The Great Routing Problem

Context

Lob processes millions of print and mail orders a day with the help of our print partners across the globe. However, not all print partners are made equal; some print partners can only process checks, some can only handle postcards, some can only handle a specific subset of postcards, etc. The distance between print partners and the destination of the mail piece also affects how long it takes for the recipient to get their mail, which affects our deliverability speed. All of these factors are taken into account when deciding how to efficiently send our mail.

Input

The input will be two JSON files containing information about the partners and orders. Here's an example of the input files (the comment will not be included in the input file):

```
// partners.json
[
    {
        "id": "partner_1",
        "resource": "checks",
        "type": [
            "cheap",
            "fancy"
        ],
        "capacity": 5,
        "address": {
            "latitude": 42.651814,
            "longitude": -71.219284
        }
    },
    ...
]
```

```
// orders.json
[
    {
        "id": "order_1",
        "resource": "postcards",
        "type": "cheap",
        "address": {
            "latitude": 33.046384,
            "longitude": -117.25313
        },
        "deliverability": "deliverable"
    },
    ...
]
```

The Challenge

Given a list of mail piece orders and a list of print partners, route each order to the correct partner, optimizing for deliverability speed.

For an order to be routable to a partner, the following must be true:

- An order must have a deliverable address (`deliverability = 'deliverable'`). If the order is not deliverable, then do not process the order, and do not include it in your output file.
- The partner must have the capability to produce the order. To be capable, a partner must support the order's `resource` (what kind of mail it is, e.g.: postcards, letters, checks) and `type` (a sub-category, e.g.: cheap, standard, fancy).
- The partner must have remaining order capacity. Given a partner's max capacity of orders per day (`capacity`), they must not receive more than that number of orders. For the sake of this problem, we can assume that it will never be the case that *all* partners reach capacity.

If multiple partners can receive an order, choose the partner that's the closest to the destination address, in terms of longitude and latitude. For this problem, we'll assume that the shorter the distance between partner and order destination, the quicker the order will be delivered.

- We used Lob's Address Verification API to get an address's deliverability, latitude, and longitude. We've given you the relevant information in your input JSON. If you're interested in learning more about what the Address Verification API does, check out the docs here.
- Use the Euclidean distance formula to calculate distances between partners and delivery addresses. For the purposes of this problem, we will pretend that the Earth is flat.

You must route orders in the order that they are provided in the input. All orders will have an id of the format `order_X`, where `X` is a positive integer. `order_1` should be processed first, then `order_2`, then `order_3`, etc.

Output

Your output should be a text file in this format:

```
partner_a
order_2
order_3
partner_b
order_5
partner_c
order_1
order_4
partner_e
partner_f
```

The orders for each partner should be printed in the order that they were routed. We've provided an `answers.txt` file, which is the correct and expected output for the provided `orders.json` file. You can verify the correctness of your program by comparing your own `output.txt` file to the `answers.txt` file.

What's Included

- `partners.json`
    - the list of partners and their capabilities that you will be using for all of your orders files.
- `orders.json` and `answers.txt`
    - a list of orders and its corresponding answer file (used to test your program)
- `orders1.json` through `orders4.json`
    - you will be submitting your answers to these orders files

Submission

Here's what you need to submit:

- A README file: please provide a README file with instructions on how to run your code. Include as much detail as possible, including the language and version used, how to install dependencies/packages used, how to run your script/program, etc. We are looking for strong written communication skills at this step, so don't skimp on this!
- Your source code. Depending on how you tackle this problem, this can be a single file, a directory with multiple files, etc. It is important to include your source code and not just an executable; we want to see the code you've written.
- For each `ordersX.json` file (where X is an integer from 1 to 4), generate an `outputX.txt` file with your answers for the corresponding orders file, and include them in your submission.
- OPTIONAL: You are free (and encouraged) to create your own test input files for testing purposes. Please include any test files you use to test your code.