LineGraph Utility

a software module *for* use in

flexible *routing*

of public transportation

Jonas Bergman, jobe0900@student.miun.se

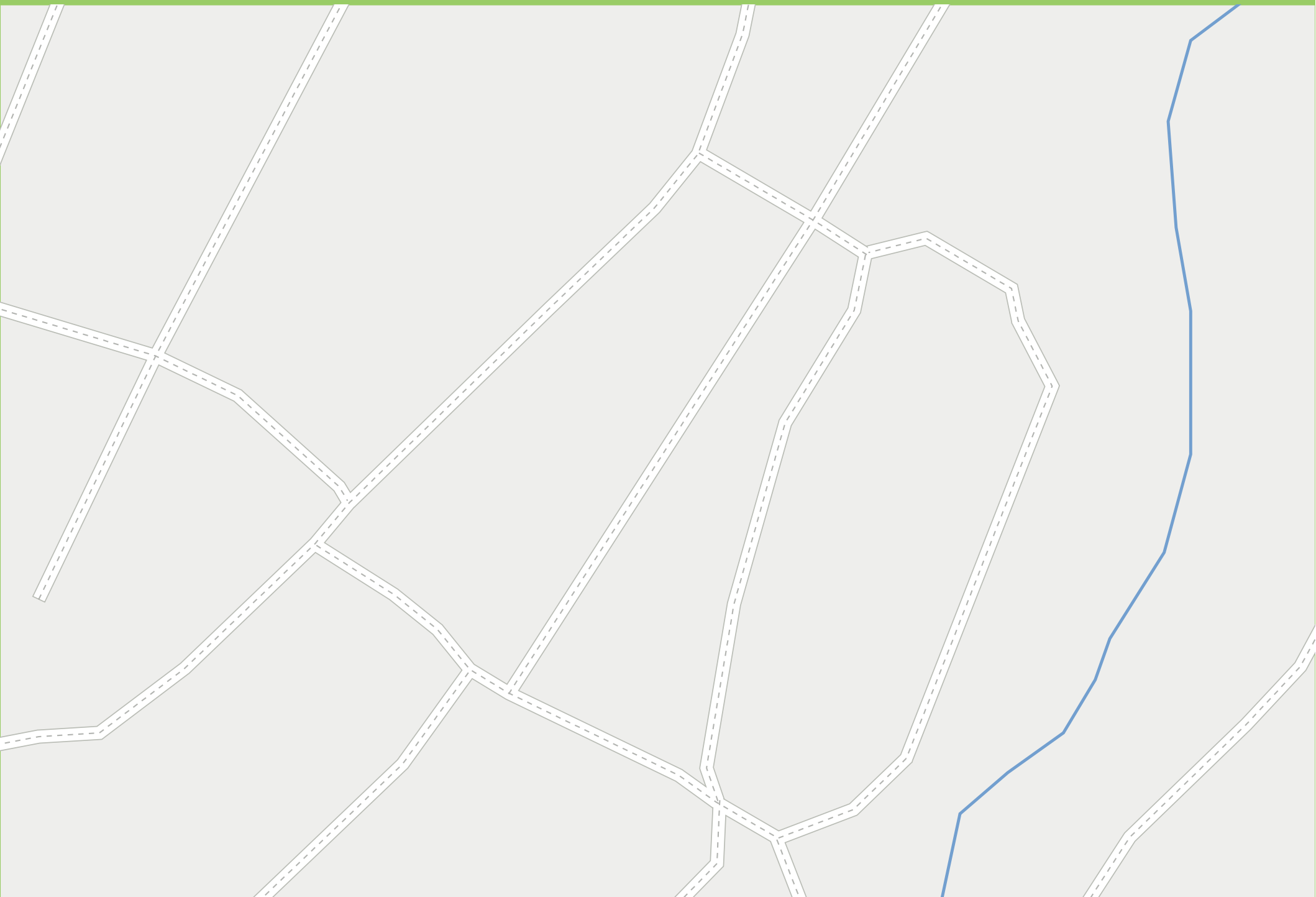
requirements

in: ***map*** data

out: ***line graph***

1 load ***map*** data

1. map



1. map

source:

OpenStreetMap

www.openstreetmap.org

load:

```
$ osm2pgsql -U user -d map_db -s -k mapdata.osm
```

store:

PostGIS

PostgreSQL + spatial data extension

www.postgis.net

1 load ***map*** data

2 build *topology*

2. topology

2. topology

tool:

`postgis_topology`

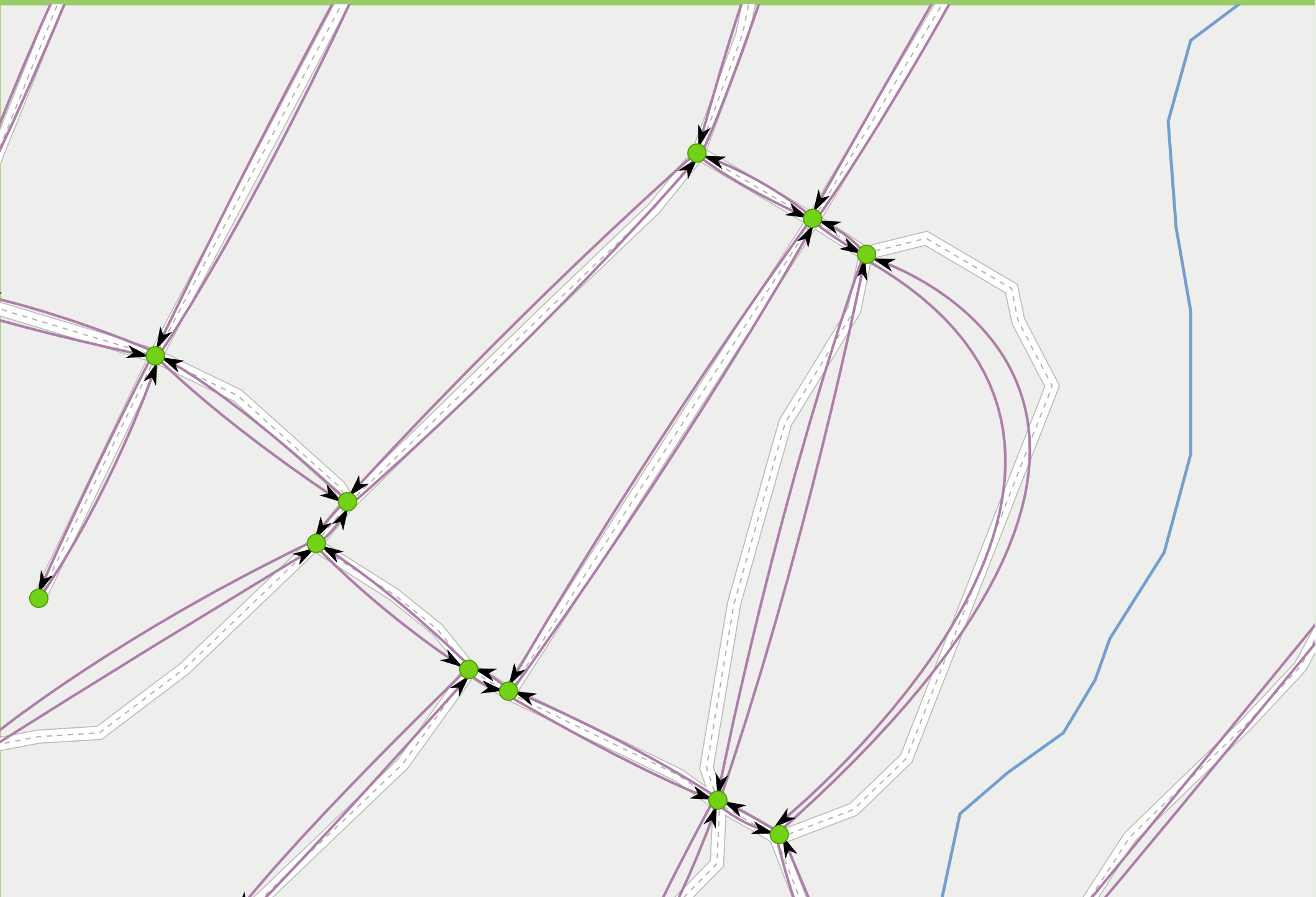
1 load ***map*** data

2 build *topology*

3 apply *restrictions*

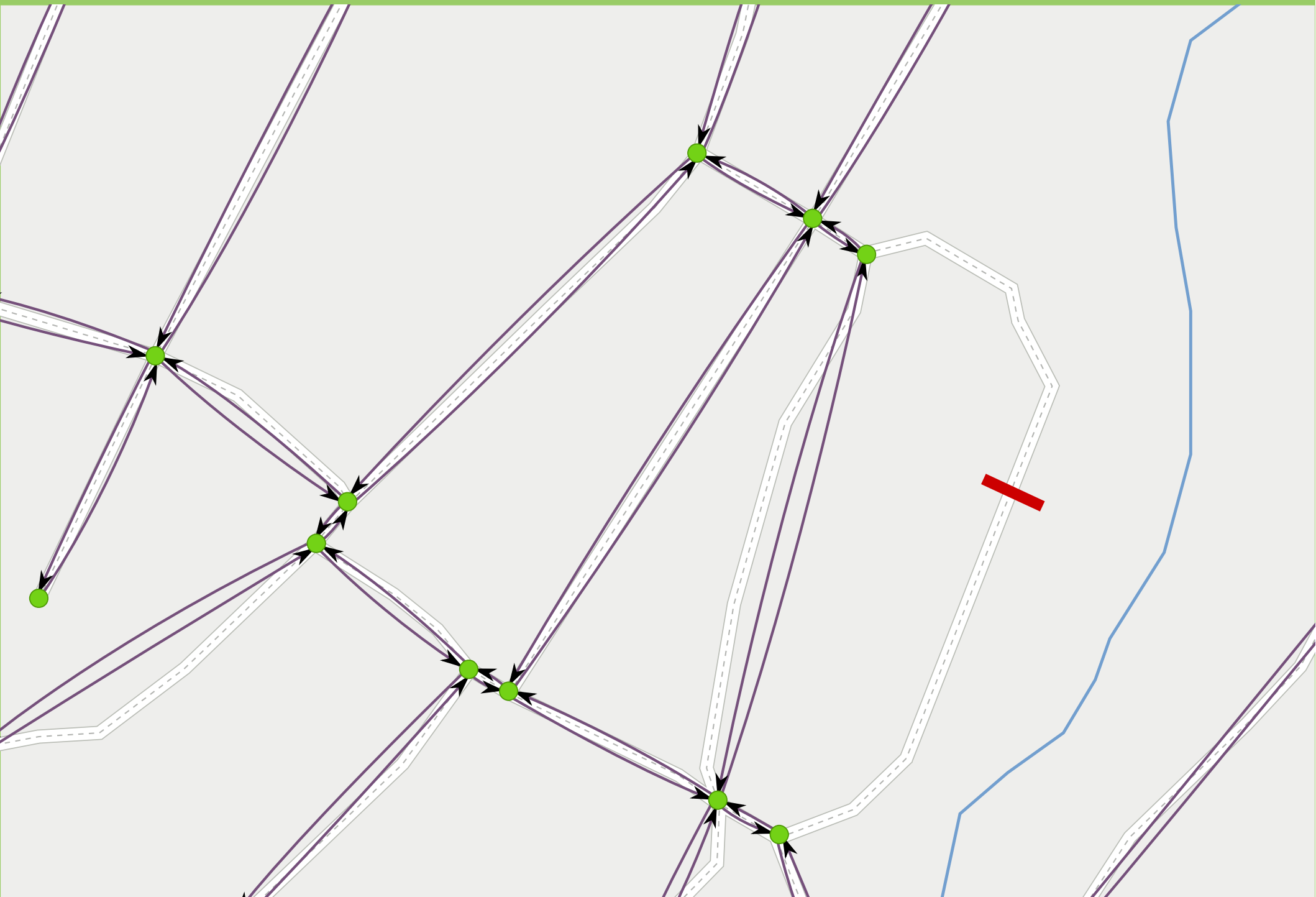
3. restrictions (directed graph)

3. restrictions (directed graph)

