Derek Choi (derekchoi@college.harvard.edu) Ramsey Fahs (cfahs@college.harvard.edu)
Alice Tang (alicetang@college.harvard.edu) Jessica Zhu (jessicazhu@college.harvard.edu)

CS51 Project, Task 3: Final Specification
MOO-pQuest

Signatures/Interfaces

1) Fetch data
    a) Fetch crime data from http://thomaslevine.com/!/nyc-crime-map/ (originally from http://maps.nyc.gov/crime/). Parse into Python.
        i) `crimedata.geojson → parsedata.py → crimedict.txt` (unreadable because in pickle format)
        ii) This dictionary will not decide weights yet so that we can still manipulate the crimes' properties later.
    b) Fetch map data from https://www.openstreetmap.org/.
2) Combine crime data and OSM data.
    a) Parse OSM file and add crime tags to nodes: https://docs.python.org/2/library/xml.etree.elementtree.html
    b) Use node tags to identify crimes in ways.
    c) `data.osm → crimedict.txt → insertcrime.py → nodetoway.py`
3) Integrate crime data in Pyroute algorithm.
    a) Pyroute: https://wiki.openstreetmap.org/wiki/Pyroute
    b) modify `pyroute/weights.py`, potentially `pyroute/loadOsm.py`
4) Optimize user interface
    a) GUI implementation: install pycairo, `gui.py`
    b) Cool features

the flow of this processing is changed slightly, and the entire processing segment was consolidated into one file (process.py)

this ended up being much less GUI-based, but still had a lot of user input

Abstraction Barrier

- A lot of abstraction is baked into pyroute's algorithm now.
- All values and functions will be hidden from the client. They will only input their location and their destination.
- Even for the final OSM data inputted into Pyroute, the weights have not been determined. This will allow the user to make specifications about safety-time parameters and modify the weights in Pyroute rather than through generating a new OSM file.

Client Restrictions

- The inputs will have the same format as pyroute takes because we are just changing the backend of pyroute, not the front end.
- For now, this means node ids as inputs. If the GUI starts working, we will have a much nicer interface.

Modules/Actual Code

See our zip file for our work so far. Note that `pyroute` has not been modified at all, `crimedata.geojson` and `data.osm` are unmodified data from the Internet, and everything else is our own creation.

Functional code includes `parsedata.py` and its inputs and outputs (`crimedict.txt` which works as a dictionary in python) .

Timeline

By Friday April 24th

- All core features will be fully functional.
  - We will have the crime data in a reasonable Python format.
  - We will have the OSM NYC data based on the boundaries of the crime data.
  - We will have a modified OSM data file with crime tags.
  - We will have modified the hard-coded weights in Pyroute to align with our OSM file.
- The command line Pyroute will output a route with our (hard-coded for now) safety parameters.

ended up not using all of NYC because of runtime

By Friday May 1st

- The GUI will be fully functional. As of now we do not know what this will look like because we have not gotten it to work.
- Extra features implemented:
  - Allow user manipulation of crime type.
  - Allow user specification of safety and time parameters.

If we're beyond awesome:

also implemented other features, such as address API

implemented distance

- Allow user specification of maximum time.
- Implement crime parameters based on radius rather than point.
- Allow users to compare multiple routes.

If we're good enough to take over Google:

- Cut down runtime.
- Modify GUI to reflect crime areas.

Progress Report

`parsedata.py`, `crimedict.txt`,and `insertcrime.py` are evidence of code we generated (the first two) and skeleton laid out (`insertcrime.py`). Again, note that `crimedict.txt` is in pickle format.

Version Control

Our project currently lives on a box in nitrous.io where we can all collaborate. It is synced to https://github.com/mengyazhu96/MOO-pQuest. We also have a copy on one of our local machines, also linked to git.