Foundations of Algorithms, Fall 2022
Homework #7

In this course, critical thinking and problem analysis results in discovering appropriate and better, the best, algorithm for a problem; defining the algorithm in pseudocode; demonstrating (we don't usually prove) that the algorithm is correct; and determining the asymptotic runtime for the algorithm using one or more of the tools for asymptotic analysis you have studied this semester. Each algorithm you design must follow the **Pseudocode Restrictions** (Canvas page *Pseudocode Restrictions & Programming Guidelines*) when preparing your solutions to these problems.

Exercising the algorithm you design using one or more example data set(s) removes any doubt from the grader that the algorithm is structurally correct and all computations are correct. When a problem supplies data you are expected to use it to demonstrate that your algorithm is correct.

All members of the collaboration group are expected to participate fully in solving collaborative problems, and peers will assess performance at the end of the assignment. Note, however, that each student is required to write up their solutions individually. Common solution descriptions from a collaboration group will not be accepted. Furthermore, to receive credit for a collaboration problem, each student in the collaboration group must actively and substantially contribute to the collaboration. This implies that no single student should post a complete solution to any problem at the beginning of the collaboration process.

You are permitted to use Internet resources while solving problems, however complete references must be provided. Please follow the Sheridan Libraries' citation guidance at "*Citing Other Things* - HOW DO YOU CITE AN INTERVIEW, A TWEET, OR A PUBLIC WEB PAGE?" and continue to the APA Academic Writer *Sample References*. Additional example citations are provided at the Purdue University, Purdue Online Writing Lab (OWL), College of Liberal Arts *Reference List: Electronic Sources*. You can also use training provided by Victoria University Library, Melbourne Australia on *Harvard Referencing: Internet/websites*

All written assignments are expected to be typed in a word processor, the preferred submission type is PDF allowing for ease of grading. When using equations, make use of the built-in equation editor for these tools. Try to avoid scanning hand-drawn figures, if needed please ensure they are readable. Only scans from flat bed scanners or printers will be accepted; all scans from phones or phone photos will be rejected. The grader will make the determination of "readability". If the assignments are not legible, they will be returned to the student with a grade of zero.

1. [30 points] (0-1 Integer Linear Programming)

   In a 0-1 linear integer program, the solution vector x must be members of $\{0, 1\}$ instead of real-valued x in linear programming. A 0-1 linear integer program is a specialization of a linear program. The linear programs in CLRS Chapter 29 are valid starting points for this question. Differences will be in specifying integer and $\{0, 1\}$ aspects of the equations. The text does not have detailed information on 0-1 linear integer programming. Consult Problems 34.5-2 page 1098 and 34.5-3 page 1098 for the text's spefication.

   If you wish to read more, A Tutorial on Integer Programming, by Gérard Cornuéjols, Michael A. Trick, and Matthew J. Saltzman or Chapter 9 Integer Programming, 15.053 Optimization Methods in Business Analytics, James B. Orli. [There are two WWW links in this paragraph.]

   Multi-object tracking within optical video is a common problem in machine learning. Tracking is extremely difficult in general as the number, sizes, and dynamics of objects can be large. Further, objects can by occluded by other objects and scenery, causing variable time gaps between one detection and the next.

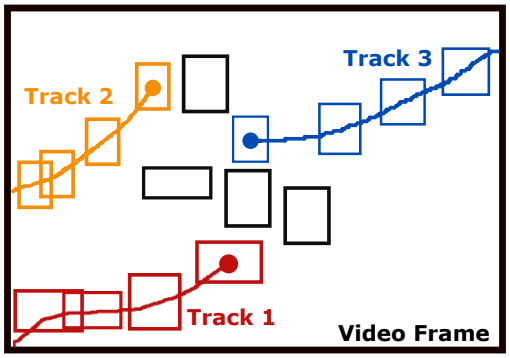

Figure 1: A tracking system that has three active tracks and has received four detections (black boxes) from the *Detector*. Note that the time intervals between detections assigned to tracks varies, as does the shape of their bounding boxes.

