Assignment-6: Routing, GEO-1006

Ming-Chieh Hu, Carmem E. F. Aires, Benthe Kock

November 2024

Project Description

This report was made by:

1. **Ming-Chieh Hu**, student number: 6186416
2. **Carmem E. F. Aires**, student number: 4325893
3. **Benthe Kock**, student number: 6104819

Contents

6.A Why Routing Inside The Database?

Give reasons in favor and against doing routing inside the database.

Reasons in Favor of Routing Inside the Database:

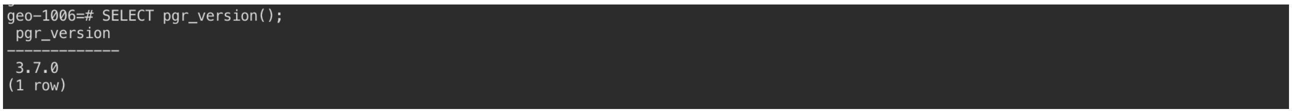
- Data Accessibility: Multiple users can operate at the same time, no need to transfer large datasets between systems.
- Consistency: Keeps calculations in sync with the source data.
- Performance: Databases can leverage powerful indexing and optimization techniques.
- Data Integration: Easily combine routing with other logic stored in the same database.

Reasons Against Routing Inside the Database:

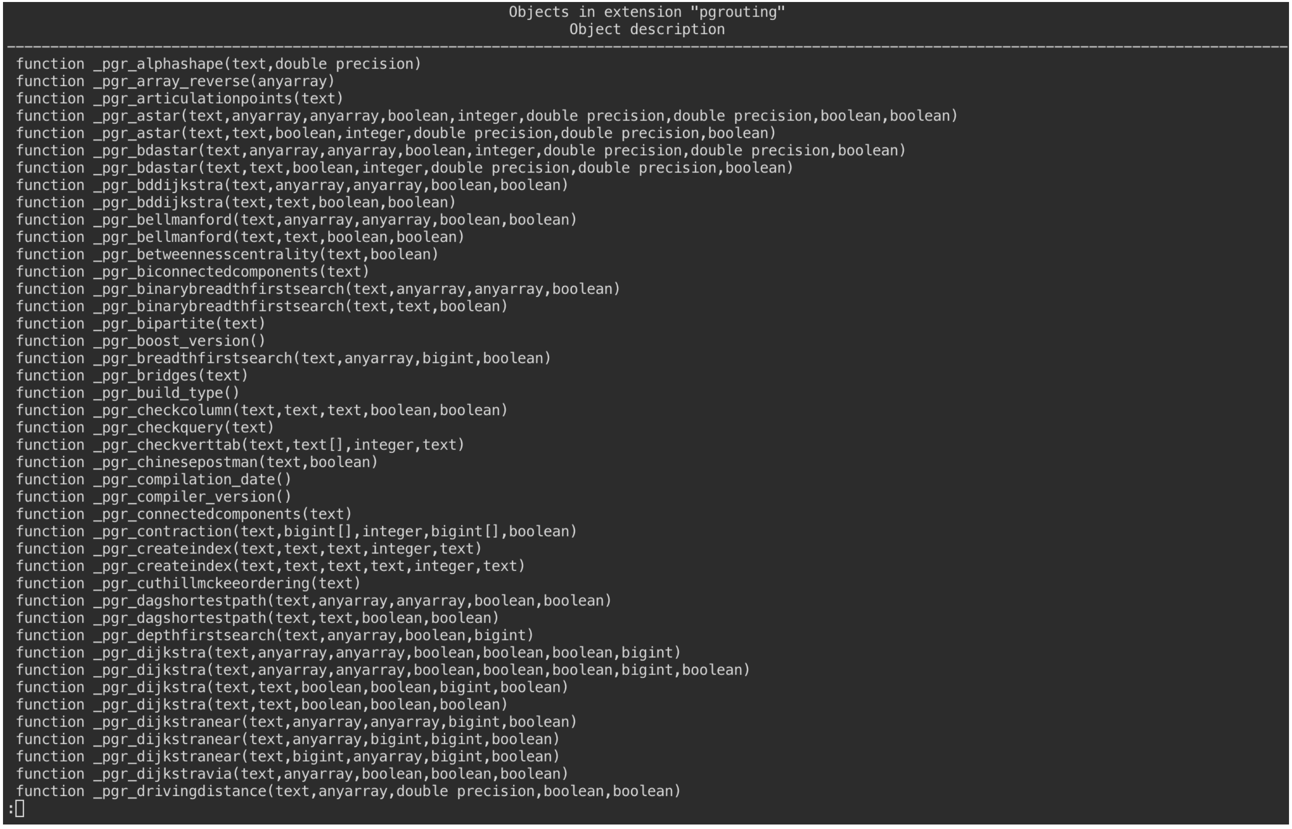
- Complex Setup: It can be tricky to set up the necessary schema and extensions.
- Database Load: Heavy routing calculations can slow down other database operations.
- Scalability: For very large datasets, databases might not scale well with complex graph computations.

