# Project 2

# Released: 19 Mar 19 (Tue, Week 11)

# Deadline: 2 Apr 19 (Tue, Week 13) 11:30pm

Server will start accepting submissions on 22 Mar 19 (Fri, Week 11)

# Always check “Announcements” at red.smu.edu.sg for errata and project-related information

## General Instructions:

* There are 2 questions in this project to be attempted in teams of three.
* Each member of the team is expected to contribute fairly to this project. A peer review may be conducted after the deadline to identify non-contributors. Individual team members may get different scores based on the peer review results.
* This project constitutes 15% of your final grade for this course.
* You need to install the numpy library on your laptop to run this project. Installation is straightforward: <https://www.youtube.com/watch?v=YWn8Wj69Dm4>. Do contact your TA or instructor for help if you have difficulty installing numpy.

**Requirements for q1 and q2.py:**

* Your **schedule1()** and **schedule2()**must each complete running within a specified time limit on the server (the actual limit will be posted at the server’s “announcements”).
* Your functions must return “correct” solutions – i.e. all orders must be accounted for, the answer returned must be syntactically correct. For Q2, the load capacities for each truck must not be breeched.
* The “quality score” and “time taken” on the scoreboard at red is merely a guide. After the deadline, your code may be executed using different data sets to derive your final score.

**Requirements for Written Report:**

* Explain your algorithm adopted to approach both questions. If possible, compute the time complexity of your algorithm and show your working. You will be given credit for clarity of explanation, novelty of approach and evidence of analytical thinking. Use diagrams and examples where relevant.
* Your report must be synchronized with the latest version of code submitted to red.
* Your report must not be longer than 2 pages (excluding the cover page, references and appendices, if any). Content beyond 2 pages may not be marked.
* Include your team ID (G?T??) and names of all members on the cover/first page.
* Plagiarism is an offence. If you have used any third-party libraries or code, or taken an idea from any source, you must acknowledge all sources in a reference page. Include URLs if relevant.

**Context:** A logistics company receives many orders and schedules the delivery. Each order includes the following information: order ID, weight (of the items in that order), and delivery location.The delivery trucks start and end at the same location. All the items to be delivered are housed at the starting point. The company has a limited number of trucks which is specified in each test case. The total number of trucks may be larger or smaller than the total number of orders. One truck can deliver more than 1 order at a time (but limited to the maximum load capacity in Question 2).

Here is the description of some given files:

* **locations\_1.csv**: this file shows the distances between any 2 locations. Do not assume that the distance from location X to location Y will be the same as the distance from Y to X. For simplicity, you may assume that the trucks travel at the same constant speed. So effectively, the total time taken by a truck to move between two locations is basically proportional to the distance between these two locations. The actual coordinates of each location are not relevant, and not available.
* **order\_1.csv**: This is an example of a list of orders to be fulfilled. The fields are Order ID, weight of items in that order, and the delivery location for that order. Each test case uses only 1 order CSV file. Remember that the number of orders is independent from the number of trucks: in some test cases, you may have many orders but fewer trucks or vice versa. It is also possible for multiple orders to have the same delivery location. You can assume that order IDs are unique in the CSV file. You are encouraged to come up with your own CSV files to test your solution. The weight field is only applicable for question 2.
* **load\_1.csv** (only applicable for question 2): This shows the maximum weight of items that each truck can carry (load capacity). When coming up with the schedule for each truck, your algorithm needs to ensure that the sum of the weight of all the items in the orders for a particular truck’s schedule does not exceed that truck’s load capacity. You can assume that the truck IDs in this CSV file are unique, and will always be in running order starting from 1 (i.e. 1, 2, 3… etc.). The number of trucks (or rows) in this CSV file will indicate the number of trucks in the fleet.
* **utility.py**. Contains functions used by **q1\_main.py** and **q2\_main.py**.

Your solutions to both questions should be able to work with variations of these CSV files that may not be provided.

**Question 1**. For Q1, you can assume that each truck has infinite load capacity (i.e. the weight of the items in the orders that it has to deliver does not matter). Your job is to come up with an algorithm to determine the schedule of each truck, so that (i) all orders are fulfilled, and (ii) you want to minimize the total time taken for all trucks to come back to the starting point. Since all trucks will start off at the same time, the total time taken for all trucks to come back will be the time taken by the slowest truck (or the truck travelling the longest distance based on its schedule). You want to minimize this time (or distance).

**To Write:**

Your algorithm should determine the delivery schedule of each truck which is numbered from **1** to **num\_trucks**. You are required to fill up the body of this Python function in **q1.py**:

**def schedule1(locations, start, num\_trucks, orders):**

Where:

* **locations**: is a list of rows read from **locations\_1.csv** or a similar file. Each item in the list represents the first location, the second location, and the distance required to be travelled when moving from the first to the second location. **locations** may look like this: **["USA,CAN,731", "USA,BHM,1623", "USA,CUB,1813"...]**. You are expected to process **locations** in any way you want in order to utilize it in your algorithm.
* **num\_trucks**: The number of trucks in your fleet. You will need to come up with a schedule for each truck. So if **num\_trucks** is 3, your algorithm should return 3 schedules, one for each truck. You can assume that **num\_trucks** is an integer that is 1 or bigger.
* **start**: this is the starting location for all trucks. This is also where all the items for delivery are housed.
* **orders**: this is a list of orders that your fleet of trucks must fulfil. Orders are read from an **order\_x.csv** file. Each item in the list is a tuple that shows the ID, weight and delivery location of each order. Here is an example of the list of orders read from **order\_1.csv**:

**[(1, 100, 'RUS'), (2, 150, 'CAN'), (3, 250, 'SIN'), (4, 300, 'KOR'), (5, 200, 'CHN'), (6, 250, 'MEX'), (7, 350, 'AUS'), (8, 270, 'GMY'), (9, 180, 'FRN'), (10, 230, 'SPN')]**

You can assume that there will not be an order that has the delivery location set to the starting location since this scenario does not make sense. (When creating your own test cases, do avoid such a situation by checking that the value of **START** in your **main.py** does not appear in **order\_x.csv** that you will be using).