Tracking systems require several key components:

- **Detector**– A model that signals when an object of interest is apparently present, and returns a bounding box in pixel coordinates.
- **Tracker**– Software module that stores information regarding active tracks, and assigns object detections to tracks using a similarity score matrix produced by the *Evaluator*. Optimally, the *Tracker* assigns detections to tracks in a manner that maximizes the sum of the similarity scores from each track/detection pair.
- **Evaluator**– Given N active tracks and M detections, produces a score matrix $\boldsymbol{S} \in \mathbb{R}^{M \times N}$ such that element $s_{ij} \in \boldsymbol{S}$ indicates the similarity of detection i to track j under some similarity measure. Note that this definition leaves the actual definition of "track" fairly abstract. In the simplest case, for example, the *Evaluator* may define s_{ij} as the Euclidean distance between the center of the final detection within track j and the new detection i. More sophisticated models may uses multiple past detections within a track, or model the dynamics of an objects (e.g., with a Kalman filter).

In this problem, assume that you'd like to produce a novel tracking system. The *Detector* and *Evaluator* have already been created, and you must now describe the *0-1 integer linear program* that the *Tracker* will solve to assign detections to tracks. **Integer linear programs are constraint problems where the variables are restricted to be integers, and are typically NP-hard. In particular, 0-1 integer linear programs restrict the variables further by requiring them to be Boolean (i.e., zero or one).**

Assume that (a) you are provided with the score matrix $\boldsymbol{S}$, (b) detections can only be matched to a single track, and (c) that tracks cannot be assigned more than once.

(a) [5 points] What do the variables in this problem represent? How many are there? CLRS Pages 850-853 (first pages of Chapter 29) is a guide for answering this question.

(b) [10 points] Define the objective function for this 0-1 integer linear program. This is to be a mathematical expression as in CLRS Equation 29.11

(c) [15 points] Define the 0-1 integer linear program [objective function, all constraints], in standard form. This is to be a mathematical expression as in CLRS Equations (29.11)-(29.13) How many constraints are there in the 0-1 integer linear program?

2. [30 points] ***Collaborative Problem*** : In the course content, we explained how we can solve two-player, zero-sum games using linear programming. One of the games we described is called "Rock-Paper-Scissors." In this problem, we care going to examine this game more closely. Suppose we have the following "loss" matrix for Player 1 (i.e., we are showing how much Player 1 loses rather than gains, so reverse the sign):

$$A = \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix}.$$

(a) [6 points] What is the expected loss for Player 1 when Player 1 plays a mixed strategy $x = (x_1, x_2, x_3)$ and Player 2 plays a mixed strategy $y = (y_1, y_2, y_3)$?

(b) [8 points] Show that Player 1 can achieve a *negative* expected loss (i.e., and expected gain) if Player 2 plays any strategy other than $y = (y_1, y_2, y_3) = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$.

(c) [8 points] Show that $x = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$ and $y = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$ form a Nash equilibrium.

(d) [8 points] Let $x = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$ as in part (c). Is it possible for (x, y) to be a Nash equilibrium for some mixed strategy $y' \neq \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$? Explain.

(Homework #7 continues on the next page)

3. [40 points] ***Collaborative Problem*** : Consider the following problem. There is a set U of nodes, which we can think of users (e.g., these are locations that need to access a service, such as a web server). You would like to place servers at multiple locations. Suppose you are given a set S of possible sites that would be willing to act as locations for servers. For each site $s \in S$, there is a fee $f_s \geq 0$ for placing a server at that location. Your goal will be to approximately minimize the cost while providing the service to each of the customers.

   So far, this is very much like the Set Cover Problem: The places s are sets, the weight of a set is f_s, and we want to select a collection of sets that covers all users. But there is one additional complication. Users $u \in U$ can be served from multiple sites, but there is an associated cost d_{us} for serving user u from site s. As an example, one could think of d_{us} as a "distance" from the server. When the value d_{us} is very high, we do not want to serve user u from site s, and in general, the service cost d_{us} serves as an incentive to serve customers from "nearby" servers whenever possible.

   So here is the question, which we call the *Facility Location Problem*. Given the sets U and S with associated costs d and f, you need to select a subset $A \subseteq S$ (A is the set of sites in the Answer) at which to place the servers (at a cost of $\sum_{s \in A} f_s$) and assign each user u to the active server where it is cheapest to be served, $\min_{s \in A} d_{us}$. The goal is to minimize the overall cost

$$\sum_{s \in A} f_s + \sum_{u \in U} \min_{s \in A} d_{us}.$$

   (a) [20 points] Design a $\rho(n)$-approximate algorithm for this problem [pseudocode required].

   (b) [10 points] Prove that your algorithm is a $\rho(n)$-approximate algorithm for this problem. See CLRS Equation (35.1).

   (c) [10 points] What is the asymptotic runtime of your algorithm?