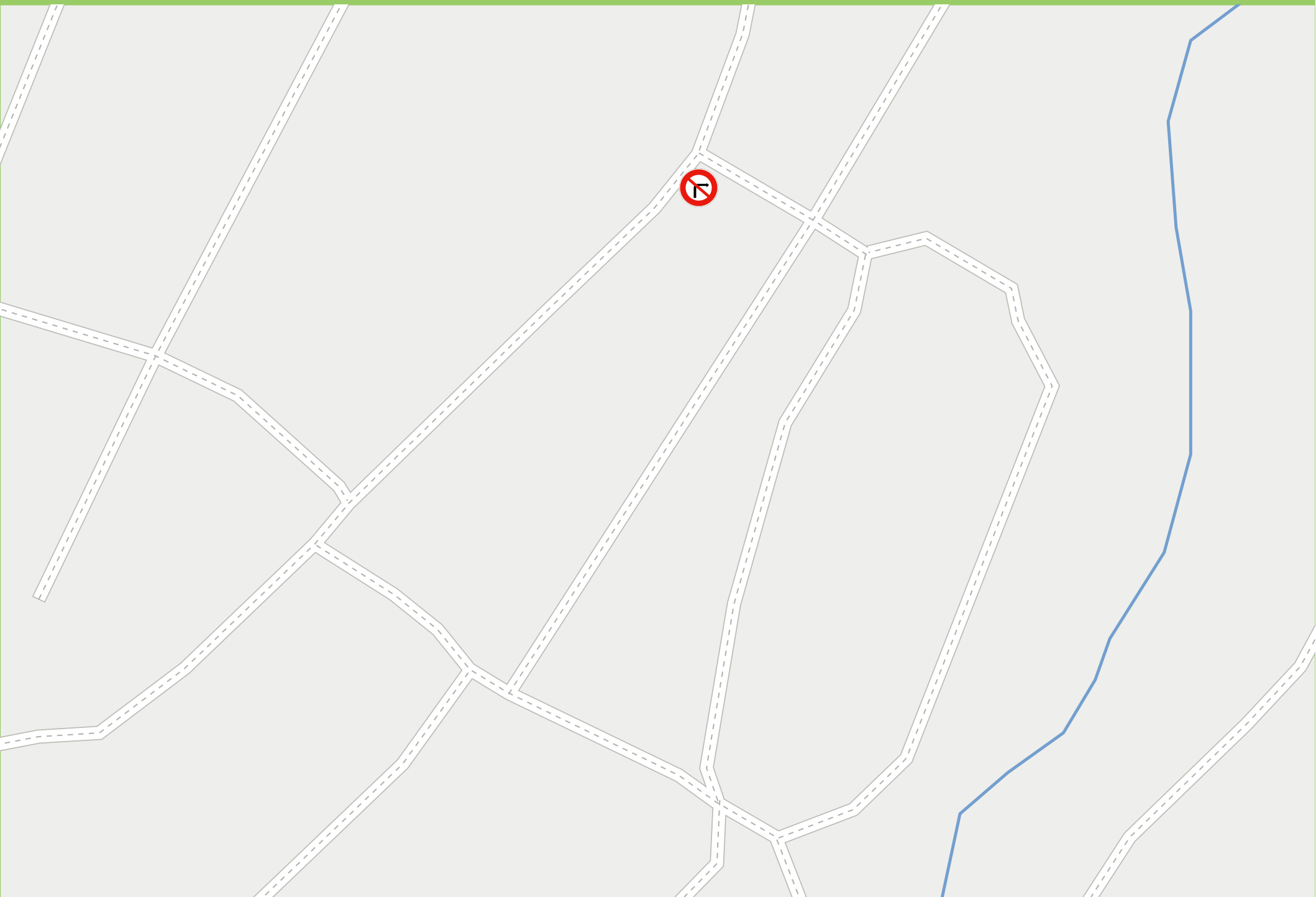


3. restrictions

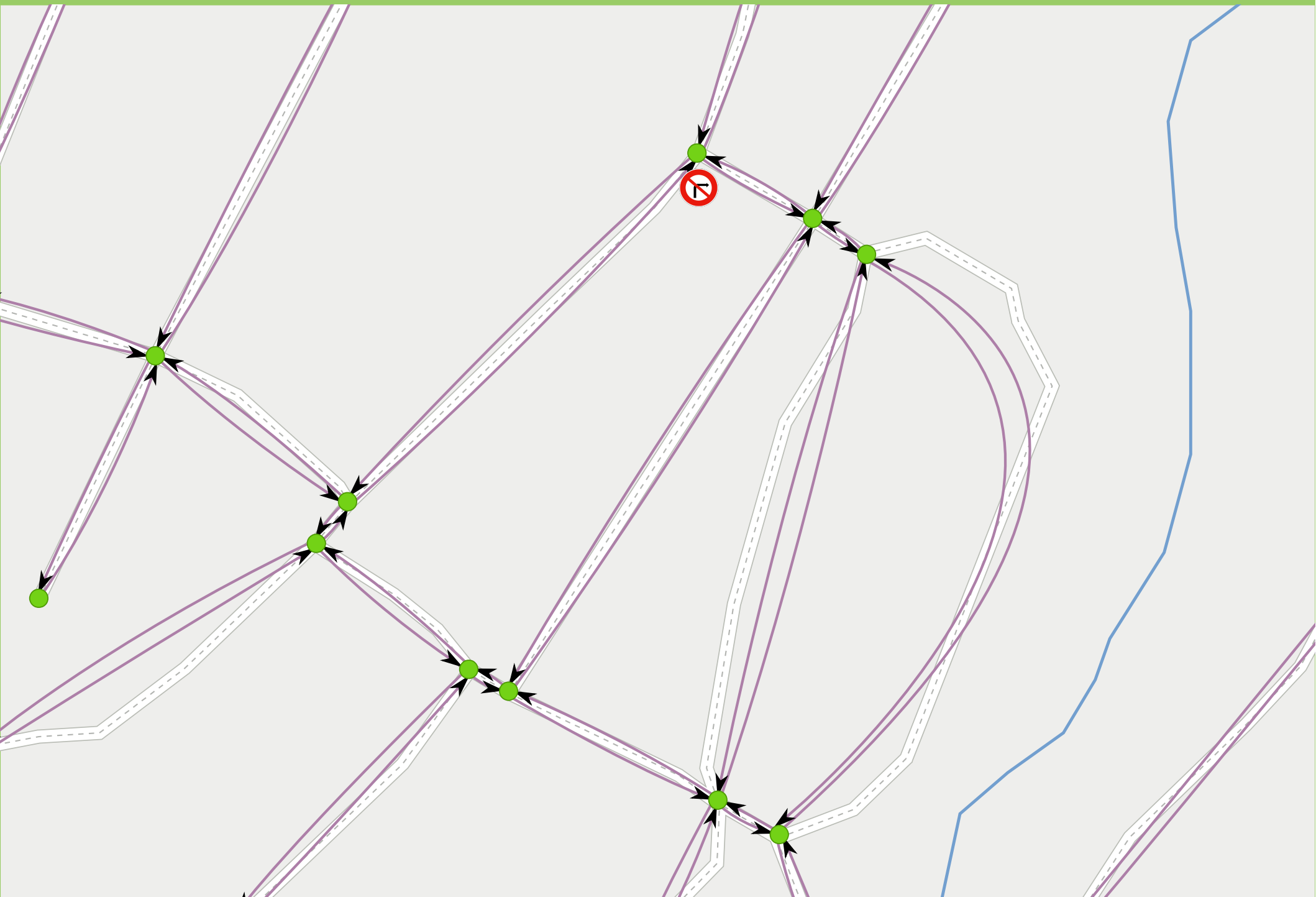
3. restrictions



3. restrictions



3. restrictions



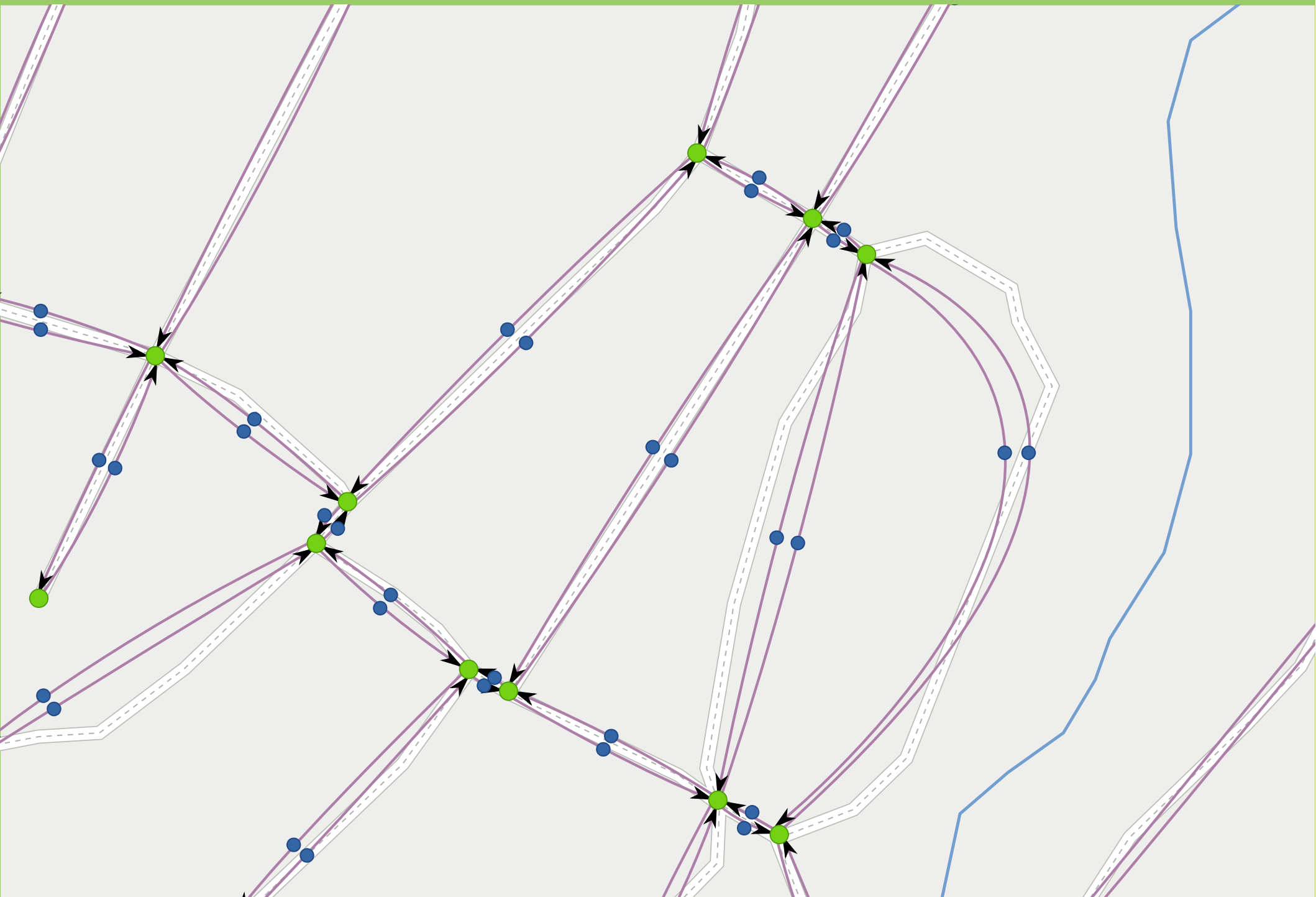
1 load ***map*** data

2 build *topology*

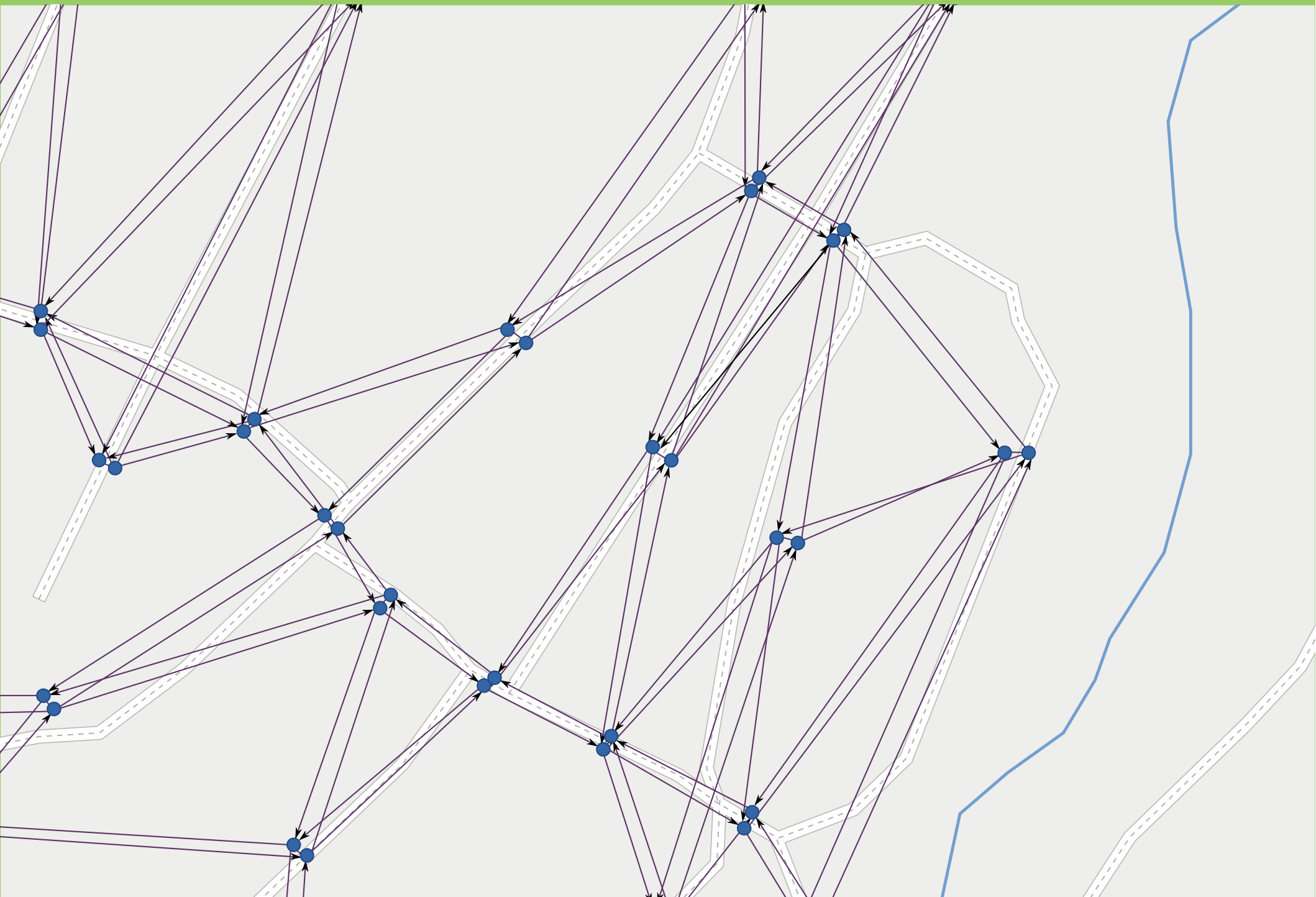
3 apply *restrictions*

4 build ***line graph***

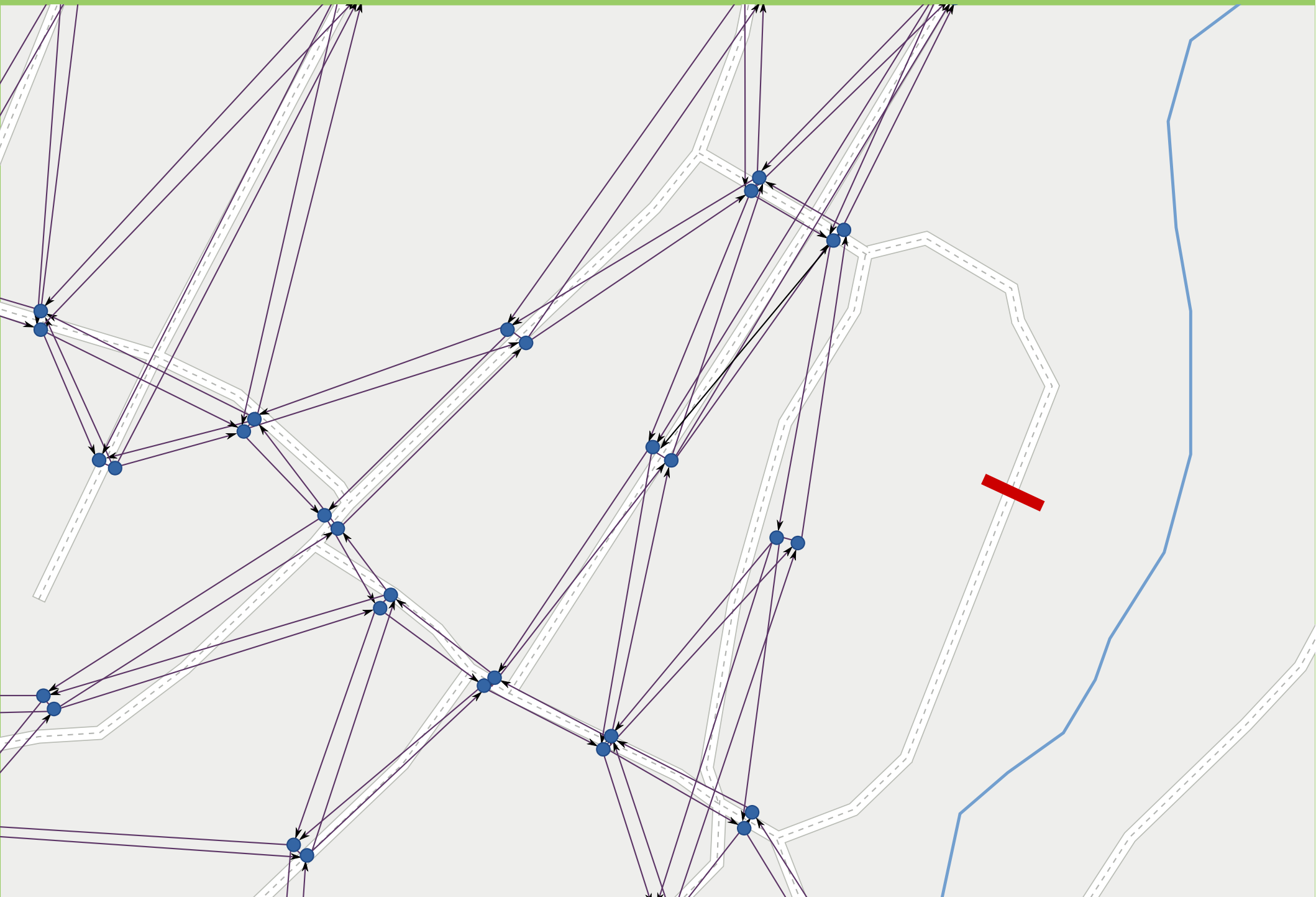
4. line graph



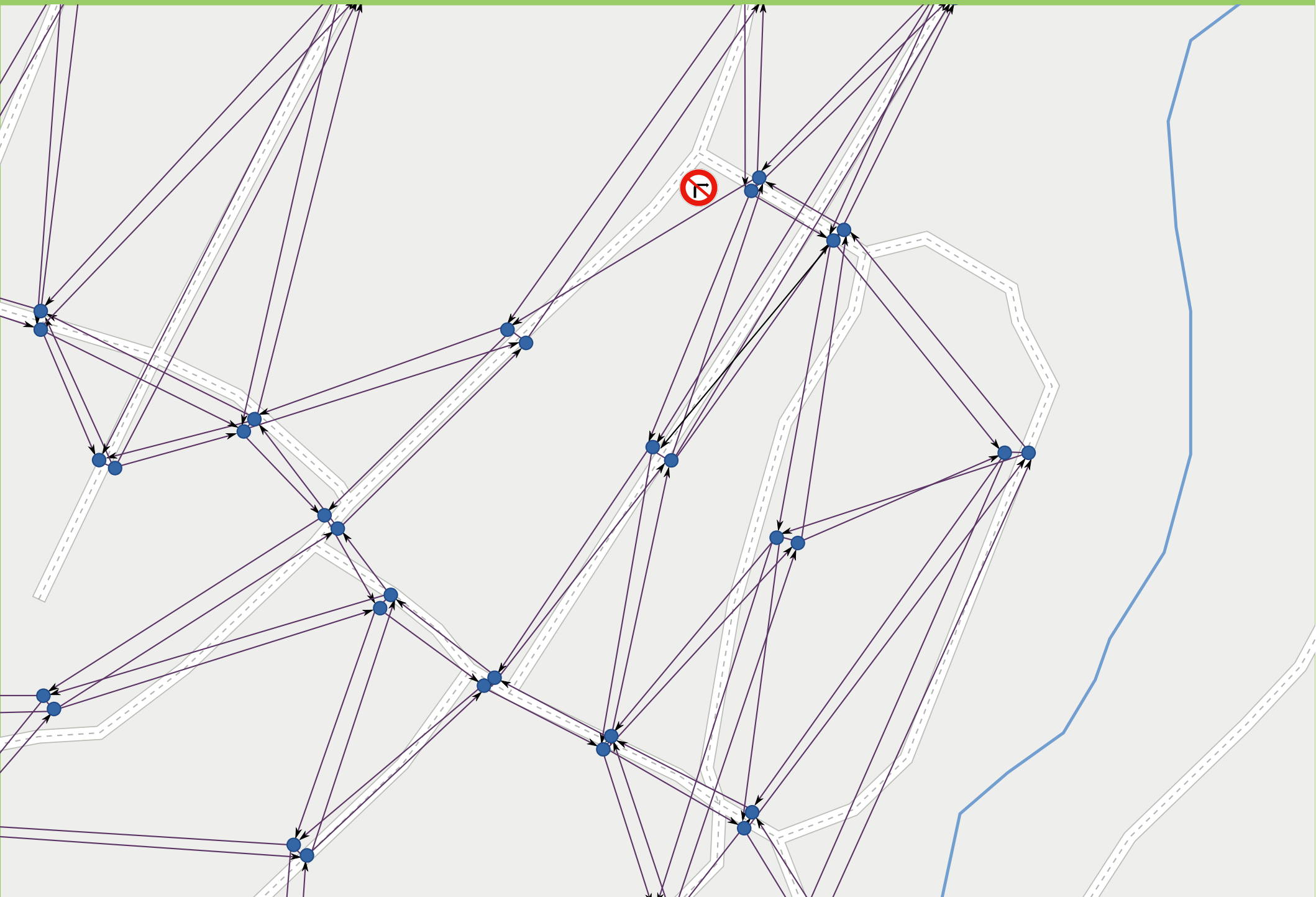
4. line graph



4. line graph



4. line graph



tool:

Boost graph library

```
typedef boost::adjacency_list
    < boost::listS, boost::vecS, boost::directedS,
      LineGraphNode, LineGraphLine >
LineGraphType;
```

remarks

preliminary

load ***map*** data

build *topology*

on demand

apply *restrictions*

build ***line graph***

configurable

json file
for ***settings***

