The assignments will be delivered through email. Once the email is sent, you expect to receive the assignments as early as possible. After you finish the assignment, please send the solutions to the email you received from

There are two coding test assignments. One is to recognize your programming style and the other is to understand your asynchronous + concurrent programming knowledge

We expect you to use either Python 3.x or Javascript(Node.js). It is possible to refer to the Internet, to write an external library.

1.General Coding

The problem is compute the best way an order can be shipped (called shipments) given inventory across a set of warehouses (called inventory distribution).

Your task is to implement InventoryAllocator class to produce the cheapest shipment.

The first input will be an order: a map of items that are being ordered and how many of them are ordered. For example an order of apples, bananas and oranges of 5 units each will be

```
{ apple: 5, banana: 5, orange: 5 }
```

The second input will be a list of object with warehouse name and inventory amounts (inventory distribution) for these items. For example the inventory across two warehouses called owd and dm for apples, bananas and oranges could look like

```
[ { name: owd, inventory: { apple: 5, orange: 10 } }, { name: dm:, inventory:
{ banana: 5, orange: 10 } } ]
```

You can assume that the list of warehouses is pre-sorted based on cost. The first warehouse will be less expensive to ship from than the second warehouse.

Please write unit tests with your code, a few are mentioned below, but these are not comprehensive.

Examples

*Happy Case, exact inventory match!**

Input: `{ apple: 1 }, [{ name: owd, inventory: { apple: 1 } }]`
Output: `[{ owd: { apple: 1 } }]`
Not enough inventory -> no allocations!

Input: `{ apple: 1 }, [{ name: owd, inventory: { apple: 0 } }]`
Output: `[]`
Should split an item across warehouses if that is the only way to completely ship an item:

Input: `{ apple: 10 }, [{ name: owd, inventory: { apple: 5 } }, { name: dm, inventory: { apple: 5 }}]`
Output: `[{ dm: { apple: 5 }}, { owd: { apple: 5 } }]`

2. Asynchronous and Concurrent Programming Questions

This is a task to show your capabilities regarding an asynchronous programming and performance solving.

You are making simple chatbot (servers & clients)

If you choose Python, you can start from provided scripts

- chatserver.py
- index.html
- requirements.txt

After installing the package described in requirements.txt, run the following to run the server:

python chatserver.py
You can test it by connecting to http: // localhost: 8000.

Multiserver Broadcast Implementation

The first assignment is to modify chatting application so the broadcast application is running well even when it is running in multi-process. Currently, chatting application works only when it running on single process because it is broadcasting as socket information is in memory.

Once implemented, we expect to run multiprocess with the following command:

gunicorn -k "geventwebsocket.gunicorn.workers.GeventWebSocketWorker" -w 4 -- bind 0: 8000 chatserver

You can test it by connecting to http: // localhost: 8000.

We strongly recommend using rabbitmq or redis as the backend if possible. If you are familiar with other asynchronous technologies such as asyncio

than gevent, you can create a new chatserver.py. Of course, you also need to write your own gunicorn command.

Measuring Chat Server Performance

Write a test client that measures the performance of the Chatting server you created above. Please uses Python and you can use any asynchronous technology you are familiar with. You should measure the items below:

- each client send a message every 3 seconds
- It should take less than 1 second to receive any messages from all client

You can write code that tests how many clients can be attached under the conditions above. Please do not count network problems, or server / client load sharing issues. It should work well on your own computer by running server and the client on your machine.

If you are using Node.js, you need to implement from the scratch

The way of Node.js async & concurrent programming can be different compared to Python. Please implement in Node.js way to solve these issues

Multiserver Broadcast Implementation (description above)

Measuring Chat Server Performance (description above)

Again we strongly recommend using rabbitmq or redis if required.

Suggested technics but not required: Worker Threads, Cluster Modules

What are we looking for

We'll evaluate your code via the following guidelines in no particular order:

1. **Readability**: naming, spacing, consistency
2. **Correctness**: is the solution correct and does it solve the problem
3. **Test Code Quality**: Is the test code comperehensive and covering all cases.
4. **Tool/Language mastery**: is the code using up to date syntax and techniques.

Please email (danny.park@travelflan.com) if you have any questions. We will be able to respond as soon as possible.

Thank you. [1]