Exercise Sheet 1

Funke
FMI/ALG
Universität Stuttgart

Guidelines

Programming Language: C++.
Usage of Libraries: In principal, you are free to use any libraries in your project. But it must be apparent in your code that you did the tasks yourself. To give some examples: It is fine to use a library for parsing and reading PBF files. It is not fine to use a library that implements a Geocoder. The reason for this is that the former is only a part and implementation detail of the task while the latter is the core of the task. Please ask on the forum if in doubt.
Doing Tasks: The tasks are meant to be completed as one incrementally growing project. There will be three exercise sheets over the course of the semester. Each exercise sheet will have mandatory and optional exercises. The mandatory exercise give a baseline of what we except at the very least, while optional exercises extend the tasks for more functions while also offering a greater challenge. Both the mandatory and the optional tasks will be graded. To receive a good grade, optional tasks are expected.
Performance: Pay attention to the performance of your algorithms and data structures. This will be part of the grade, in the query stage as well as in preprocessing.
Submissions: There will be a code submissions (via Ilias) for each exercise sheet. In the code submissions you have to provide the code of your project as well as instruction how to compile and run the project. Also include which dependencies need to be installed and how to install them. The instructions should work on Ubuntu 22.04.

The submissions **must** be on time. Failure to do so will result in an exclusion of the course.

After the last submission there will be meetings for presentations. In these you should present your project in 10 to 15 minutes, focusing on the optional tasks you did and how you solved them. A short demo of the project as well as any other interesting challenge you solved on the way is always appreciated in the presentations. Be prepared for questions about your project.
Attendance: Attendance will be mandatory throughout the semester. Unexcused absence may result in a deduction from the grade. At each meeting each participant reports on their progress on the project, as well as planning the next steps.
Grading: Grading will happen at the end of the semester. Part of your grade will be performance of your code, quality of the returned results, as well as your presentation.

Overview

Goal: The goal of this lab course is to build a Geocoder as well as reverse Geocoder. A Geocoder takes as Input a string (e.g. "Aalen Hauptstraße 10") and returns objects (e.g. houses, streets, cities,...) on a map, which match the query as well as possible. A reverse Geocoder does the opposite and matches a location on a map to an object.
Sheet 1: (Submission on the 16. November) On the first exercise sheet you will familiarize yourself with geodata from OpenStreetMaps. From a file your task is to extract the relevant data and store them in a suitable format. Then you will preprocess the data for later usage.
Sheet 2: (Submission around late December) The task is to develop a Point in Polygon Test as well as the reverse Geocoder.
Sheet 3: (Submission at the end of the lecture period) The task is to develop the Geocoder.

Mandatory Tasks

Task 1: Understanding OSM Data Structures

As a first step, we need to get to know how data is organized inside OSM. We are primarily interested in "nodes" and "ways".

Resources

- The OSM wiki explains the core data structures as well as interpretations of individual tags.
- This youtube-playlist explains the OSM primitives and tags with lots of examples

Task 2: Extract Relevant Data from a PBF File

For a (reverse) Geocoder we are mostly interested in houses and streets. Since for houses in particular the city/country/zip-code/... isn't saved, we are also interested in administrative areas and their different layers. Extract these two from a PBF file and save them in a suitable format. For Meta-Data pay attention that you only extract the data in only a single language. For test purposes you can also save (some of) the data into a GeoJson and visualize it on sites like Geojson.io.

For a house you don't need to save the entire geometry. Find a way to reduce the geometry to a single point.

For later tasks you may want to extract more data from OSM objects. Therefore try to keep your code as expandable as possible.

Resources

- For testing, downloads of OSM data by region from geofabrik are useful (Regierungsbezirk Stuttgart is a good starting point. For grading your code should run on at least Europe)
- PBF Format

Task 3: Visualization

All features implemented during the course should be usable through a GUI. Lay the foundation for that GUI here and extend it to support every upcoming feature. At this point you can test how to display points or polygons on a map. To test your implementation write a naive routine to display all houses in the current viewport. We recommend you use Leaflet as a map framework.

Optional Tasks

Task 1: Object Information

Objects (e.g. houses, shops, ...) may lack information like a street address or the shopping mall they are contained in. Find ways to add adresses to these objects.