# ISE 435-535 Project 1

**Last updated 10/20/2023** (new/modified content in blue). The main changes from the previous version are updating the deadline, updating the deadline for questions, and adding clarifying details and tips for some of the tasks and deliverables.

**Problem Context:** You have joined forces with a start-up food services company seeking to establish operations in various U.S. cities. Their business plan entails setting up rapidly deployable "cloud kitchens" in vacant commercial real estate locations and delivering meals to service stations via large drones.

Your team has been hired to deploy your Python and IE skills to help guide the start-up company about how to best serve your chosen city. To this end, your team will engage in three main tasks: (I) gather, calculate, and store data related to the selected cloud kitchen locations and sample service stations (II) construct and solve an optimization model that determines what cloud kitchen will deliver to each service station so as to minimize the total distance, (III) analyze and visualize the solution results.

## Task I

- Select a team of 2-3 students (or contact me to assign you). Each team must choose a unique city on which to base its project (it is possible for teams to select different cities from the same state); selections are on a first-come, first-served basis. Try to select from well-known and highly populated metropolitan areas to simplify the data collection tasks. Specify your team's information in this Google Sheet.
- Select the cloud kitchen locations by gathering the addresses of 25 business buildings for lease in your chosen city; these locations must not be too close to one another (at least half a mile apart).
- Record the location indices and addresses in a file (in formats such as Excel or CSV), and subsequently load it from inside your Python program; avoid direct copy & pasting of the data. Then, obtain the longitude and latitude coordinates of the cloud kitchen locations using specialized Python libraries. If using the library "geopy" for this purpose, note that multiple accesses and extending the default timeout may be required to collect all coordinates.
- Plot the locations of the cloud kitchens onto a 2-dimensional map, with the help of matplotlib. Draw the rectangular plot region so that it contains all 25 data-points within its boundaries, plus 2.5 miles east of the easternmost point, 2.5 miles west of the westernmost point, 2.5 miles north of the northernmost point, and 2.5 miles south of the southernmost point. Perform the requisite calculations in Python.
- Next, since the locations of the service stations are not yet known, you will generate a sample by randomly selecting 50 points from the cloud kitchen map (assume that service stations can be set up even if there is a lake or other geographical obstacle). Set a random seed to allow for replication of the sample.

- Construct a Python table of the collected location data using the “tabulate” library. The table should contain the cloud kitchen and service station indices along the first column (in other words, after the table header, the first 25 rows of the table will contain the cloud kitchen data, and the next 50 rows will contain the service station data). The other columns should be “Street Address”, “Zip Code”, and “Coordinates”; the address should be a string, the zip code an integer, and location a tuple containing the latitude and longitude (floats). The street address and zip code of service stations can be omitted.

- Write a Python function called “distance” that calculates the Euclidean distance between every cloud kitchen $i \in I$ and every service location $j \in J$, given their longitudes and latitudes. Use miles as the unit of measurement; use (and document in the code) appropriate sources used to determine how to translate distances between coordinates into miles. Apply this function to calculate and store the distances in a matrix $[d_{ij}]_{i \in I, j \in J}$.

## Task II

- Formulate the assignment problem specified below using the PuLP library. Make sure to use the same names for the parameters and variables.

  **Parameters:**

$$\begin{aligned} i \in I &\text{ Cloud kitchens} \\ j \in J &\text{ Service stations} \\ d_{ij} &\text{ Distance between cloud kitchen } i \text{ and service station } j \end{aligned}$$

  **Variables:**

$$z_{ij} : \begin{cases} 1, & \text{if cloud kitchen } i \text{ delivers to service station } j \\ 0, & \text{otherwise.} \end{cases}$$

  **Objective function:**

$$\min \sum_{i \in I} \sum_{j \in J} d_{ij} z_{ij} \text{ (Minimize total distance traveled)}$$

  **Constraints:**

$$\sum_{i \in I} z_{ij} = 1 \text{ for } j \in J \text{ (Deliver to each service station)}$$

$$\sum_{j \in J} z_{ij} = 2 \text{ for } i \in I \text{ (Deliver from each cloud kitchen to two service centers)}$$

- Solve and store the resulting assignments resulting from the solution using a sparse data structure.

**Task III**

- Build and populate an "Origin and Destination (OD)" table with the following columns "Cloud Kitchen Index (Origin)," "Service Station Index (Destination)," and "Distance (miles)."; make sure to print only the pairs actually selected by the model (i.e., with $z_{ij} = 1$).
- Plot the assignment solution onto the map. You may use specialized Python libraries ("networkx") or do this manually.
- Create a frequency graph using for three different distance ranges (short, medium, long). The x-axis should have the distance ranges (e.g., $<3$ miles, 3-6 miles, $>6$ miles, and the y-axis should have the frequency values (e.g., % of Origin-Destination assignments within each range).

**Deliverables**

Submit the following as a single zipped folder, titled as "Project1_Team#_Submission.zip" (where # is your team number from the sign-up sheet):

1. Python program as a .py file (do not submit Python/Jupyter notebooks), titled as "Project1.py". The code should be well documented and include run instructions (if needed) at the very top.
2. Locations table as a .txt file, titled as "Locations.txt". The file should be generated with the tabulate library using the "simple" format; it should contain the indices and columns described under Task I.
3. Map of cloud kitchen and sampled service station locations as a .jpeg or other common extension, titled as "Locations.jpeg" (substitute file extension as needed). Include the information sources as comments in your Python code. All 75 locations (25 cloud kitchens and 50 service stations) can be plotted onto a blank background or overlaid onto the actual geographical map of the area — both options are acceptable.
4. Distances matrix $[d_{ij}]$ as a CSV file, titled as "Distances.csv". Include only the data; do not include row or column indices.
5. Instantiated formulation as a .mps file, titled as "AP.mps". This type file is generated with a command from the PuLP library after the model has been defined and the instance data loaded into it. It is recommend to import the full PuLP library rather than individual modules of it; otherwise, errors may occur).
6. Solution matrix as a .csr or other sparse matrix extension, titled as "Solution.csr". The extension ".npz" is acceptable.

7. OD table as a .txt file, titled as “OD.txt”. The file should be generated with the tabulate library using the “simple” format; it should contain the indices and columns described under Task III.
8. Frequency graph as a .jpeg or other common extension, titled as “Frequency.jpeg” (substitute file extension as needed).
9. Solution map as a .jpeg or other common extension, titled as “Solution.jpeg” (substitute file extension as needed).

All nine files should be included in the zipped folder (preferably saved with a .zip extension). Moreover, note that the .py file should generate deliverables #2-#8 when it is executed. Deliverable #9 does not have to be generated by the Python program.

**Due Date**: 10/26/2023 05:30pm (i.e., half an hour before class).

**Important Note**: We will not answer questions about this project at the “last minute”, herein defined as after 11/20/2022 at 11:59pm.