these homeworks is to prepare for the exams. You may talk with your classmates about these problems, but you **must write up your solutions independently**, without using common notes or worksheets. You must indicate at the top of your homework with whom you worked. Your write up should be clear and concise. It should be submitted via Autolab (`https://autolab.cs.cmu.edu/02713-s14/`) as a typeset PDF.
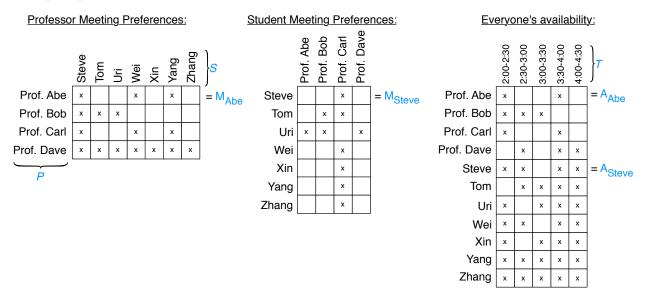
1. **"Thom's Problem."** On graduate school visit days, professors need to meet with visiting students to explain to them how great CMU is. Visiting students have certain professors they want to meet with and professors have certain visiting students they want to impress. Over visit day, there are meetings at various fixed time slots, and some professors and students are only available at certain times. Your task is to find an assignment of students, professors, and times that allows as many meetings to take place as possible.

   More formally, we have:

   - A set of visiting students $S = \{s_1, \ldots, s_m\}$.
   - A set of professors $P = \{p_1, \ldots, t_n\}$.
   - A set of time slots $T = \{t_1, \ldots, t_a\}$.
   - A collection of sets $A_{p_i} \subseteq T$, where A_{p_i} gives the time slots that professor p_i is available.
   - A collection of sets $A_{s_i} \subseteq T$, where A_{s_i} gives the time slots that student s_i is available.
   - A collection of sets $M_{p_i} \subseteq S$, where M_{p_i} gives the students with whom professor p_i is interested in meeting.
   - A collection of sets $M_{s_i} \subseteq P$, where M_{s_i} gives the professors that visiting student s_i wants to meet.

   Professor p_i will only meet with students on her list M_{p_i} and student s_i will only meet with professors on their list M_{s_i}. (We'll relax these requirements later.) A meeting can only happen at time t_i if both the student and professor are available. At any time, each student can meet with at most 1 professor and each professor can meet with at most 1 student.

   Example input:

(a) Write an integer linear program that finds maximum the number of (prof, student, time) meetings that can happen.

(b) Explain how to get the actual meeting schedule from your integer linear program.

(c) Suppose you are now also given, for each professor, a "bigshot score" b_{p_i} that is a positive number that says how important it is to satisfy that professor's demands. For every meeting that professor p_i takes with a student in her preference list M_{p_i}, we get b_{p_i} points. Describe how to modify your integer linear program to find a schedule that maximizes the number of bigshot points your schedule gets.

(d) One more extension: visit days are successful if students meet with a lot of people, even if neither the student or the professor put one another in their preference lists. Describe how to modify your integer linear program so that you get b_{p_i} points if you schedule one of professor p_i's meetings with a student on her list M_{p_i} but only 1 point if you schedule her with someone who is not on her preference list. Assume now that students will meet with anyone they are assigned to, but the time constraints must still be satisfied.

2. **"Greg's problem (sort of)."** You have two images A and B. Within each image are sets of points $P_A = \{p_1^A, \ldots, p_m^A\}$ and $P_B = \{p_1^B, \ldots, p_n^B\}$. Each point has a label $\ell(p_i)$, and several points will have the same labels. You are given a set $E \subseteq P_A \times P_B$ of pairs of points that are allowed to be assigned to each other.

You want to assign points in P_A to points in P_B subject to the following conditions:

(1) all points in A must be assigned to some point in B (some points in B might go assigned though).

(2) two points p_i^A, p_j^B can be assigned to each other only if $(p_i^A, p_j^B) \in E$.

(3) two points p_i^A, p_j^B can be assigned to one another only if they have the same label: $\ell(p_i^A) = \ell(p_j^B)$.

(4) several points from A can be be assigned to a single point in B, but we require that the number of points assigned to points with the same label in B to differ by at most 1 (including those to which we assign 0 points). That is, if 2 points from A are assigned to point p_1^B then if $\ell(p_3^B) = \ell(p_1^B)$, we can't assign 4 points from A to p_3^B. Another example: if we assign 0 points to p_7^B then no other point with the same label can be assigned more than 1 point from A.

(a) Give a network flow (or circulations with demands-type) formulation to find the assignments that satisfy the above constraints or report that no such assignment exists.

(b) Give a linear program to solve the same problem.

3. **Programming problem; 02-713 only.** This is a flashback to dynamic programming, and isn't directly connected with LP or network flows. Suppose you are trying to find where genes are in a long string of DNA d. Researchers have created a number of programs that predict possible gene locations, but they often include a lot of false and contradictory predictions. Suppose you are given a set $\mathcal{G} = \{(s_i, e_i, r_i)\}$ of predictions where $0 \leq s_i < |d|$ is the start of predicted gene i, $1 \leq e_i < |d|$ is the end of the predicted gene, and $0 < r_i < 1$ is probability that the prediction is correct. The predicted genes in $\mathcal{G}$ can overlap, but in real genomes overlapping genes are somewhat rare (they actually happen, but for the purposes of this problem, assume that they do not).

Write a Python program that implements a dynamic programming algorithm to find a subset $G \subseteq \mathcal{G}$ of predictions such that (1) no pair of predictions in G overlap, and (2) $\sum_{g \in G} r_g$ is maximized. (For simplicity, we are dealing with only 1 strand of DNA at a time.)

Input: A file containing $|\mathcal{G}|$ lines of the form 10 100 0.234, meaning that a gene was predicted to go from position 10 to position 100 with probability 0.234.

Output: A series of lines of the form `Include:   10 100 0.234`, meaning to include the gene with those parameters. A single line of the form `Sum Probability:   0.01`, that gives the sum of the probabilities of the chosen set of genes.

Technicalities: Your program should be called `choosegenes.py`. You can use matplotlib, numpy, or networkx, or anything in the python standard library. Otherwise, you can use no other code that you didn't write.