Getir Algorithm Challenge

March 2021

Problem Description

At this challenge, you are given a snapshot of GetirSu (Getir's water delivery service). This is a daily example of GetirSu: vehicles are roaming around the neighborhood with a specific stock of products at a given time, delivering orders one by one.

There are n vehicles roaming around the city and m orders waiting for the delivery. For the sake of simplicity, you can assume that vehicles carry infinite amount of stock and delivery process itself takes zero time (handing over the carboy, climbing the ladders, parking the vehicle etc).

Given the input data, find the routes that minimizes the total delivery duration. (Note that vehicles can end their journey in any location, they do not need to return back to any depot since they carry infinite stock already.)

You have two options in this challenge, pick only one of them:

1. Solve the problem by looping over every possible vehicle-route matchings - aka brute force - , using Python. In this option, You are expected **not to use any libraries or third-party tools that model or solve the problem.** If you want a harder challenge you can implement a heuristic, meta-heuristic or genetic algorithm. Whatever algorithm you use, you must implement it completely yourself.
2. Develop an HTTP microservice that gets the input data with a POST request and outputs the optimal routes back in the response. Note that in this option, you do not need to solve the problem by yourself. You can use any 3rd party tool of your choice, some examples : Google OR-tools, VROOM.

In both options, you can go further and consider vehicles have a limited amount of stock and predefined service durations. These options are not mandatory but might make a positive impact.

Input Data

You are given a json file that contains vehicles, orders and duration matrix.

1. Vehicle object in vehicles array
   - id : Vehicle id
   - start_index : Index of the vehicle's starting location (see matrix below)
   - capacity : Initial carboy capacity of the vehicle (Optional)
2. Job object in jobs array
   - id : Order id
   - location_index : Index of the order location (see matrix below)
   - delivery : The amount of carboy that will be delivered in this job (Optional)
   - service : Service duration, in seconds (Optional)
3. Matrix (list of list of integers)

   List of each row in the duration matrix. Element (i, j) indicates the amount of seconds it takes to travel from location index i to location index j

Output Data

You need to submit a route for each vehicle, **note that some routes might be empty.** This is a JSON object with the fields indicated below.

- total_delivery_duration : Total delivery duration of all vehicles to all jobs
- routes : Vehicle - Routes mapping objects. Maps vehicle ids to routes as an array of job ids and defines delivery duration for the route.

Output data must follow the definition above.

Example :

```
{
  "total_delivery_duration": 4891,
  "routes": {
    "1": {
      "jobs": [
        "1",
        "4",
        "2"
      ],
      "delivery_duration": 3047
    },
    "2": {
      "jobs": [
        "3",
        "5",
        "6"
      ],
      "delivery_duration": 844
    },
    "3": {
      "jobs": [],
      "delivery_duration": 0
    }
  }
}
```


Evaluation Criteria

For both problems :

- 50% algorithm, implementation details, performances etc.
- 50% design, architecture, structure, code quality etc.

Notes

- Please **do not** use the “Getir” keyword at any point of your case.
- Please keep a clear commit history.
- Simplicity is the key.