6.B Getting Started

6.B.1 First routing functionality in your PostgreSQL database



```
geo-1006=# SELECT pgr_version();
 pgr_version
-------------
 3.7.0
(1 row)
```

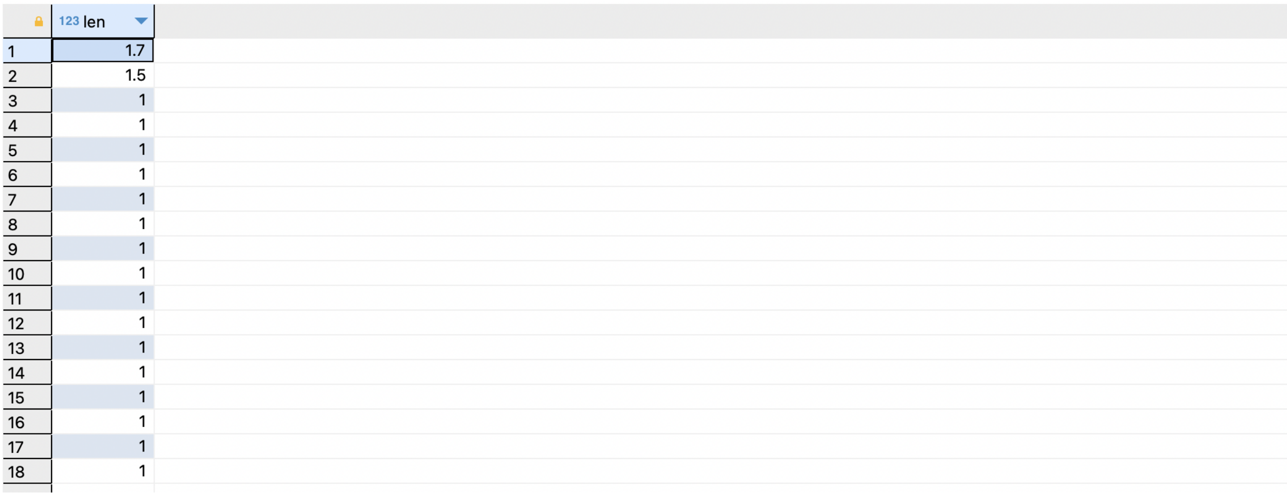


```
                                                                          Objects in extension "pgrouting"
                                                                                 Object description
--------------------------------------------------------------------------------------------------------------------------------------------------------------------
 function _pgr_alphashape(text,double precision)
 function _pgr_array_reverse(anyarray)
 function _pgr_articulationpoints(text)
 function _pgr_astar(text,anyarray,anyarray,boolean,integer,double precision,double precision,boolean,boolean)
 function _pgr_astar(text,text,boolean,integer,double precision,double precision,boolean)
 function _pgr_bdastar(text,anyarray,anyarray,boolean,integer,double precision,double precision,boolean)
 function _pgr_bdastar(text,text,boolean,integer,double precision,double precision,boolean)
 function _pgr_bddijkstra(text,anyarray,anyarray,boolean,boolean)
 function _pgr_bddijkstra(text,text,boolean,boolean)
 function _pgr_bellmanford(text,anyarray,anyarray,boolean,boolean)
 function _pgr_bellmanford(text,text,boolean,boolean)
 function _pgr_betweennesscentrality(text,boolean)
 function _pgr_biconnectedcomponents(text)
 function _pgr_binarybreadthfirstsearch(text,anyarray,anyarray,boolean)
 function _pgr_binarybreadthfirstsearch(text,text,boolean)
 function _pgr_bipartite(text)
 function _pgr_boost_version()
 function _pgr_breadthfirstsearch(text,anyarray,bigint,boolean)
 function _pgr_bridges(text)
 function _pgr_build_type()
 function _pgr_checkcolumn(text,text,text,boolean,boolean)
 function _pgr_checkquery(text)
 function _pgr_checkverttab(text,text[],integer,text)
 function _pgr_chinesepostman(text,boolean)
 function _pgr_compilation_date()
 function _pgr_compiler_version()
 function _pgr_connectedcomponents(text)
 function _pgr_contraction(text,bigint[],integer,bigint[],boolean)
 function _pgr_createindex(text,text,text,integer,text)
 function _pgr_createindex(text,text,text,text,integer,text)
 function _pgr_cuthillmckeeordering(text)
 function _pgr_dagshortestpath(text,anyarray,anyarray,boolean,boolean)
 function _pgr_dagshortestpath(text,text,boolean,boolean)
 function _pgr_depthfirstsearch(text,anyarray,boolean,bigint)
 function _pgr_dijkstra(text,anyarray,anyarray,boolean,boolean,boolean,bigint)
 function _pgr_dijkstra(text,anyarray,anyarray,boolean,boolean,boolean,bigint,boolean)
 function _pgr_dijkstra(text,text,boolean,boolean,bigint,boolean)
 function _pgr_dijkstra(text,text,boolean,boolean,boolean)
 function _pgr_dijkstranear(text,anyarray,anyarray,bigint,boolean)
 function _pgr_dijkstranear(text,anyarray,bigint,bigint,boolean)
 function _pgr_dijkstranear(text,bigint,anyarray,bigint,boolean)
 function _pgr_dijkstravia(text,anyarray,boolean,boolean,boolean)
 function _pgr_drivingdistance(text,anyarray,double precision,boolean,boolean)
:
```

