**HASH CODE ARCHIVE – DRONE DELIVERY**

**Introduction**

The Internet has profoundly changed the way we buy things, but the online shopping of today is likely not the end of that change; after each purchase we still need to wait multiple days for physical goods to be carried to our doorstep. This is where drones come in ­ autonomous, electric vehicles delivering online purchases. Flying, so never stuck in traffic.

This is a synthetic code challenge to sharpen the programming skills. This problem was first released during the 2016 qualification round of Google's annual coding competition. Hase code released it as a Playground Code Competition to help and sharpen the skills. Hash Code is a team programming competition, organized by Google, for students and professionals around the world.

**Products**

There are a number of product types available for order. Each product type has one or more product items available in warehouses. Each product type has a fixed product weight, identical for all product items. Every product weight is guaranteed to be smaller or equal to the maximum payload that a drone can carry.

**Warehouses**

Product items are stored in several warehouses. Each warehouse is located in one particular cell of the grid, different for each warehouse. Each warehouse initially stocks a known number of product items of each product type. No new product items beyond the initial availability will be stocked in the warehouses during the simulation, but the drones can transport product items between the warehouses. Any warehouse does not necessarily need to have every product type available.

**Orders**

Each order specifies the product items purchased by the customer. The product items in an order can be of one or multiple product types and can contain multiple product items of the same product type.

Each order specifies the cell in the grid where the product items have to be delivered. It is possible to have multiple orders with the same delivery cell. No order has the delivery cell that is a location of a warehouse.

The order is considered fulfilled when all of the ordered product items are delivered. Individual product items can be delivered in multiple steps, in any order. It is valid to deliver the individual product items of an order using multiple drones, including using different drones at the very same time.

**Drones**

Drones transport product items from warehouses to customers and between the warehouses. The drones always use the shortest path to fly from one cell in the grid to another.

**● Load:** Moves the specified number of items of the specified product type from a warehouse to the drone’s inventory. If the drone isn’t at the warehouse it will fly there using the shortest path before loading the product items. The requested number of items of the specified product type must be available in the warehouse. The total weight of the items in the drone’s inventory after the load cannot be bigger than the drone’s maximum load.

**● Deliver:** Delivers the specified number of items of the specified product type to a customer. If the drone isn’t at the destination it will fly there using the shortest path before delivering the product items. The drone must have the requested number of items of the specified product type in its inventory. Each drone can also be given the following advanced commands. These commands are not necessary to solve the problem, but you can use them to further improve your solution.

**● Unload:** Moves the specified number of items of the specified product type from drone’s inventory to a warehouse. If the drone isn’t at the warehouse it will fly there using the shortest path before unloading the product items. The drone must have the requested number of items of the specified product type in its inventory.

**● Wait:** Waits the specified number of turns in the drone's current location.

**Simulation**

The simulation proceeds in T turns, from 0 to T − 1 . A drone executes the commands issued to it in the order in which they are specified, one by one. The first command issued to the drone starts in turn 0.

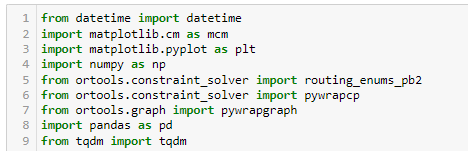
**DATASET**

The dataset I have used is busy.in and submission.csv.

The given below are the available Jupyter notebooks

* 1st Place solution.
* 2nd Place solution
* 3rd Place solution
* Very Greedy solution

**STEP 1: Importing the necessary Libraries**



Python Datetime module supplies classes to work with date and time. These classes provide a number of functions to deal with dates, times and time intervals. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps.

Datetime– Its a combination of date and time along with the attributes year, month, day, hour, minute, second, microsecond.