database, vehicle properties,

road speeds, surfaces,

restrictions and costs, ...

mapping IDs

OSM +

topology +

graph +

line graph

restrictions

values:

yes, no, permissive, designated, private, discouraged, delivery, customers ...

routing:

one-way (explicit / implicit), lanes ...

transportation mode:

all, foot, vehicle, bicycle, motor_vehicle, motorcycle, motorcar, goods, hgv ...

by use:

psv, car_sharing, emergency, hazmat, disabled ...

dimensions:

max height, weight, width ...

conditional restrictions


```
motor_vehicle=no
motor_vehicle:conditional=yes @ (18:30-07:30)
psv=yes
```

conditional restrictions

```
maxspeed=none
maxspeed:conditional=
  120 @ (06:00-20:00);
  100 @ (22:00-06:00)
```

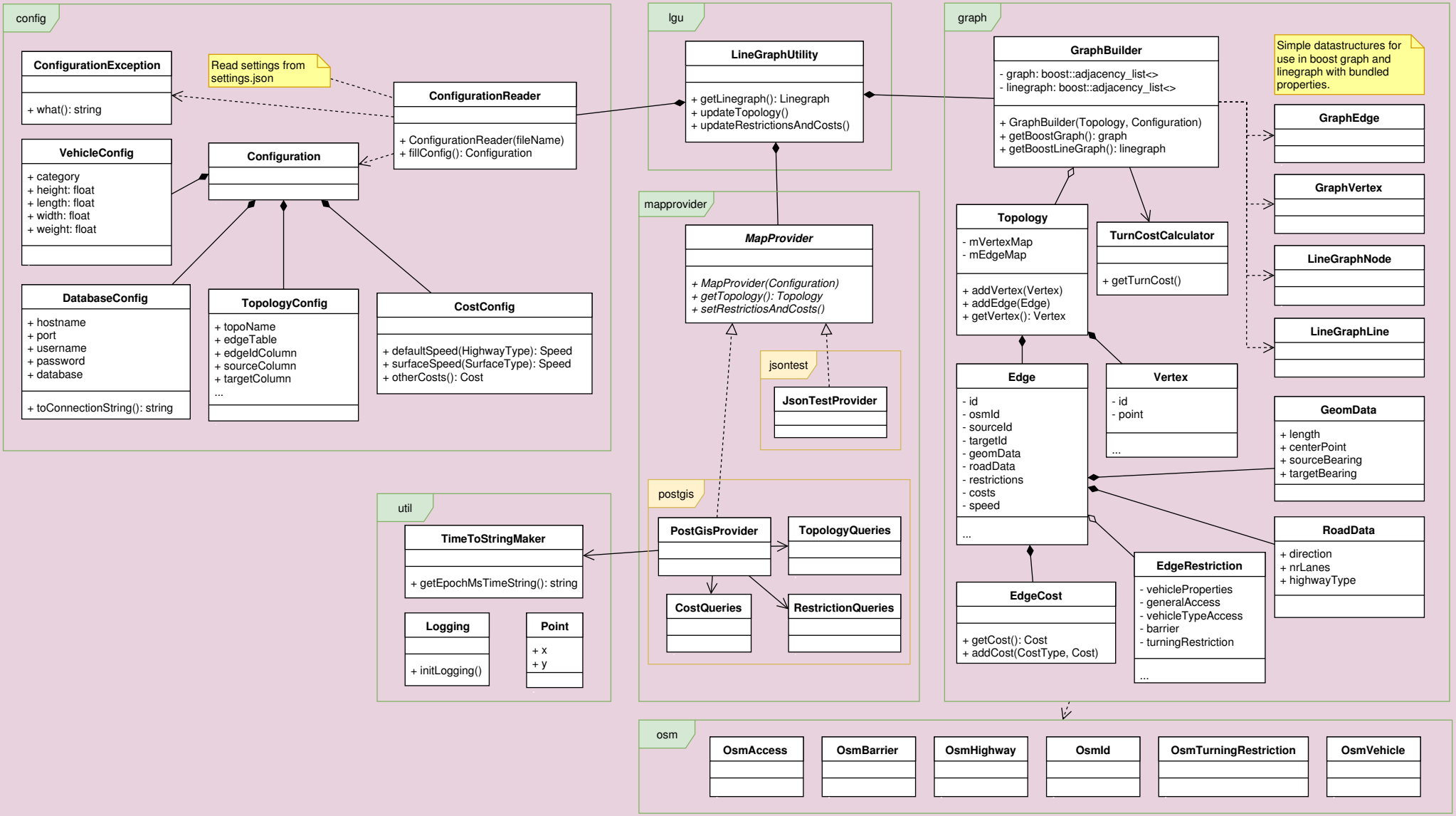
turning restrictions

Relation:

from → via → ***to***

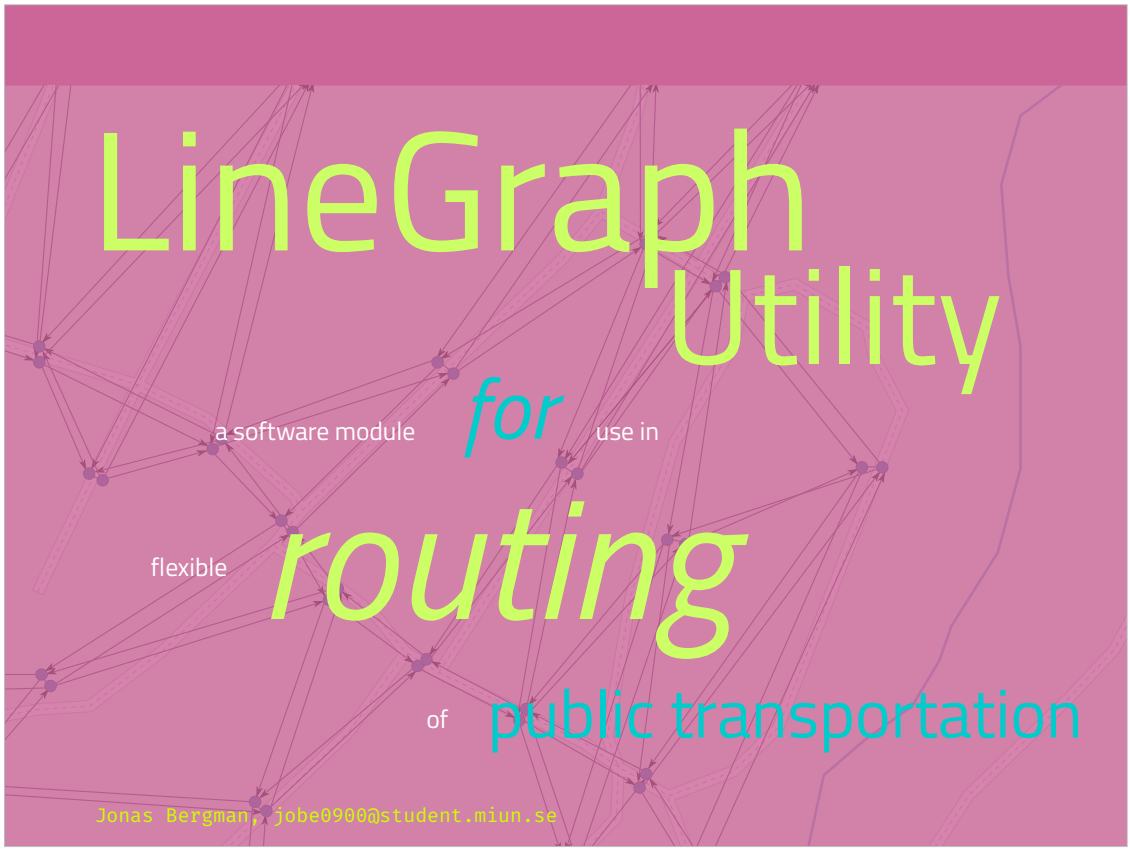
solution

class diagram



LineGraph Utility

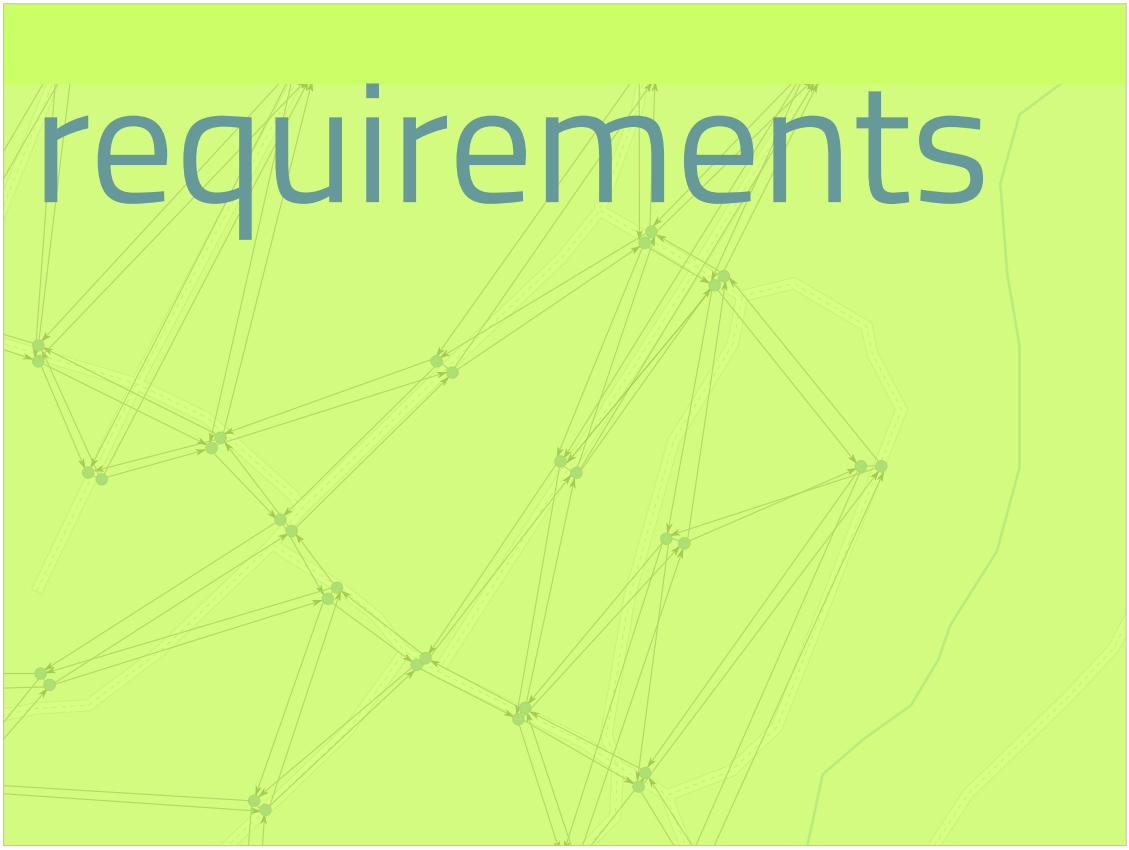
Jonas Bergman, jobe0900@student.miun.se



This is a module, part of bigger project
I have no overview of complete project.

Aim: (I think)
No waiting at bus stops
Vehicles gets directed to customers
Drivers needs real-time directions
Updated with current traffic situation