6.B.2 Load some sample data

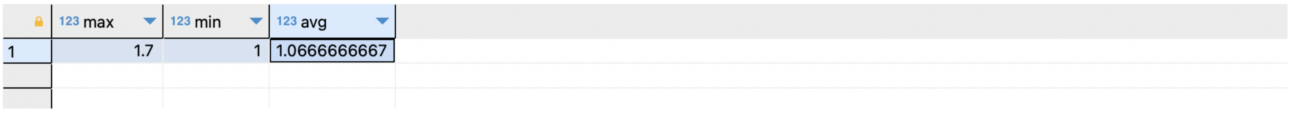
After loading and creating explicit line geometry, check the minimum, maximum and average length of the_geom in the edge table.

```sql
select ST_length(et.the_geom) as len
from edge_table et
order by len desc;
```



	123 len
1	1.7
2	1.5
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1
14	1
15	1
16	1
17	1
18	1

```sql
select max(ST_length(et.the_geom)), min(ST_length(et.the_geom)), avg(ST_length(et.the_geom))
from edge_table et;
```



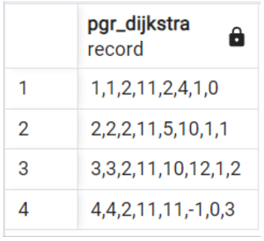
	123 max	123 min	123 avg
1	1.7	1	1.0666666667

max = 1.7, min = 1.0, avg = 1.066...

6.B.3 Check if graph is correct and next do some routing

Double check and analyze the network

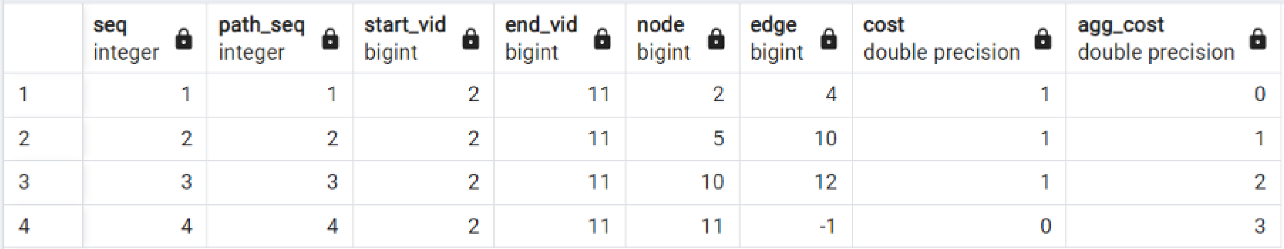
```sql
select pgr_analyzegraph('edge_table', 0.001);
select pgr_dijkstra('SELECT * FROM edge_table', 2, 11);
```



	pgr_dijkstra record
1	1,1,2,11,2,4,1,0
2	2,2,2,11,5,10,1,1
3	3,3,2,11,10,12,1,2
4	4,4,2,11,11,-1,0,3

Routing from node 2 to 11

```sql
SELECT * FROM pgr_dijkstra(
    'SELECT id, source, target, cost, reverse_cost FROM edge_table',2, 11
);
```

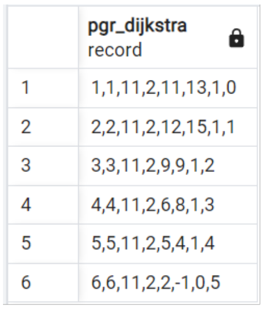


	seq integer	path_seq integer	start_vid bigint	end_vid bigint	node bigint	edge bigint	cost double precision	agg_cost double precision
1	1	1	2	11	2	4	1	0
2	2	2	2	11	5	10	1	1
3	3	3	2	11	10	12	1	2
4	4	4	2	11	11	-1	0	3

Total route cost = 3

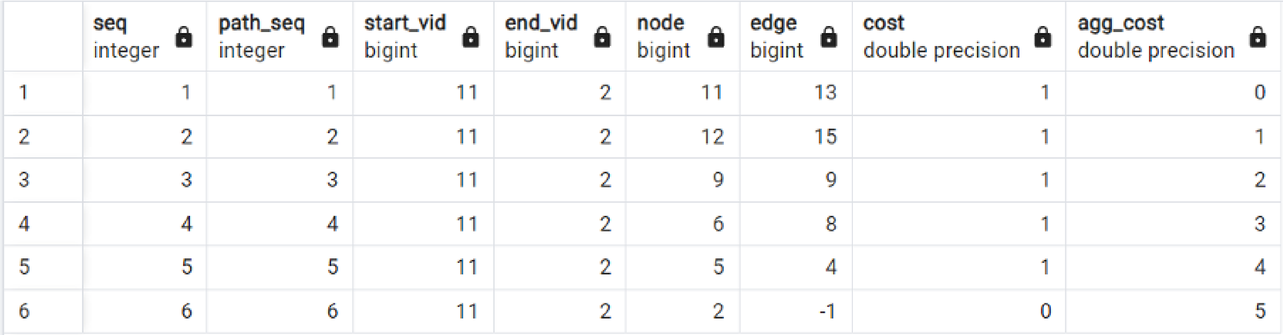
Routing from node 11 to 2

```sql
select pgr_dijkstra('SELECT * FROM edge_table', 11,2);
```



	pgr_dijkstra record
1	1,1,11,2,11,13,1,0
2	2,2,11,2,12,15,1,1
3	3,3,11,2,9,9,1,2
4	4,4,11,2,6,8,1,3
5	5,5,11,2,5,4,1,4
6	6,6,11,2,2,-1,0,5

```sql
SELECT * FROM pgr_dijkstra(
    'SELECT id, source, target, cost, reverse_cost FROM edge_table',11, 2
);
```

