Routing example

Introduction

In this class we'll be learning about **routing**.

Routing consists in determining paths or travel times using a network. Routing can be used to solve an important variety of problems, such as:

- The best path (according to some cost function) between point A and B;
- Finding service areas (e.g. the area covered by T min driving);

Internally, routing algorithms rely on the mathematical Theory of Graphs, that you may have learnt before. For example, algorithms like the Dijkstra algorithm, is utilised by pgRouting to determined the shortest path between two given points.

Tools and data

- osm2po - a tool to download and prepare data for routing
- pgrouting - the PostgreSQL/PostGIS extension for routing
- geofabrik - where you can get data

About `osm2po`

Before we start: to run `osm2po` you need to have Java installed, at least Java 1.8

I have also make a version of `osm2po` for this class, that you can download **here**. For this class, use this version.

The freeware `osm2po` is a command line utility that we can use to download network data, and (very importantly) to preprocess it. Not all networks are ready to be routable. `osm2po` does that for us.

In the file `osm2po.config` we can file all possible configuration parameters. The file is long and dense, and arguably not the most friendly way to tune the tool for a specific application.

To use, reuse the one of the two demo scripts. If you are on a Mac or Linux, use `demo.sh`. If you're using Windows, use `demo.bat`. If you open any of the two scripts, you'll see a long line, where Java is called to perform all the necessary processing.

By default, data is downloaded from `geofabrik`. If you want to get data from a particular region, go to `geofabrik` website, and copy the link of the file `.osm.pbf`. Alternatively, you can download the file yourself, and replace the url for the internal file path to the data.

When running `osm2po` you'll be presented with heaps of information in the command line that informs you how the process is going on. Once the process

is finished, you will find a new folder, by default, `hh/`. In it, you'll find a large number of files. But the file you need is the only `.sql` file you'll find there.

The generated `.sql` file contains all the network that was processed. Carefull when open it, since this file, depending on the size of the area that you've selected, can be quite large, and thus it can be quite hard for a regular text editor to open it.

With this file, you can simply load it into a database (assuming that datatabase contains the extensions `postgis` and `pgrouting` activated), and run it. After some time, you will have a table (by default in the schema `public`), ready to be used for routing applications.

Tutorial

Download and process data with `osm2po`

Because we do not want to be bound by connectivity problems, I have downloaded the file that we'll be using: `new-york-latest.osm.pbf`.

Next run `demo.sh` if you're using a Mac or Linux. And run `demo.bat`, if you are running windows.

Load data into a database

- With `pgAdmin4`, connect to NYC database that we've created before.
- Create the extension `pgrouting`.
  - Remember the NYC database has `postgis` extension already activated.
- Open the `.sql` file created by `osm2po`, and run it.
- Check if a new table, named `hh_2po_4pgr` is present in `public` schema.

Explore the network table

- Use `QGIS` to connect to the database and load it into the canvas.
- Click in one segment. What do you make of its attributes?

Find the shortest path from vertice `608635` to vertice `273736`

To answer to this question, we'll use the Dijkstra Algorithm. Read more here.

First, run the following query:

```
SELECT *
FROM pgr_dijkstra(
    'SELECT id, source, target, cost, reverse_cost
    FROM hh_2po_4pgr', 608635, 273736, true
);
```

What do you make of it?

- `km` is the distance between source and target (using the Sphere as approximation)
- `kmh` is the maximum speed allowed in that segment
- `cost` is the time (hour as unit) taken to travel the segment (`km` / `kmh`)

Next, run the following query, and using QGIS DB Manager, load it into the canvas:

```
SELECT t0.*, hh.*
FROM (
    SELECT *
    FROM pgr_dijkstra(
        'SELECT id, source, target, cost, reverse_cost
        FROM hh_2po_4pgr', 608635, 273736, true
    )
) t0
INNER JOIN hh_2po_4pgr hh
ON t0.edge = hh.id
ORDER BY t0.seq;
```

What do you see?

Get the 5-min "driving distance" polygon from vertice 587224

Have a look at the driving distance function documentation here Run the following query:

```
SELECT *
FROM pgr_drivingDistance(
        'SELECT id, source, target, cost, reverse_cost
        FROM hh_2po_4pgr',
        587224, 1.0/12.0
      );
```

Next run this:

```
SELECT t0.*, hh.*, t0.agg_cost * 60 AS travel_time
FROM (
    SELECT *
    FROM pgr_drivingDistance(
        'SELECT id, source, target, cost, reverse_cost
        FROM hh_2po_4pgr',
        587224, 1.0/12.0
      )
) t0
INNER JOIN hh_2po_4pgr hh
ON t0.edge = hh.id
ORDER BY t0.seq;
```

Next convert the outcome in a polygon. Hint: use `ST_ConvexHull`.