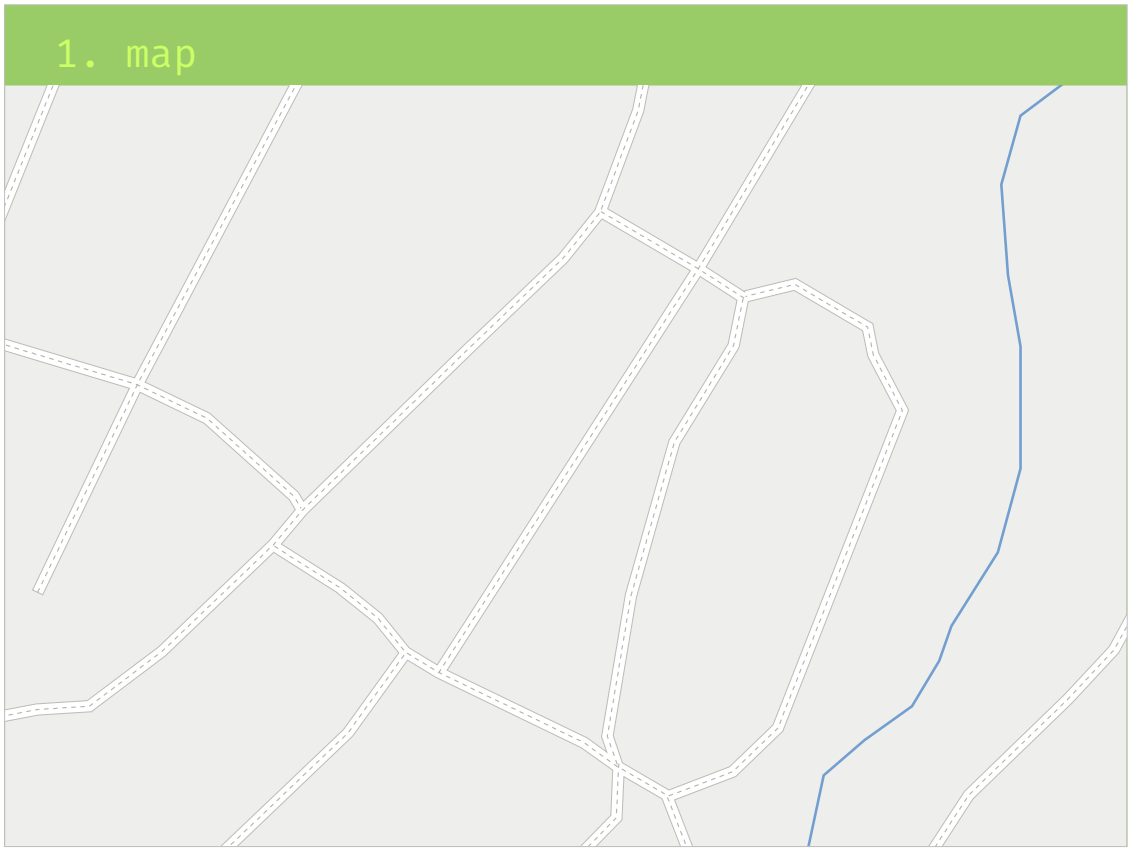
requirements

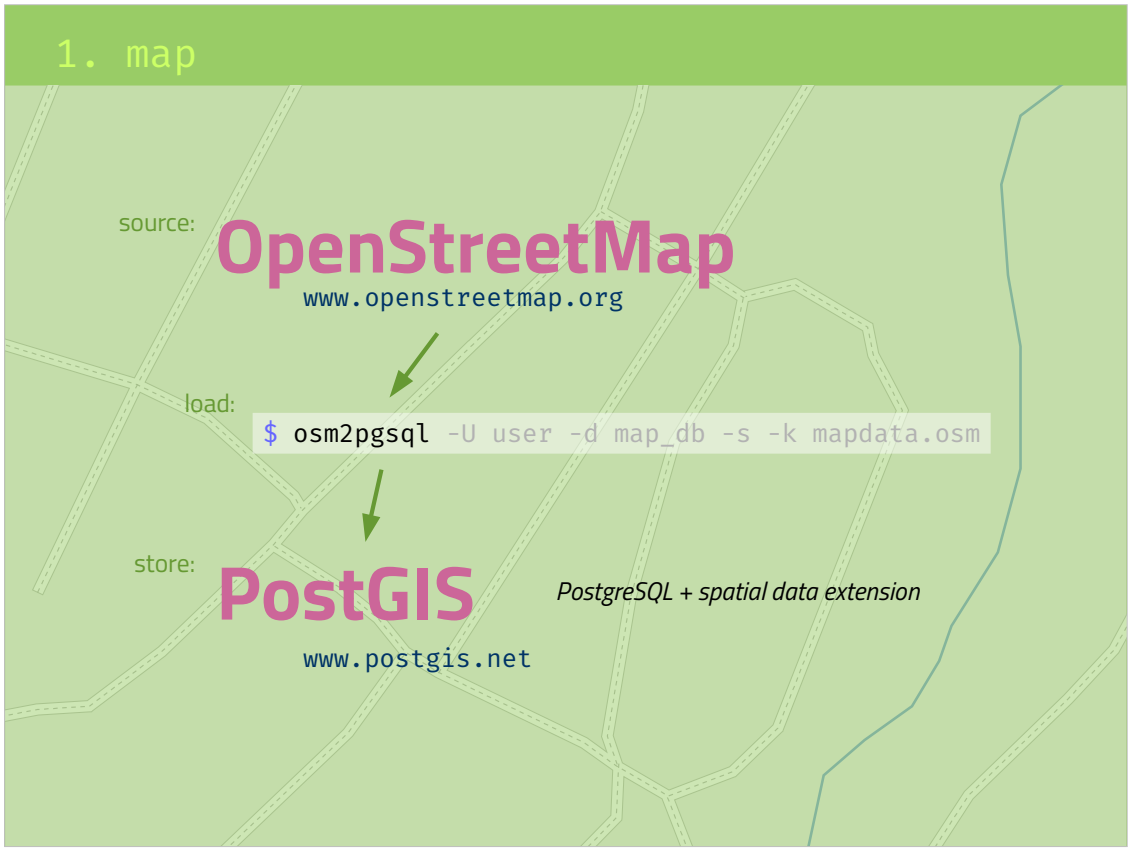


in: ***map*** data

out: ***line graph***

1 load *map* data


1. map



Open street map:
- crowdsourced, open data
- xml file format

PostgreSQL
+ geometric & geographic data

Osm2pgsql
Commandline tool
Load map data in postgis
Parsing .osm file

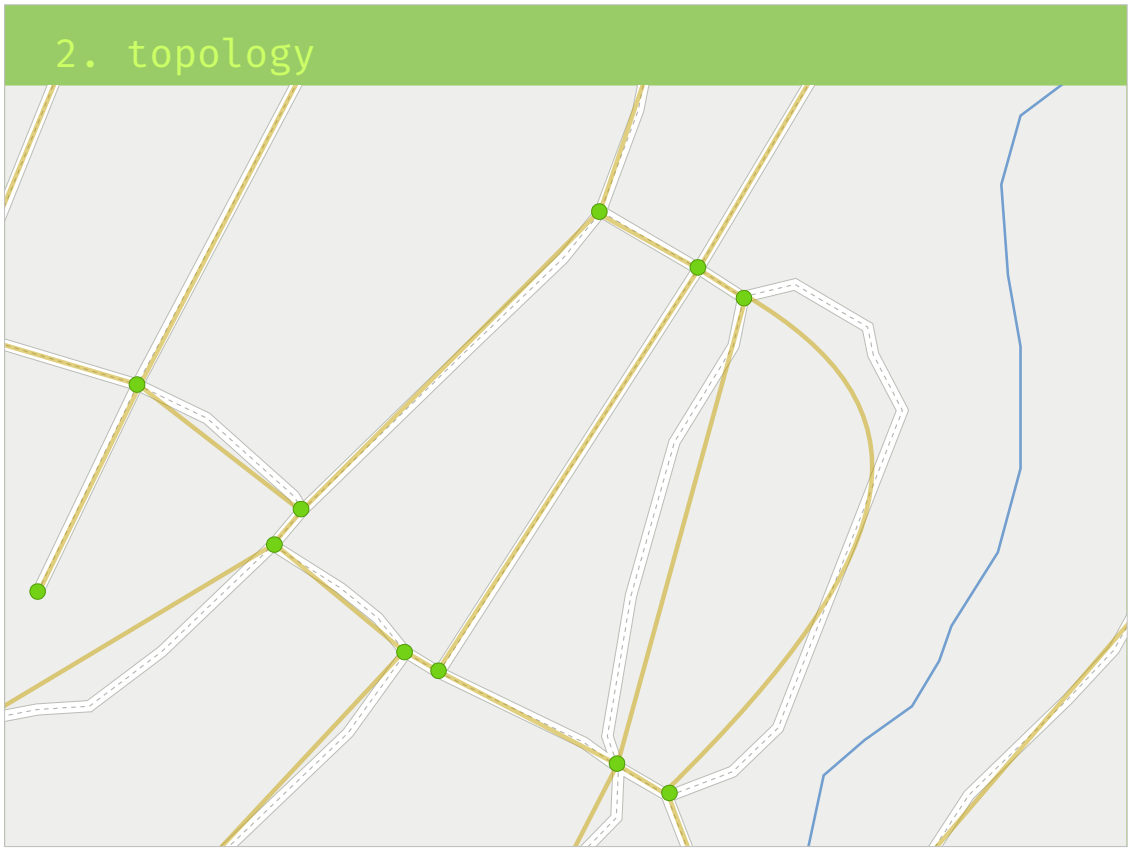
1 load ***map*** data

2 build *topology*

What is a map?

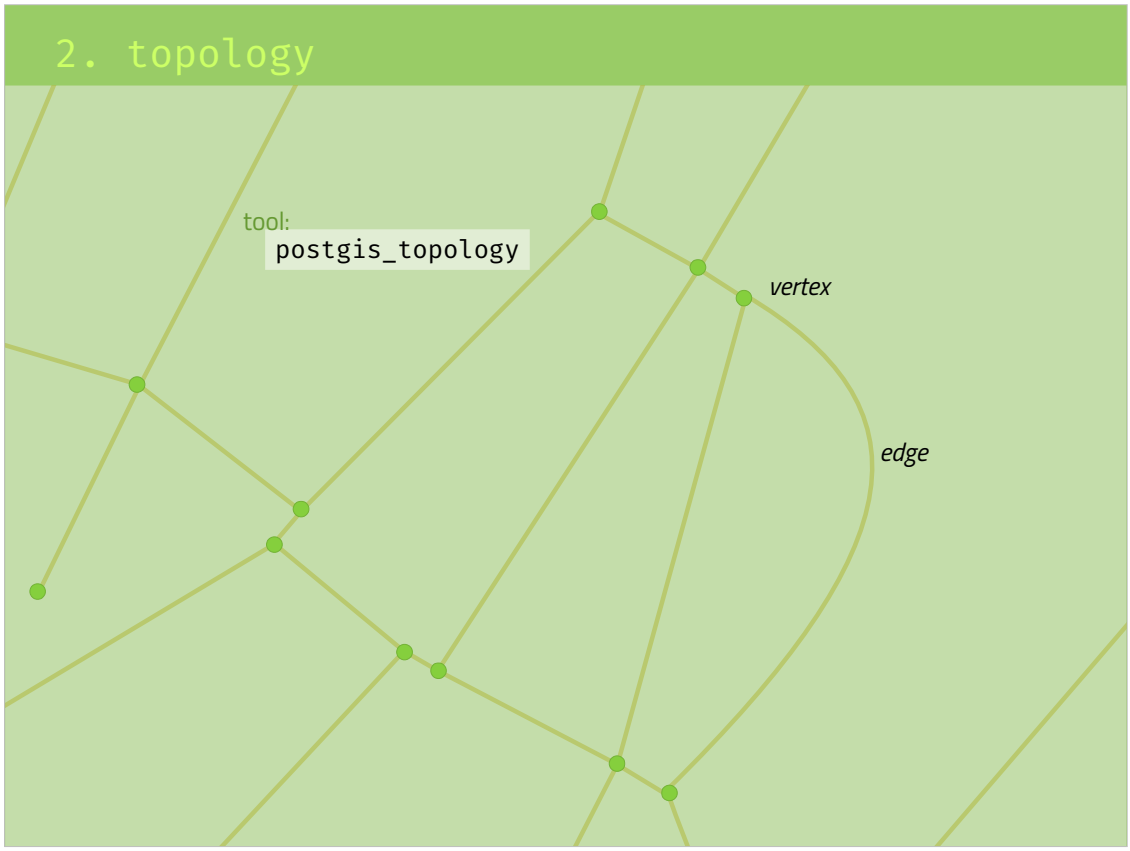
- a “projection” of reality on flat surface?
  - geometry / looks

- semantics about the connections of roads?
  - in routing YES



BUT:
- Only relations are important
- Not geometry

Topology = undirected graph
Stating relations between roads/edges



Terminology (no standard):
- vertices = nodes, points, dots
- edges = lines, arcs

When building/analyzing:
- cleaning up data
- healing misses

Chosen tool:
- extension to PostGIS
- functions to analyze and build topology

Topology should be static.
- road network : not much change

Variations reflected by costs and restrictions.

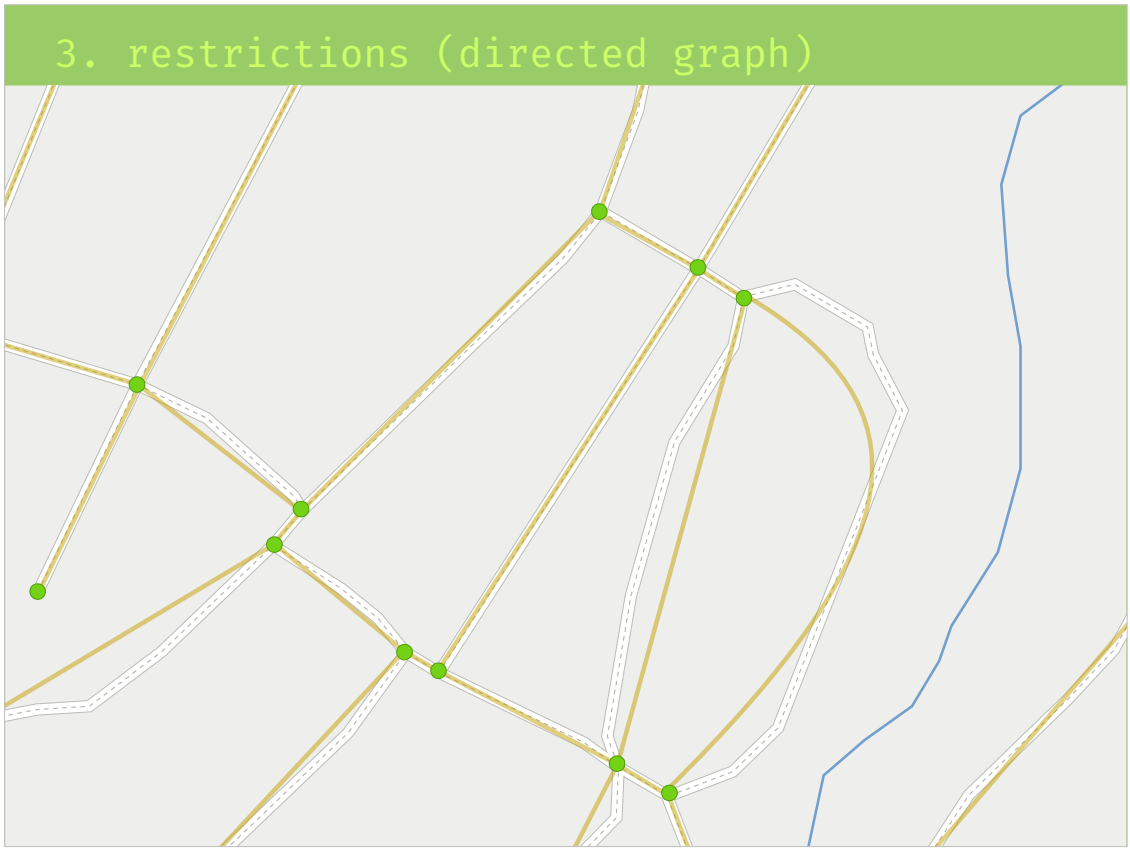
Restrictions can be static and dynamic

Restrictions: STATIC
- road signs
- traffic lights
- speed bumps

Restrictions: DYNAMIC
- accident
- road work

How to apply restriction on topology?

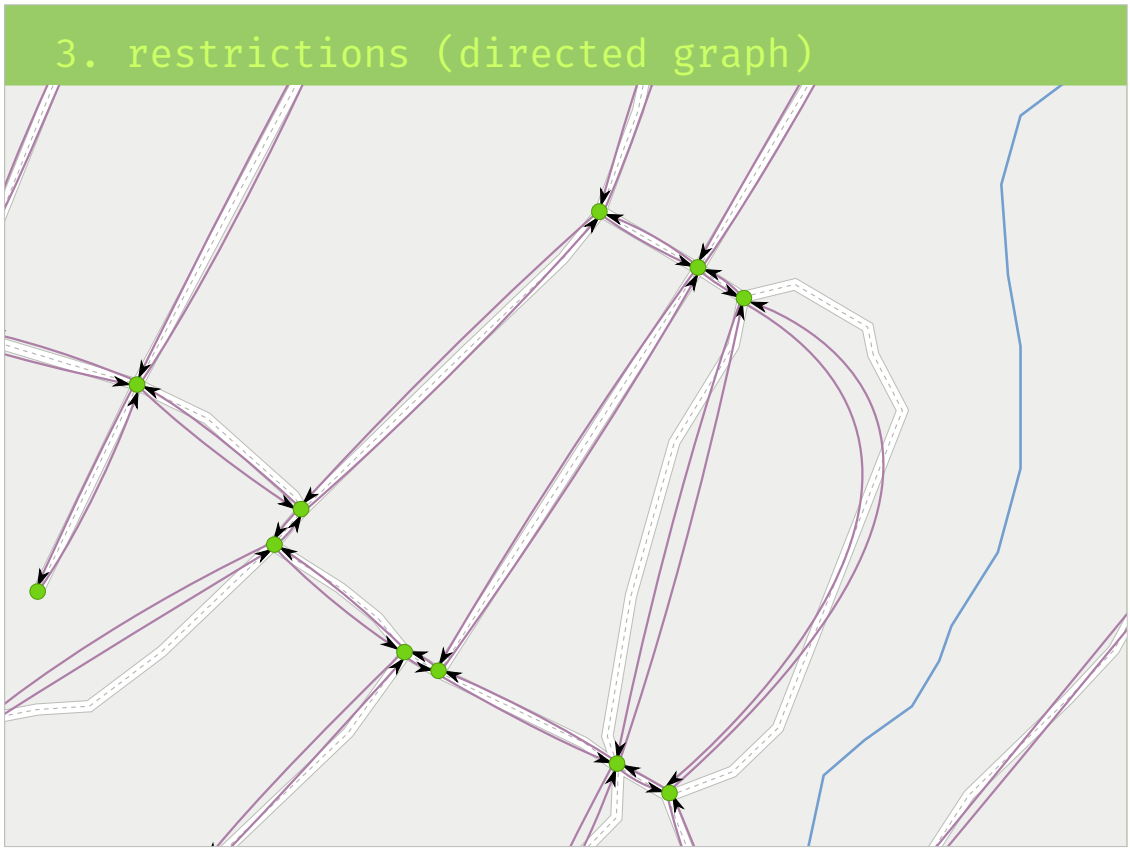
Preliminary step: directed graph



The topology is an undirected graph, no sense of direction.

