Q1

For this question, your task is to come up with an algorithm that outputs the delivery schedule of each truck. The schedules must ensure that all orders are delivered. Your algorithm should strive to minimize the time required to deliver all the orders. Since all trucks start off from the same location (0, 0) and travel at the same speed, the objective here then is to minimize the distance of the truck which travels the furthest distance based on its schedule. All distances are calculated as the Euclidean distance between two locations (i.e. the geometric between two points on the coordinate system).

Requirement

You are required to fill up the body of this function in **q1.py**:

schedule_q1(orders, number_trucks)

where:

- **orders** is a list of items describing the order ID and delivery location of each order. Each item represents the order ID, its x-coordinate, its y-coordinate. This list may look like this: **[[O001, 5.5, 3.4], [O002, -10.0, -8.9], …]**
- **number_trucks** is a positive integer that denotes the number of available trucks.

The function should return a list of schedules in a list of items in which each item describes the schedule of each truck. The list contains the same number of items as the number of trucks used in your proposed schedule. This number may be equal to or less than the total number of available trucks (you need not use all the available trucks).

For example, assuming that you have 4 delivery trucks, and there are 10 orders to be fulfilled, your function may return this:

[[O002, O005, O003, O001], [O008, O004, O006, O010], [O007, O009]]

This particular solution deploys only 3 of the 4 available trucks. The first truck's schedule is **[O002, O005, O003, O001]**. This is to be interpreted as:

1) Truck starts from the company (0, 0) and goes to the delivery location of the order '**O002**' to fulfil the delivery.
2) It goes to the delivery location of the order '**O005**' to fulfil the delivery
3) It goes to the delivery location of the order '**O003**' to fulfil the delivery
4) It goes to the delivery location of the order '**O001**' to fulfil the delivery 5) Truck then returns to the company (0, 0).

In this case, the total mileage of the first truck would be the sum of Euclidian distances between:

1) The locations of the company and '**O002**'
2) The locations of '**O002**' and '**O005**'
3) The locations of '**O005**' and '**O003**'
4) The locations of '**O003**' and '**O001**'
5) The locations of '**O001**' and the company.

The total mileage of each of the other two trucks can be calculated in the same way. The mileage of the truck which travels the furthest will determine the time taken for all orders to be fulfilled. Remember that your algorithm aims to minimize this time. The number of items in the returned list should be fewer or equal to **number_trucks.**