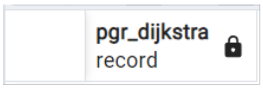


	seq integer	path_seq integer	start_vid bigint	end_vid bigint	node bigint	edge bigint	cost double precision	agg_cost double precision
1	1	1	11	2	11	13	1	0
2	2	2	11	2	12	15	1	1
3	3	3	11	2	9	9	1	2
4	4	4	11	2	6	8	1	3
5	5	5	11	2	5	4	1	4
6	6	6	11	2	2	-1	0	5

Total route cost = 5

Routing from node 1 to 14

```sql
select pgr_dijkstra('SELECT * FROM edge_table', 1,14);
```

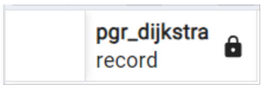


	pgr_dijkstra record

Total route cost = Not connected

Routing from node 1 to 17

```sql
select pgr_dijkstra('SELECT * FROM edge_table', 1,17);
```



	pgr_dijkstra record

Total route cost = Not connected

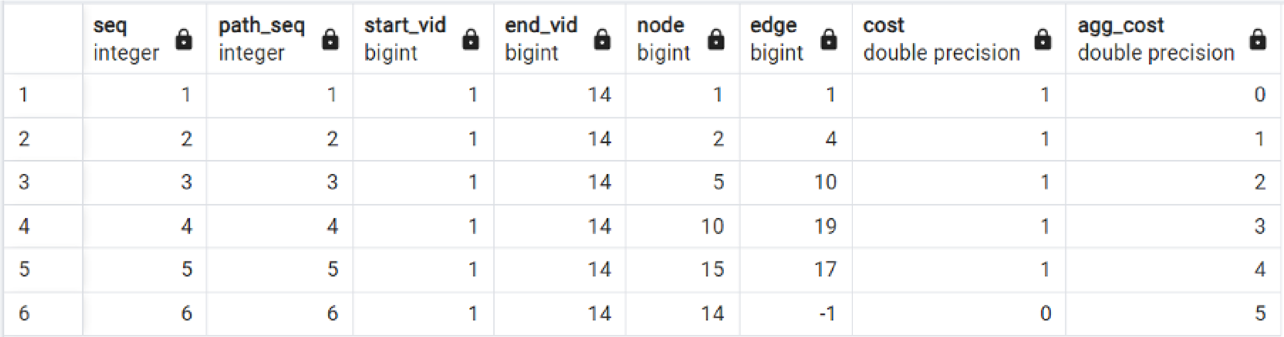
6.B.4 Add some additional edges and nodes to the network

Insert new edges and nodes

```sql
INSERT INTO edge_table (category_id, reverse_category_id, cost, reverse_cost, capacity,
    reverse_capacity, x1, y1, x2, y2)
VALUES
(3, 1, 1, 1, 80, 130, 2,3, 1.999999999999, 3.5),
(3, 1, 1, 1, 80, 130, 2,4, 3.5, 4);

INSERT INTO edge_table (category_id, reverse_category_id, cost, reverse_cost, capacity,
    reverse_capacity, x1, y1, x2, y2)
VALUES
(3, 1, 1, 1, 80, 130, 2, 0, 0, 0),
(3, 1, 1, 1, 80, 130, 0, 1, 0, 0),
(3, 1, 1, 1, 80, 130, 0, 1, 0, 2);
```

Routing from node 1 to 14

```sql
SELECT * FROM pgr_dijkstra(
    'SELECT id, source, target, cost, reverse_cost FROM edge_table', 1, 14
);
```

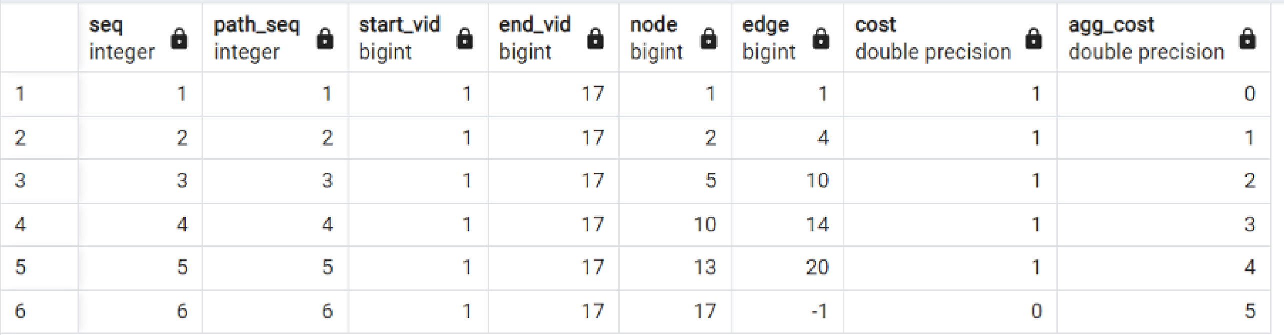


	seq integer 🔒	path_seq integer 🔒	start_vid bigint 🔒	end_vid bigint 🔒	node bigint 🔒	edge bigint 🔒	cost double precision 🔒	agg_cost double precision 🔒
1	1	1	1	14	1	1	1	0
2	2	2	1	14	2	4	1	1
3	3	3	1	14	5	10	1	2
4	4	4	1	14	10	19	1	3
5	5	5	1	14	15	17	1	4
6	6	6	1	14	14	-1	0	5

