SET UP AND DEPLOYMENT INSTRUCTIONS

1) Install anaconda on ubuntu (https://www.digitalocean.com/community/tutorials/how-to-install-anaconda-on-ubuntu-18-04-quickstart)

2) Create a python 3.5 virtual environment using conda (https://uoa-eresearch.github.io/eresearch-cookbook/recipe/2014/11/20/conda/) (name of environment is python3.5_env in our example)
e.g. conda create -n python3.5_env python=3.5 anaconda pip

3) Activate that environment ($source activate python3.5_env)

4) pip install simplejson (and equivalently anything else that produces the error “Module not found”)

5) Do chmod +x in all the .py files that you want to run (e.g. $chmod +x server.py)

6) Type ./server.py in one console and ./post.py in another console to run a test case (you have to activate the virtual environment –command in step 3-- for each console).

The repo contains 5 files.

server.py:
The server.py listens for post requests with a body that contains the information needed to create the problem and solve it (in json format) and responds with 200 and the solution if we have correct input. If the input is incorrect we return a 404 or 500, depending on the situation with an error message.

The whole process is surrounded with a try catch so that even if an edge case hasn't been dealt with, the server won't break. It will print a stacktrace to get the necessary information and send a 500 status back.

post.py:
The post.py just does a post request and prints the response status and message. If the status is 200, it prints the body as well.

utilities.py:
Contains useful functions that actually create the map and solve the problem. (Could be renamed to hoover_solver.py)

generate_test_case.py:
A simple algorithm to create a random map with patches, starting point and moves for debugging help.

sampleRun.jpg:
An image showing the default testcase (left side) and our mapping and symmetric solution (right side)

NOTES:

Although the problem suggests that the starting point, i.e. (0,0) is in the bottom left and the end is at the top right, we can see that the problem is symmetric in the xx' axon and therefore we can solve it by having the start of the array at the top left and the end at the bottom right, only if we have a

command that tells us to move up, we will move down and vice versa. (The symmetry is n xx' therefore we don't need to do this inversion for the yy', i.e. left-right movements) (See sampleRun.jpg for help)

SUGGESTIONS:

We could expand generate_test_case.py to not generate only one test case but multiple ones and save them in a gzip-ed collection of jsons that we could then iteratively read in the client (post.py) and read the responses from the server.

We could also do some automation testing with these autogenerated methods, given that we had the “golden truth” function to assert on. Or we could grab an autogenarated map and scenario and solve it on our own and do a manual test based on that solution. Unfortunately my testing experience is mostly in Java with JUnit which would be the language I would use if I didn't need to set up a quick easily explained and deployable mini http server withing a day or two.

More comments are inside the code. A good place to start reading it is the server.py