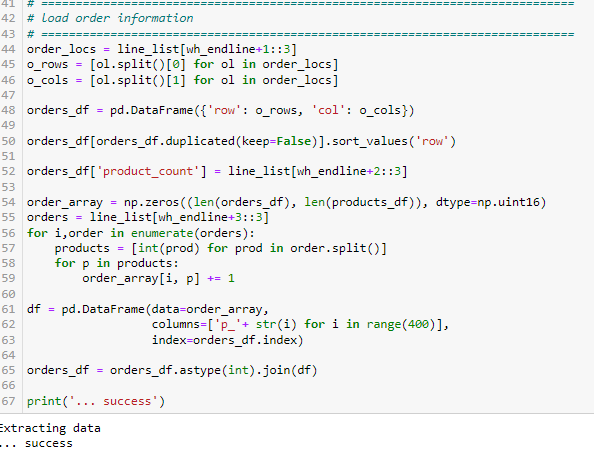
# MCM Matrix Chain Multiplication

Given the dimension of a sequence of matrices in an array **arr[]**, where the dimension of the **ith** matrix is **(arr[i-1] \* arr[i])**, the task is to find the most efficient way to multiply these matrices together such that the total number of element multiplications is minimum.

Tqdm

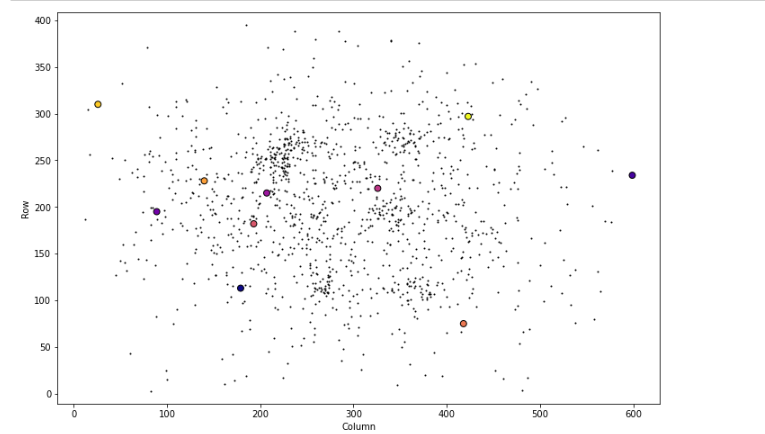
Instantly make your loops show a smart progress meter - just wrap any iterable with tqdm(iterable)

**STEP 2: Extract data**

****

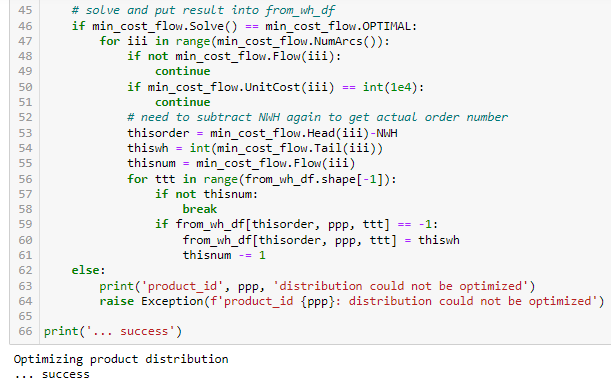
**STEP 3: Plotting warehouse and orders**

Simply plotting all warehouses (colors) and all orders (black).

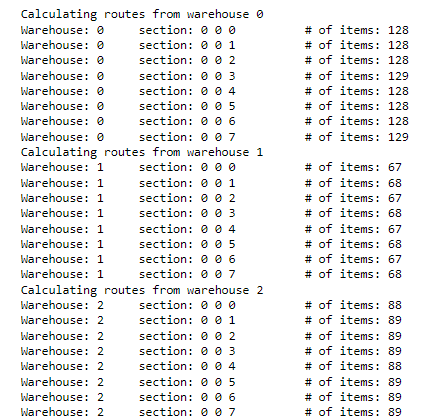
****

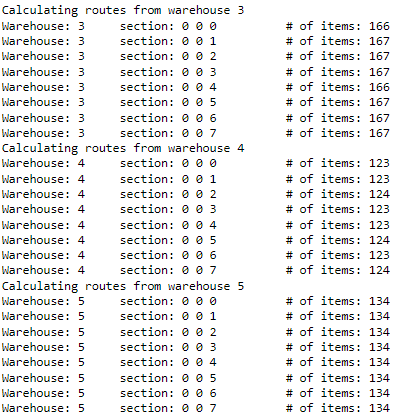
**STEP 4: Optimize Product distribution**