This function should return:

* A list of schedules for each truck in the form of a 2D list.

This is how a schedule (for one truck) looks like: **[(1, 100, 'RUS'), (5, 200, 'CHN'), (9, 180, 'FRN')]**. This means that the truck allocated this schedule should first move from START to RUS to fulfil order #1, then to CHN to fulfil order #5, then to FRN to fulfill order #9, and lastly, back to START.

This function should return a list of schedules, one for each truck. So, if **num\_trucks** is 4, and there are 10 orders, a valid 2D list that can be returned by **schedule1()** will be:

**[[(1, 100, 'RUS'), (5, 200, 'CHN'), (9, 180, 'FRN')],**

**[(2, 150, 'CAN'), (6, 250, 'MEX'), (10, 230, 'SPN')],**

**[(3, 250, 'SIN'), (7, 350, 'AUS')],**

**[(4, 300, 'KOR'), (8, 270, 'GMY')]]**

**Question 2**. For this question, each truck in your fleet has a maximum load capacity, and can transport items with limited total weights (see **load\_1.csv**). Unlike Q1 in which time is important, for Q2, fuel is important. The company aims to reduce the fuel consumption of the fleet. Implement your algorithm to derive the schedules for each truck in your fleet so that (i) all orders are fulfilled, (ii) the weight of all orders to be delivered by each truck must not exceed its load capacity, and (iii) you want to minimize the total distance travelled by all trucks in the fleet. Trucks are not allowed to return to the starting location to “reload”.

**To Write:**

Like Q1, your algorithm should determine the schedules of each truck whose weight limit is listed in **capacities**. For example: the maximum load capacity of the first truck is the first value in **capacities**. The number of elements (length) of **capacities** is also the number of trucks in your fleet. You are required to fill up the body of this Python function in **q2.py**:

**def schedule2(locations, start, capacities, orders):**

where:

* **locations**: same as Q1
* **start**: same as Q1
* **capacities**: this is a list of integers representing the maximum load capacity of each truck in the fleet. (e.g.: **[400, 10, 1000, 1500, 600]** represents the load capacity of 5 trucks.)
* **orders**: same as Q1.

You can assume that there will always be at least one valid solution for the combination of **orders** and **capacities** passed into your function. Two examples of situations with no solutions are: (i) there exists an order in **orders** that has a weight that is heavier than the highest truck capacity in **capacities**. (ii) the sum of all trucks’ capacities is smaller than the sum of all orders’ weights.

This function should return:

* Same as Q1: a list of schedules for each truck in the form of a 2D list.
* Remember that you should strive for a “correct” solution. As long as there is a valid solution for the combination of **orders** and **capacities** passed into your function, your function should return a valid solution.

**Scoring**

This assignment is worth 15% of your final IS103 grade:

* Algorithm: Q1 (6%)
* Algorithm: Q2 (6%)
* Written report (3%)

Your algorithm will be scored in 3 ways:

1. Correctness: it works as required and returns correct solutions for different test cases. Each function must complete running within a fixed amount of time on the server. This time limit given will be posted at red. Additional test cases will be executed on the server after your deadline. Correctness is VERY important, because speed and quality will only be considered if your solution is correct.
2. Speed: the time taken for your function to complete. You may be given credit for algorithms that have a good time complexity (and hence takes lesser time to complete for a large data set). The “time taken” on the scoreboard at red is merely a guide; after the deadline, your code may be executed using different data sets to derive your time taken.
3. Quality: the score for your algorithm. For Q1, the score will be the maximum time taken (by the slowest truck). For Q2, the score will be the total distance travelled by all trucks. For both questions, you want those scores to be as low as possible. Similarly, the “quality” score on the scoreboard at red is merely a guide. **For both questions, more marks will be given to quality than to speed.**

For all questions, an attempt is better than no submission. Attempts submitted to eLearn (together with your written report) may be manually marked if they do not show up as “correct” at red.

**To Submit:**

* Always refer to the “Announcements” at http://red.smu.edu.sg; clarifications, additional test cases and bug reports will be posted there.
* Submit to red:
  + You need to submit **q1.py** and **q2.py,** to red for automatic marking.
  + You can submit your solutions to red as many times as you wish, but the final submission on the deadline will be taken as your final submission.
  + **Do check both scoreboards after your last submission to check that your team’s name is on the scoreboards** (your team’s name will appear if you have submitted a correct answer).
* Submit to eLearn (Assignments):
  + Written report (Word or PDF)
  + Identical copies of your final versions of **q1.py** and **q2.py.**
* Late submissions will be penalized:
  + <1 hour late: penalty of 10%, <24 hours late: penalty of 25%
  + Submissions later than 24 hours will not be marked (you will get zero for that question).
  + red will be closed for submission at the deadline. If you are submitting anything late, you will have to contact Mok (telegram: @Mokkie) to enable submission at the server.

~End

*Acknowledgement: These assignment problems are designed by the senior TAs: Tran Huy Vu, Hoang Van Duc Thong and Nguyen Hoang Minh. Mok has adapted the questions, inserted examples and modified some of the code.*