Total route cost = 5

Routing from node 1 to 17

```sql
SELECT * FROM pgr_dijkstra(
    'SELECT id, source, target, cost, reverse_cost FROM edge_table', 1, 17
);
```

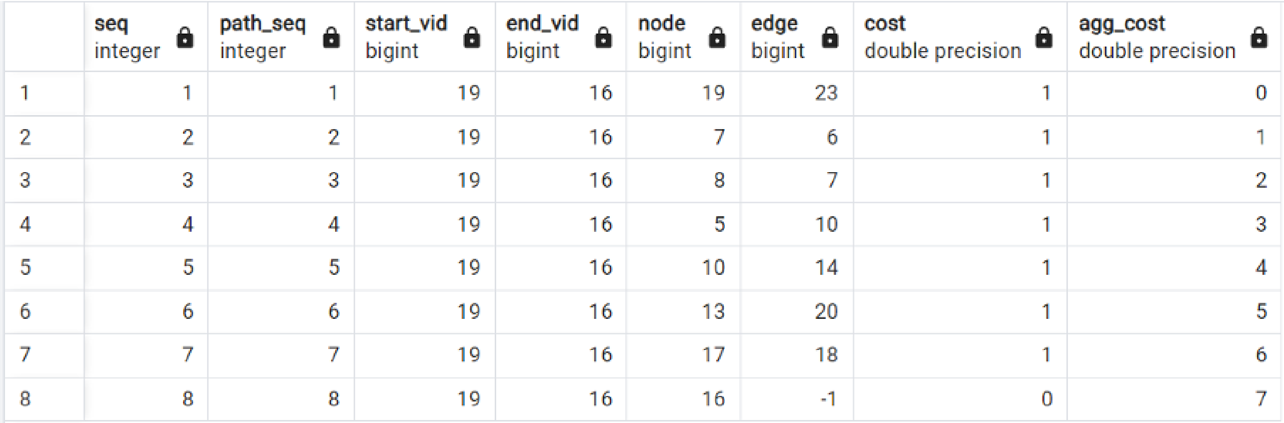


	seq integer 🔒	path_seq integer 🔒	start_vid bigint 🔒	end_vid bigint 🔒	node bigint 🔒	edge bigint 🔒	cost double precision 🔒	agg_cost double precision 🔒
1	1	1	1	17	1	1	1	0
2	2	2	1	17	2	4	1	1
3	3	3	1	17	5	10	1	2
4	4	4	1	17	10	14	1	3
5	5	5	1	17	13	20	1	4
6	6	6	1	17	17	-1	0	5

Total route cost = 5

Routing from node 19 to 16

```sql
SELECT * FROM pgr_dijkstra(
    'SELECT id, source, target, cost, reverse_cost FROM edge_table', 19, 16
);
```