Routing needs directions : needs directed graph.

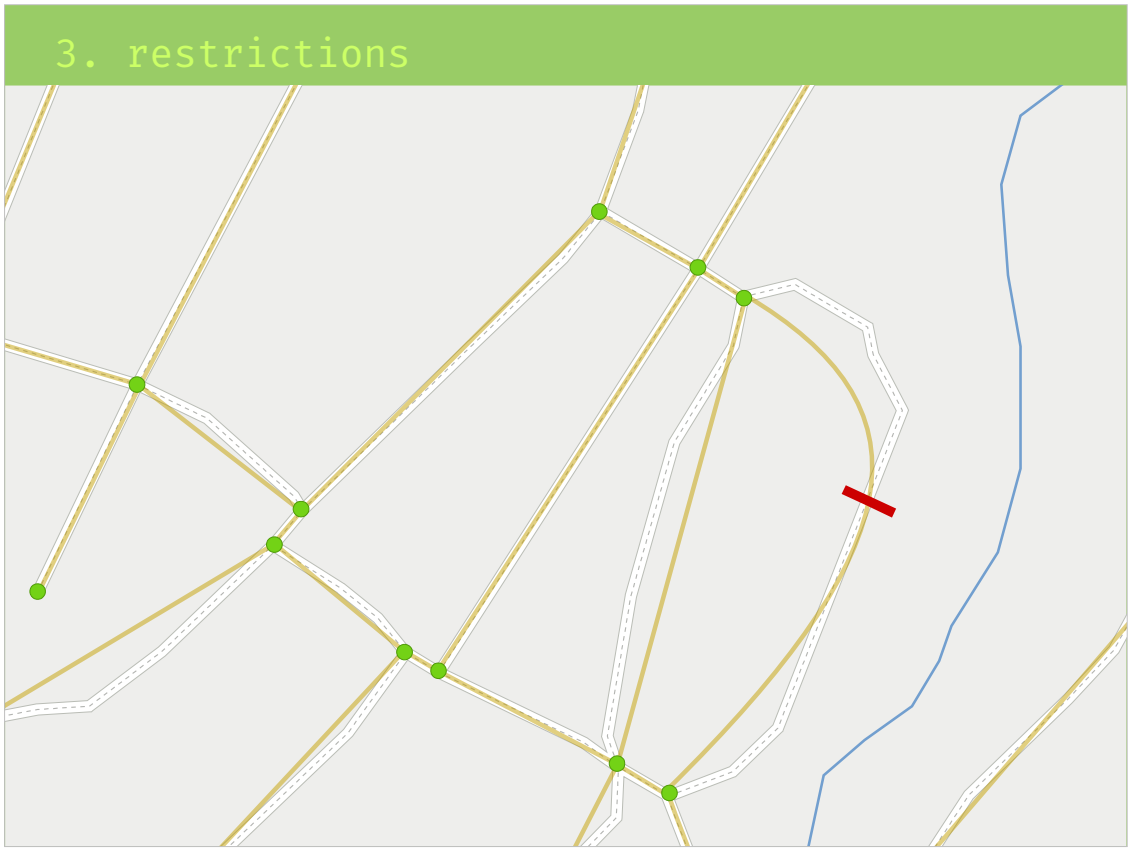
Each lane of a road gets its own edge



Now have DIRECTED graph

One edge per LANE

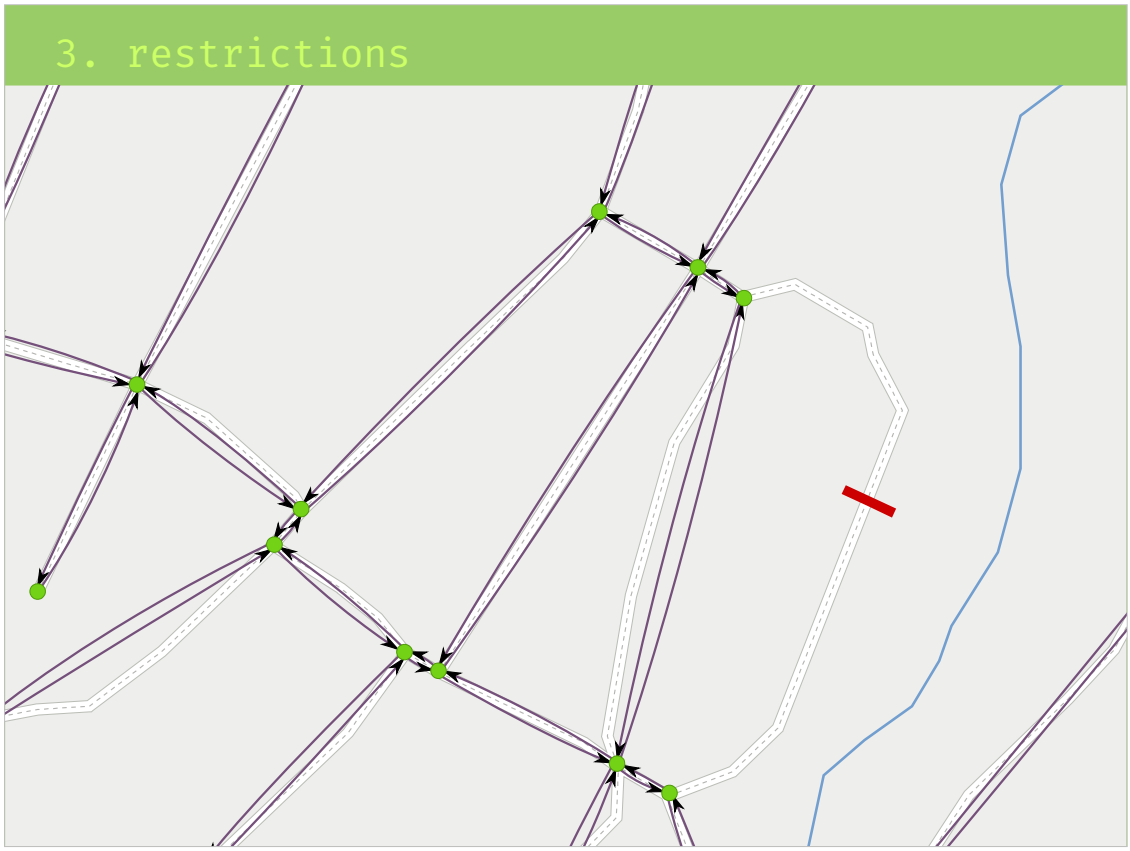
One-way = one edge



Back to restrictions:

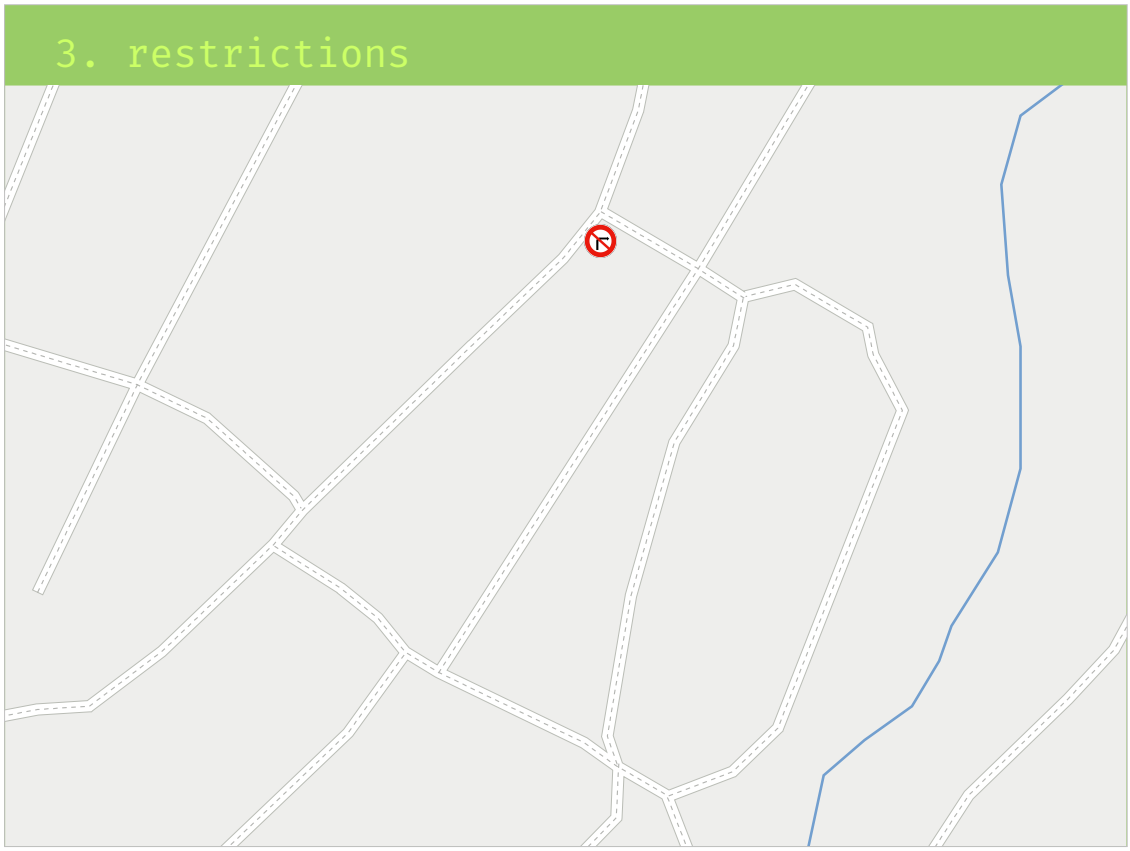
Does not affect topology

Barrier stops travel via the road = no edge



Barrier stops travel via the road = no edges

QUESTION:
What if target of route is among those edges?
- This line graph routing needs to be supplemented with short distance routing.



Turn restriction is different:
- RELATION between edges, not applied to ONE edge.