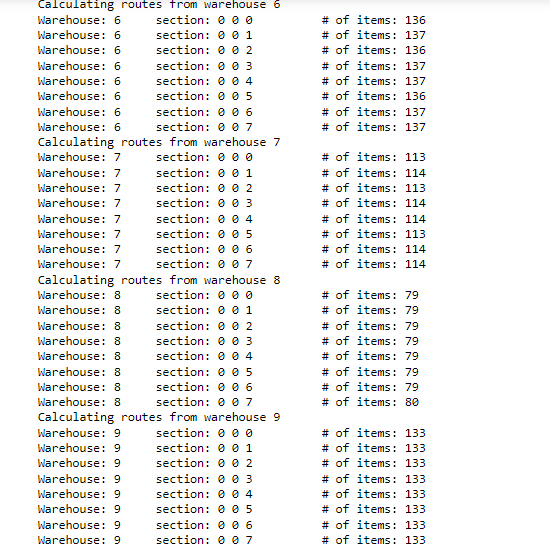
 From this information we can figure out the optimal way of distributing this product to all customers while minimizing the total distance which has to be covered to satisfy all needs.

****

**STEP 5: Create Delivery routes**

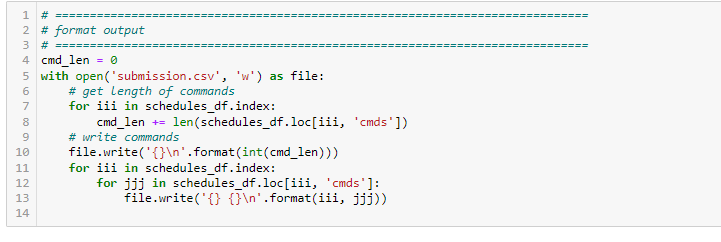
****

****

****

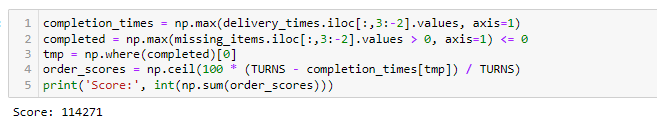
**STEP 6: Create submission file**

simply iterate through the schedules created in the previous step and write the submission file.

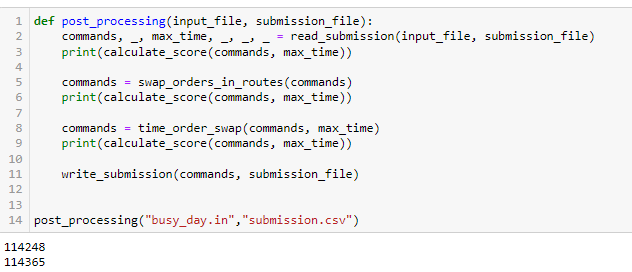
****

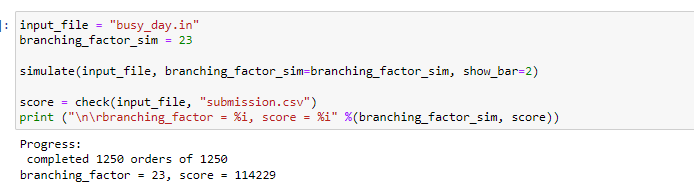
**STEP 7: Calculate Score**

calculate the final score according to the equation given in the instruction file.

****

****

****

****