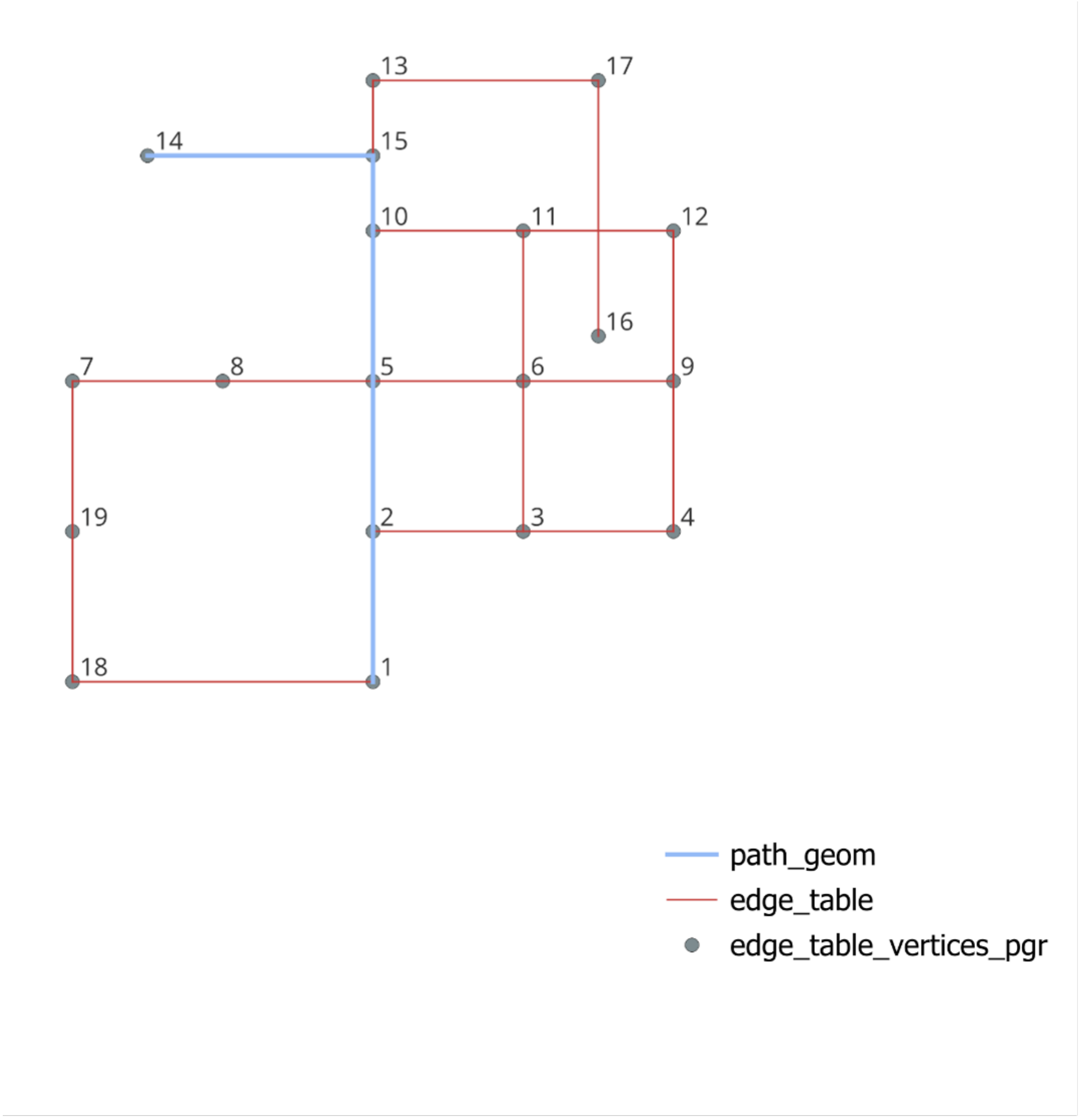


	seq integer 🔒	path_seq integer 🔒	start_vid bigint 🔒	end_vid bigint 🔒	node bigint 🔒	edge bigint 🔒	cost double precision 🔒	agg_cost double precision 🔒
1	1	1	19	16	19	23	1	0
2	2	2	19	16	7	6	1	1
3	3	3	19	16	8	7	1	2
4	4	4	19	16	5	10	1	3
5	5	5	19	16	10	14	1	4
6	6	6	19	16	13	20	1	5
7	7	7	19	16	17	18	1	6
8	8	8	19	16	16	-1	0	7

Total route cost = 7

6.B.5 Visualize in QGIS the network

```
CREATE TABLE computed_path AS
SELECT * FROM pgr_dijkstra(
    'SELECT id, source, target, cost, reverse_cost FROM edge_table',
    1,
    14
);

CREATE VIEW path_geom AS
SELECT
    edge_table.id,
    edge_table.the_geom
FROM
    edge_table
INNER JOIN
    computed_path ON edge_table.id = computed_path.edge;
```