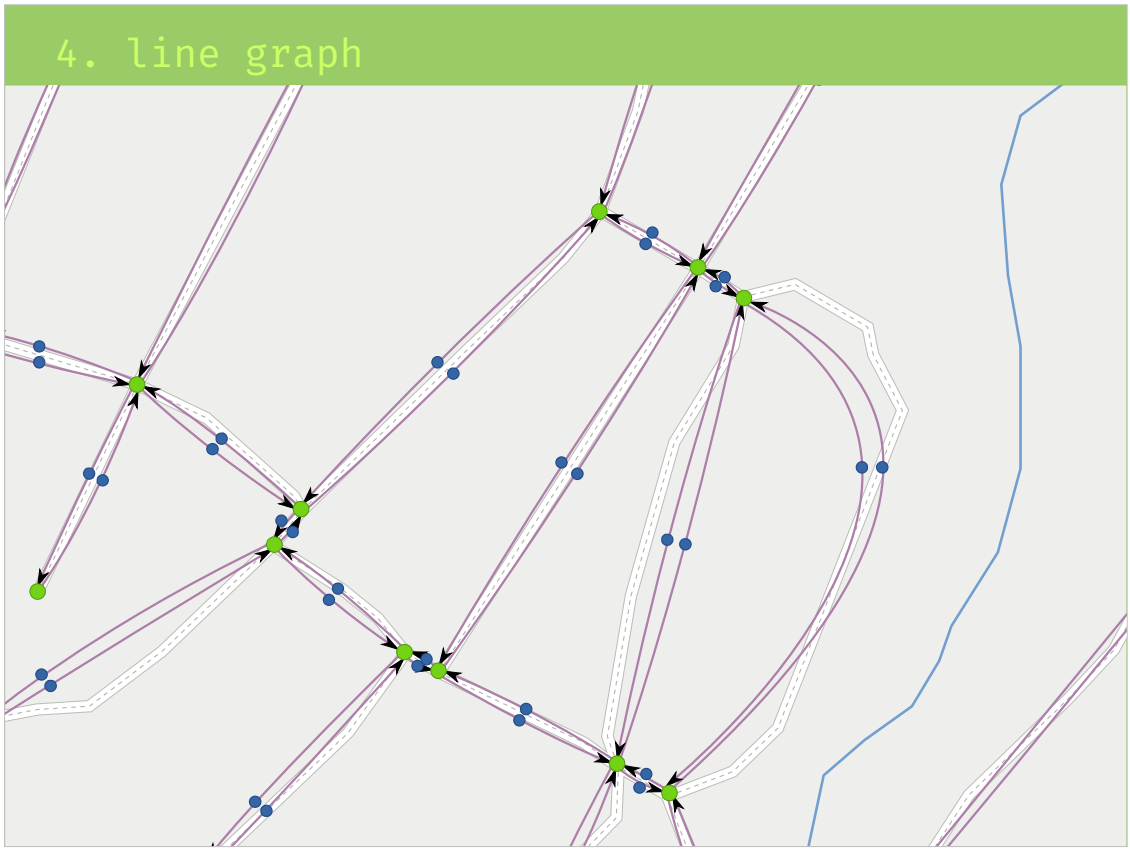
Turn restriction is different:
- RELATION between edges, not applied to ONE edge.

Means edges in graph does not disappear.

Now we have a directed graph, with restrictions.

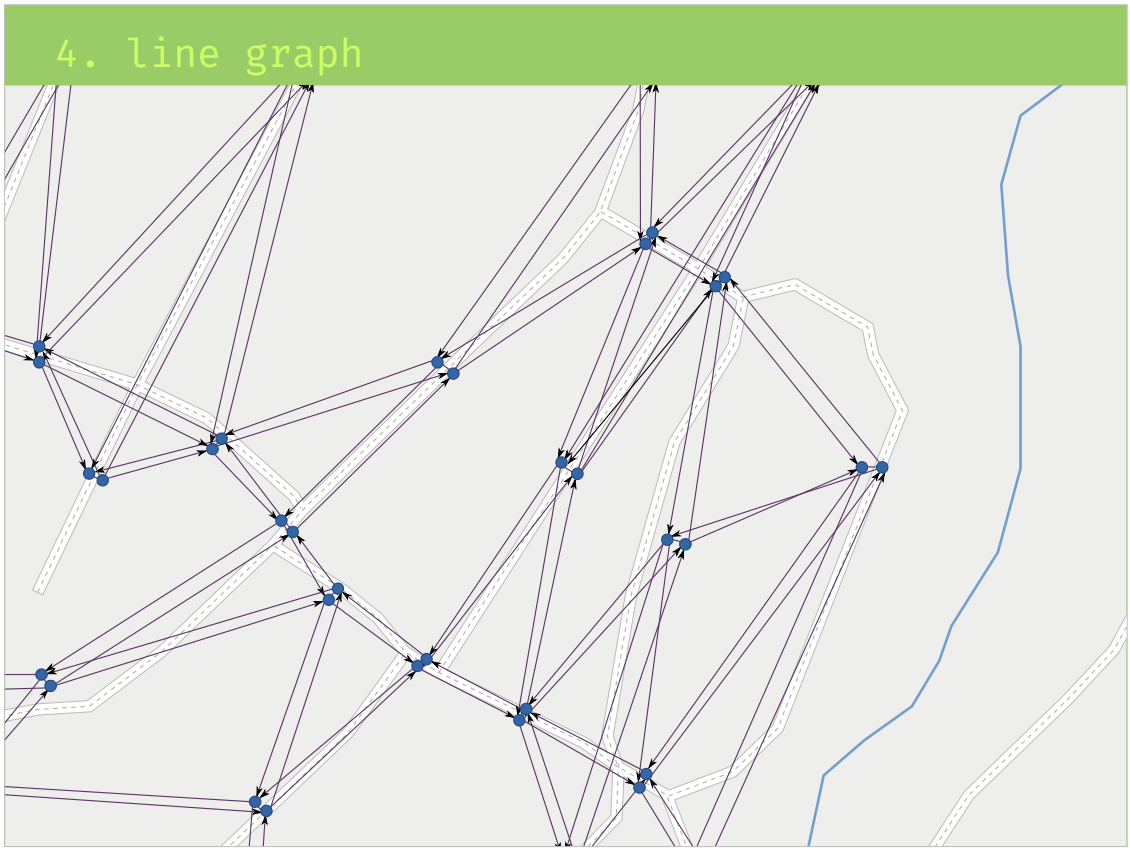
Time to build a line graph.

A line graph is a transformation of a graph.



The transformation:

Each edge in the graph is made to a “node” in the line graph.

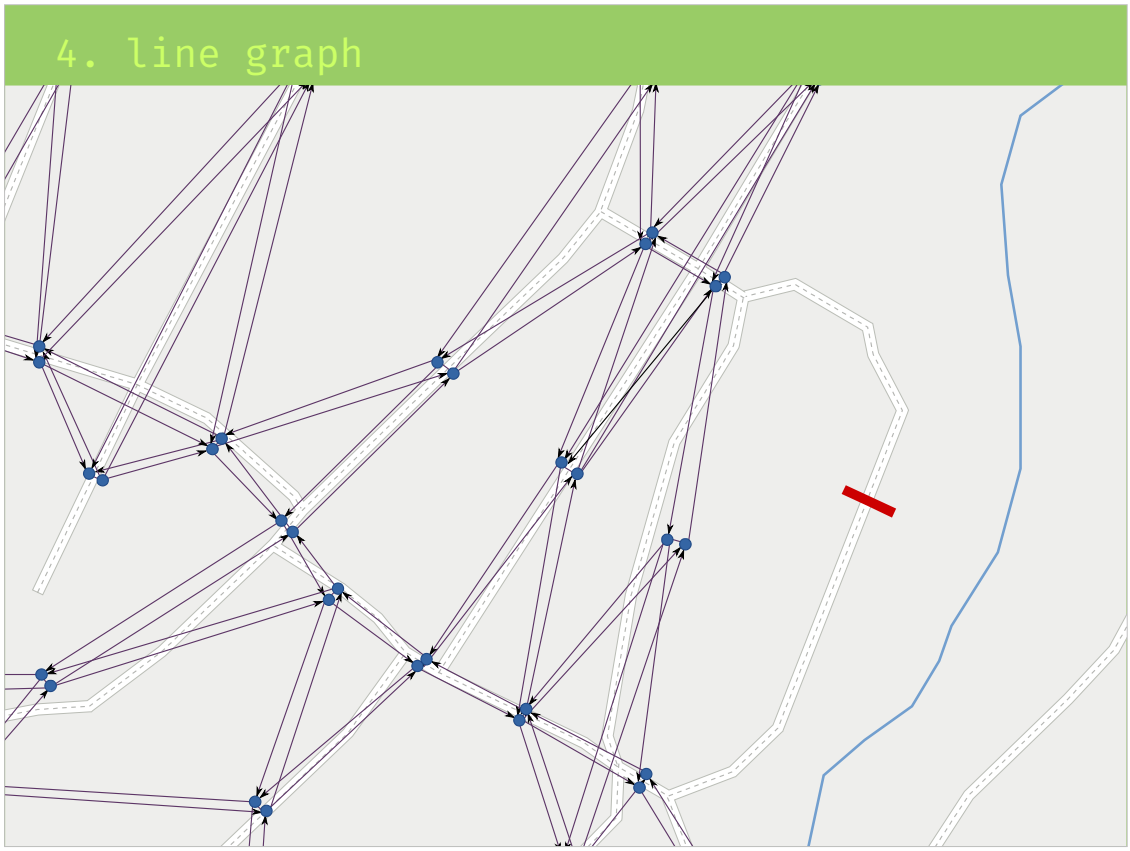


Then the nodes are connected with directed lines where travel is possible.

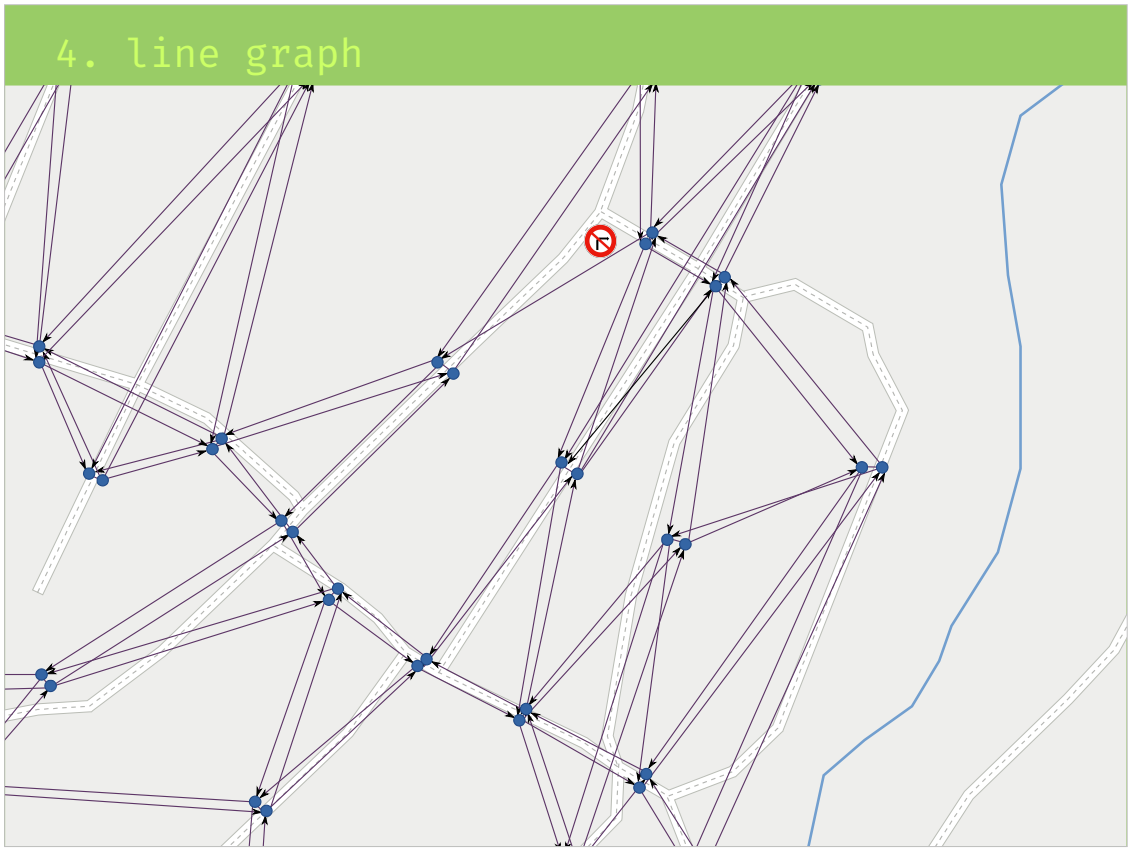
(Note: U-turns)

See: a lot more lines than there were edges → increase in space needs.

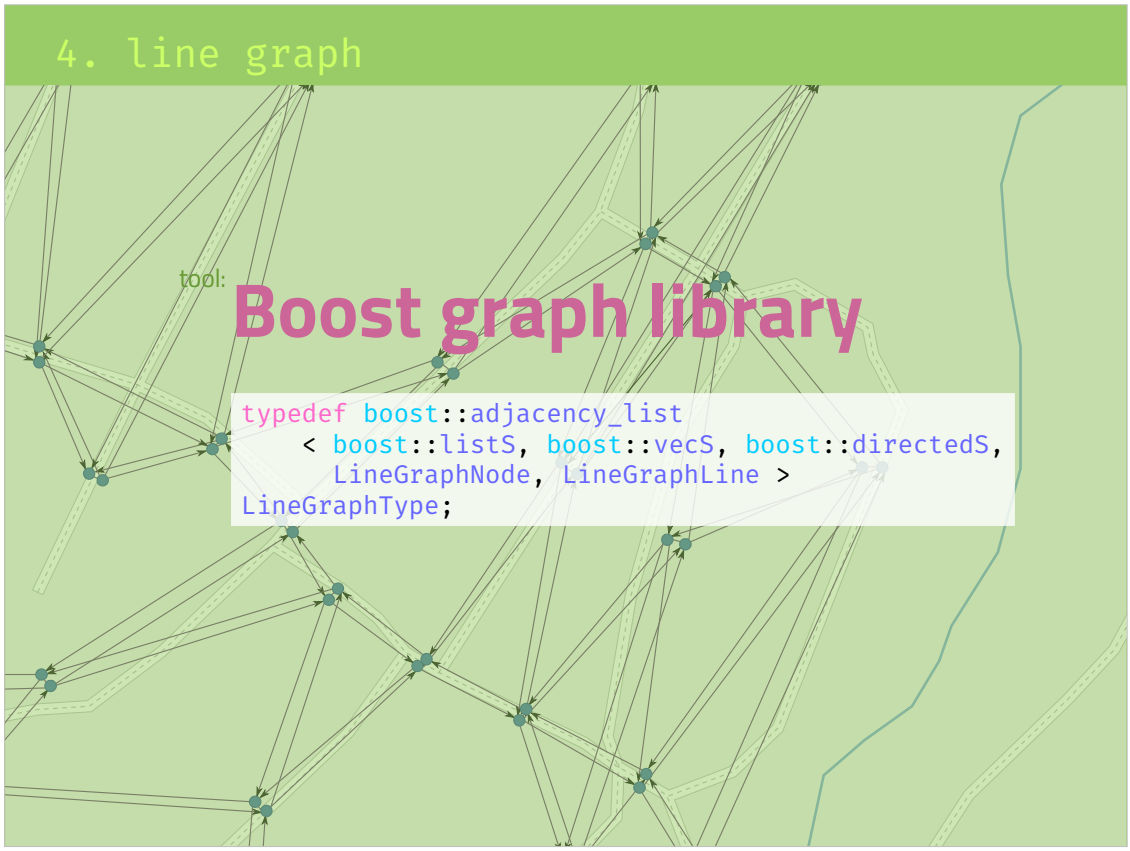
Line graph is not the most efficient technology.