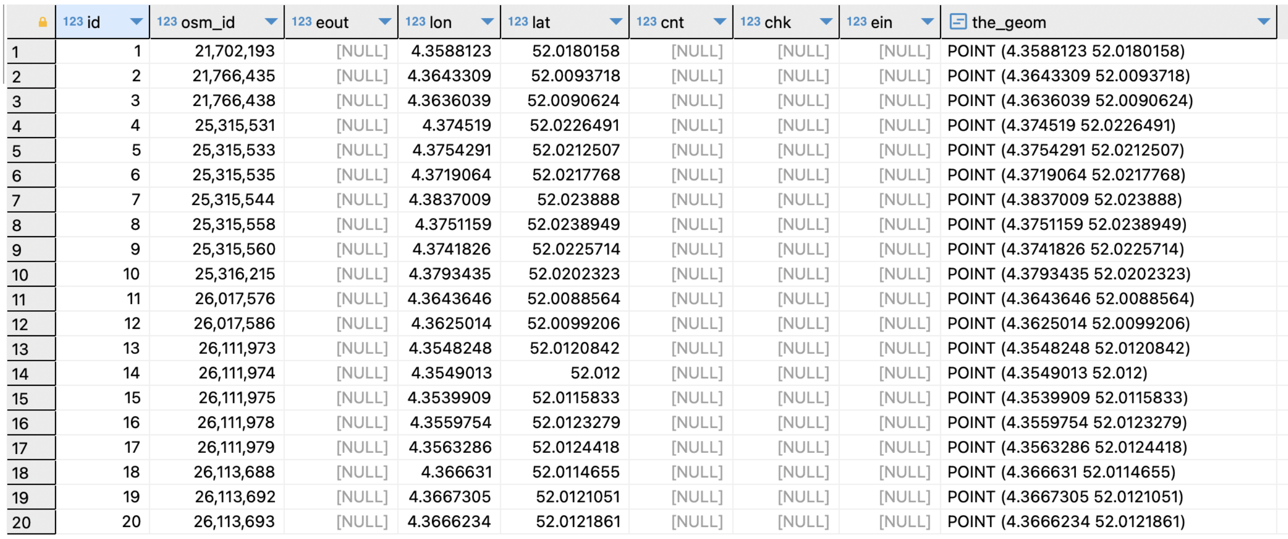
6.C Routing In Delft

6.C.1 Start with new database, load data from OSM

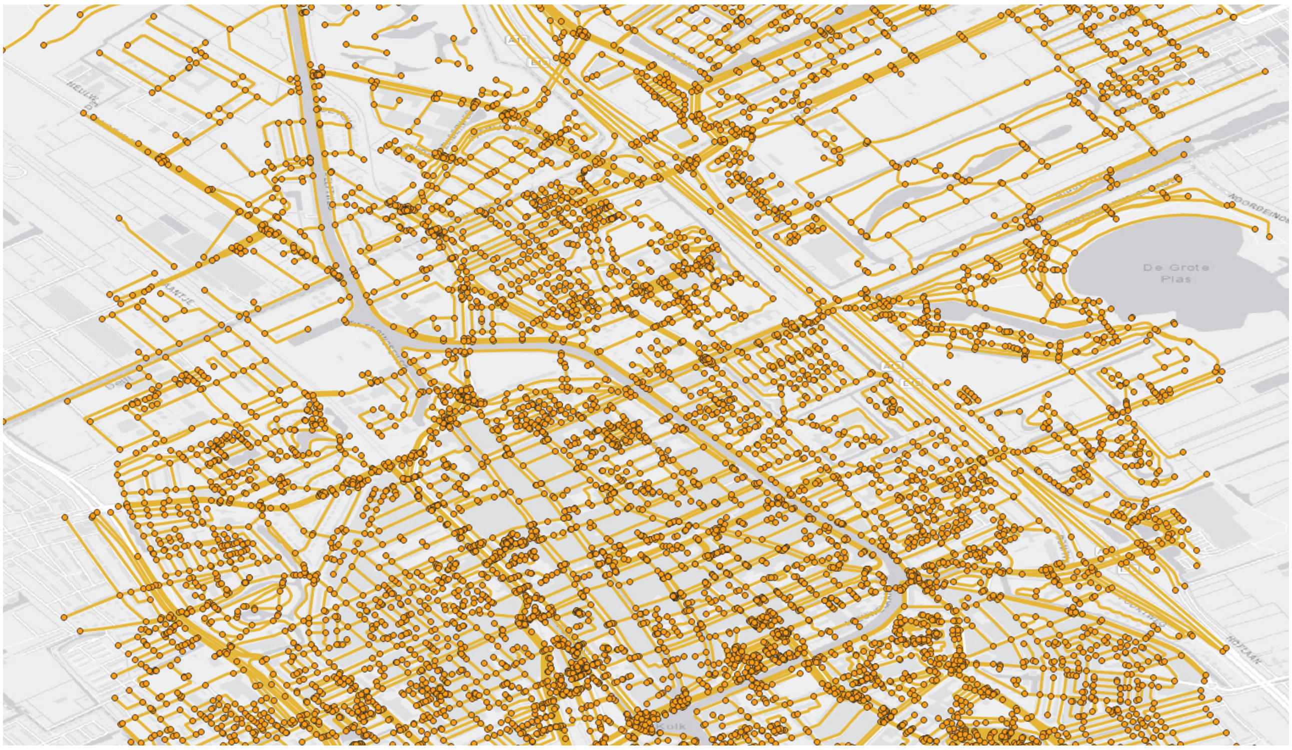
```
Export Ways ...
    Processing 38156 ways:
[**************************|                                              ] (52%) Total processed: 20000       Vertices inserted: 6131        Split ways inserted 6742
[*************************************************************************|] (100%) Total processed: 38156      Vertices inserted: 3378        Split ways inserted 5861

Creating indexes ...

Processing Points of Interest ...
#########################
size of streets: 38156
#########################
```