When barrier: no edges → no nodes → no lines



When turn restriction: only affected line = turn disappears.



Module developed in c++

Requirement:
- Return line graph as a BOOST GRAPH data structure.

First time with boost.
- long template expressions,
- typedefs are necessary

Solution:

Some remarks about solution

Preliminary:
Each time a new map is introduced or new roads are built (permanent changes to topology).

Might take some seconds per city.

On demand:
On each request by calling application.

This takes a couple hundred milliseconds.

Configurable:
Topology CAN be built on demand (but is slow).

configurable

json file

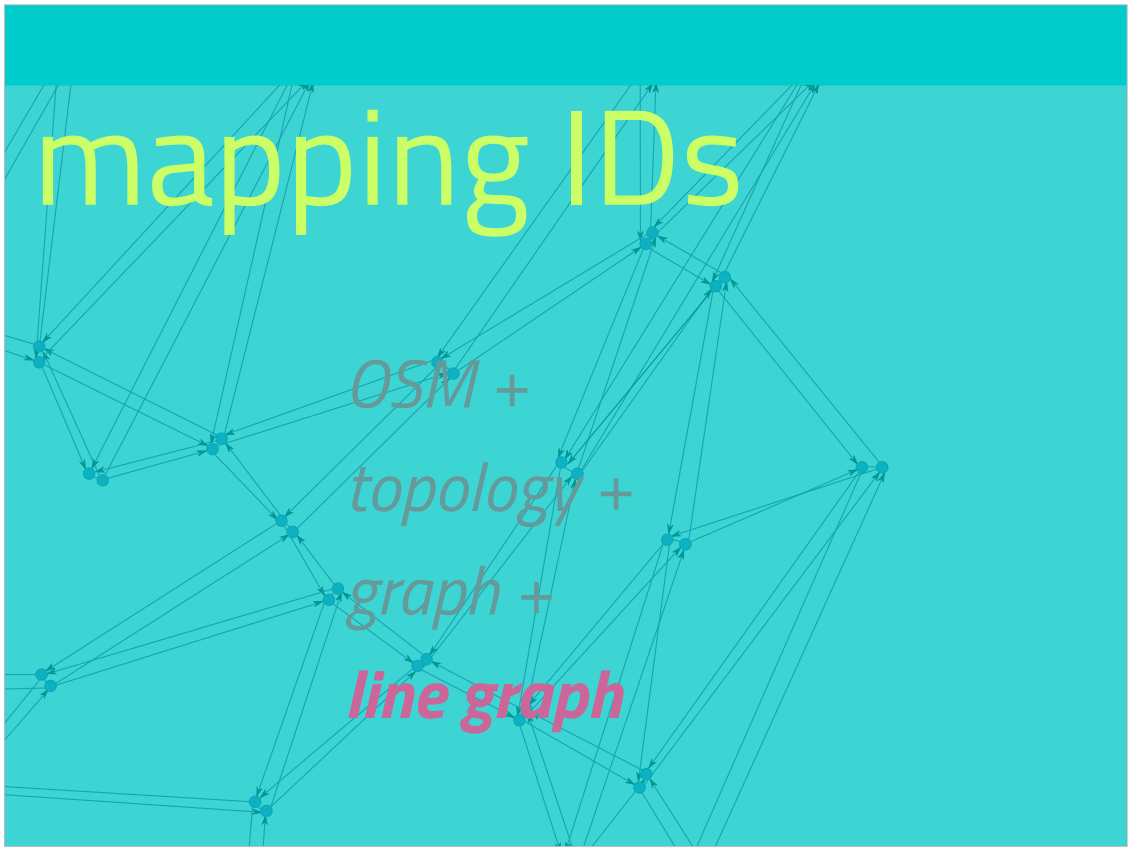
for *settings*

database, vehicle properties,

road speeds, surfaces,

restrictions and costs, ...

Module should be configurable
Takes all settings from a json file



Several layers of IDs.

restrictions

values:

yes, no, permissive, designated, private, discouraged, delivery, customers ...

routing:

one-way (explicit / implicit), lanes ...

transportation mode:

all, foot, vehicle, bicycle, motor_vehicle, motorcycle, motorcar, goods, hgv ...

by use:

psv, car_sharing, emergency, hazmat, disabled ...

dimensions:

max height, weight, width ...

The hardest part in this project.

Not all implemented yet. Needs remodeling.

Hgv = heavy goods vehicle
Psv = Public service vehicle = bus, taxi

conditional restrictions


```
motor_vehicle=no
motor_vehicle:conditional=yes @ (18:30-07:30)
psv=yes
```

Not implemented at all yet.

conditional restrictions

```
maxspeed=none
maxspeed:conditional=
  120 @ (06:00-20:00);
  100 @ (22:00-06:00)
```

Not implemented at all yet.

German highway.

Turning restrictions are RELATIONS,
Handled differently.

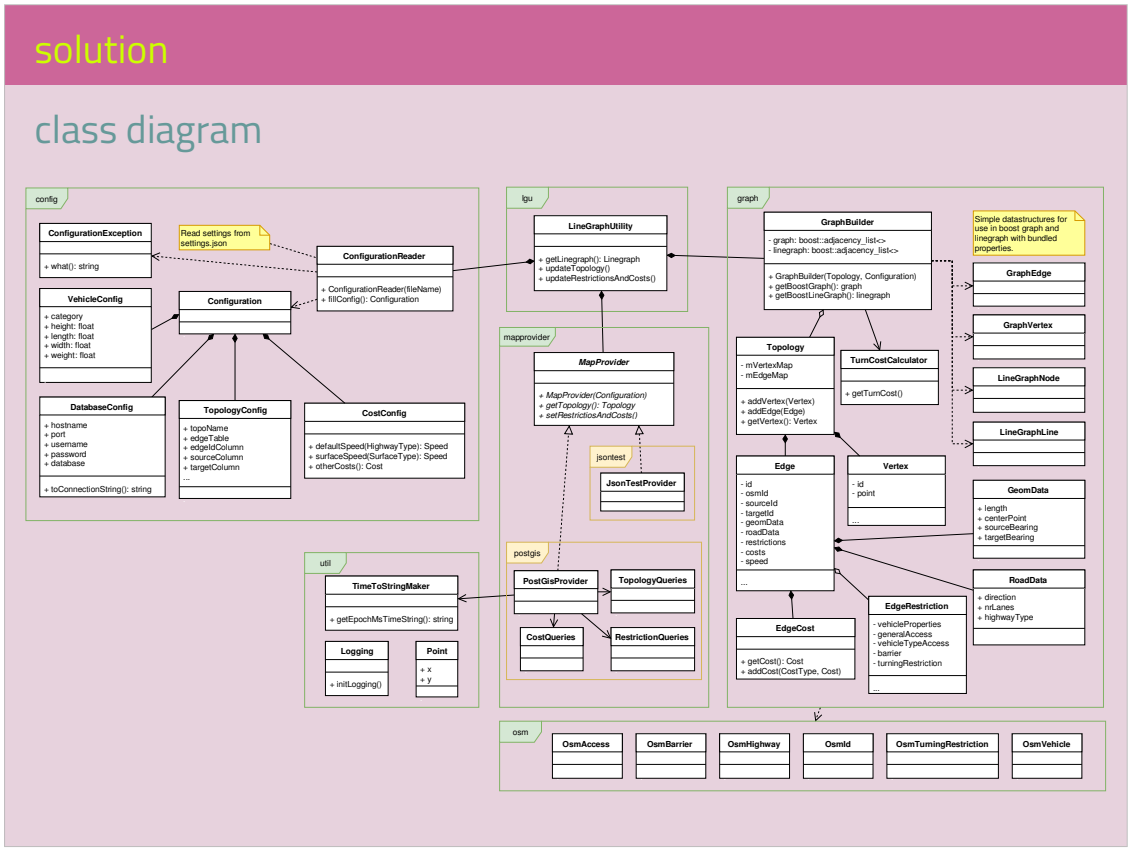
Problem for me as the importer tool
Osm2pgsql did not handle relations well.

Needs some parsing.

FROM edge

VIA vertex OR edge(s).
Difficult to handle edge case since the LGU cannot
does not know enough about the route

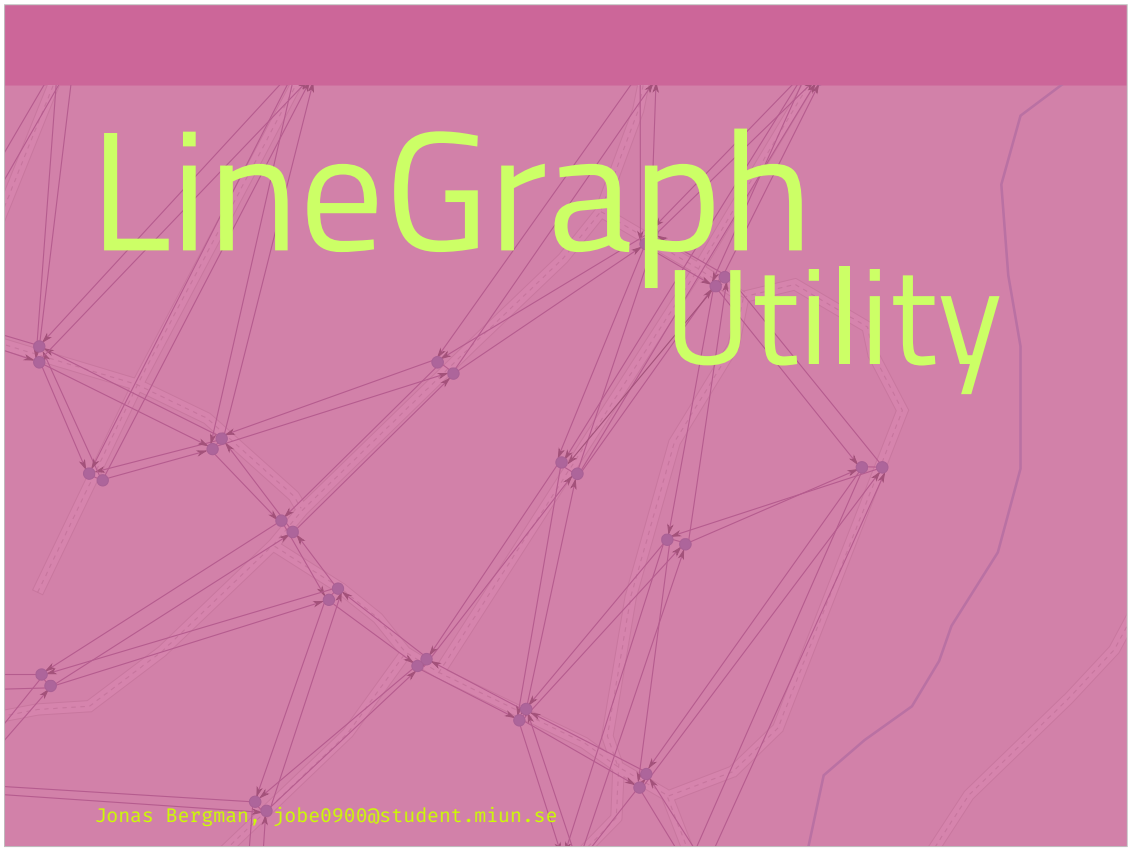
TO edge



Small look at the class diagram to give a view of the solution.

Packages:

- config
- graph
- lgu
- mapprovider
- osm
- util



Have given overview of project:

Requirements, background and remarks about the solution.

Conclusion

OSM is messy = real traffic is messy?

Restrictions are hard. Need to rework them in the LineGraphUtility.