6.C.2 Visualize in QGIS

```sql
select *
from ways_vertices_pgr;
```



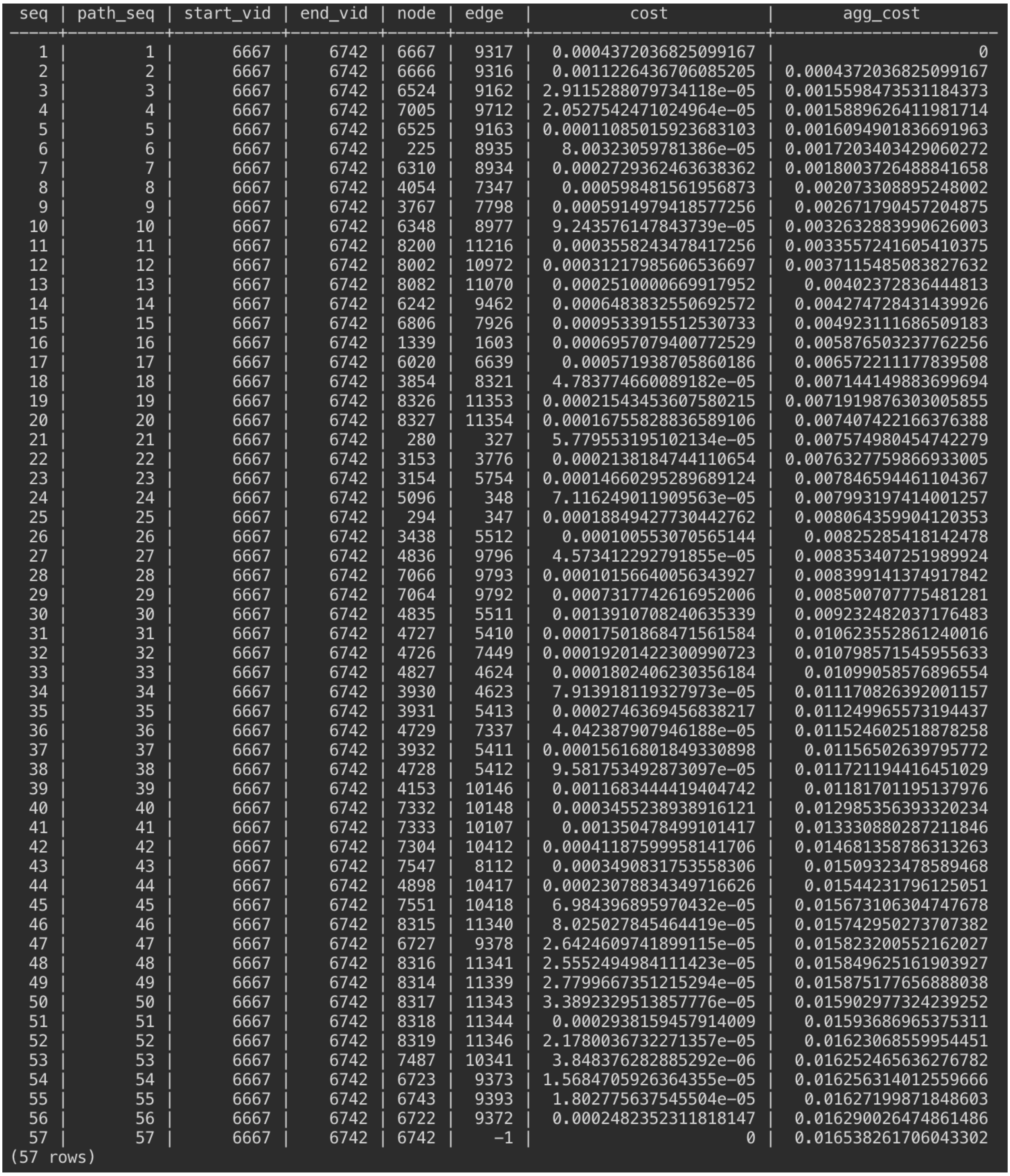
	id	osm_id	eout	lon	lat	cnt	chk	ein	the_geom
1	1	21,702,193	[NULL]	4.3588123	52.0180158	[NULL]	[NULL]	[NULL]	POINT (4.3588123 52.0180158)
2	2	21,766,435	[NULL]	4.3643309	52.0093718	[NULL]	[NULL]	[NULL]	POINT (4.3643309 52.0093718)
3	3	21,766,438	[NULL]	4.3636039	52.0090624	[NULL]	[NULL]	[NULL]	POINT (4.3636039 52.0090624)
4	4	25,315,531	[NULL]	4.374519	52.0226491	[NULL]	[NULL]	[NULL]	POINT (4.374519 52.0226491)
5	5	25,315,533	[NULL]	4.3754291	52.0212507	[NULL]	[NULL]	[NULL]	POINT (4.3754291 52.0212507)
6	6	25,315,535	[NULL]	4.3719064	52.0217768	[NULL]	[NULL]	[NULL]	POINT (4.3719064 52.0217768)
7	7	25,315,544	[NULL]	4.3837009	52.023888	[NULL]	[NULL]	[NULL]	POINT (4.3837009 52.023888)
8	8	25,315,558	[NULL]	4.3751159	52.0238949	[NULL]	[NULL]	[NULL]	POINT (4.3751159 52.0238949)
9	9	25,315,560	[NULL]	4.3741826	52.0225714	[NULL]	[NULL]	[NULL]	POINT (4.3741826 52.0225714)
10	10	25,316,215	[NULL]	4.3793435	52.0202323	[NULL]	[NULL]	[NULL]	POINT (4.3793435 52.0202323)
11	11	26,017,576	[NULL]	4.3643646	52.0088564	[NULL]	[NULL]	[NULL]	POINT (4.3643646 52.0088564)
12	12	26,017,586	[NULL]	4.3625014	52.0099206	[NULL]	[NULL]	[NULL]	POINT (4.3625014 52.0099206)
13	13	26,111,973	[NULL]	4.3548248	52.0120842	[NULL]	[NULL]	[NULL]	POINT (4.3548248 52.0120842)
14	14	26,111,974	[NULL]	4.3549013	52.012	[NULL]	[NULL]	[NULL]	POINT (4.3549013 52.012)
15	15	26,111,975	[NULL]	4.3539909	52.0115833	[NULL]	[NULL]	[NULL]	POINT (4.3539909 52.0115833)
16	16	26,111,978	[NULL]	4.3559754	52.0123279	[NULL]	[NULL]	[NULL]	POINT (4.3559754 52.0123279)
17	17	26,111,979	[NULL]	4.3563286	52.0124418	[NULL]	[NULL]	[NULL]	POINT (4.3563286 52.0124418)
18	18	26,113,688	[NULL]	4.366631	52.0114655	[NULL]	[NULL]	[NULL]	POINT (4.366631 52.0114655)
19	19	26,113,692	[NULL]	4.3667305	52.0121051	[NULL]	[NULL]	[NULL]	POINT (4.3667305 52.0121051)
20	20	26,113,693	[NULL]	4.3666234	52.0121861	[NULL]	[NULL]	[NULL]	POINT (4.3666234 52.0121861)



6.C.3 Routing in Delft

Dijkstra path query

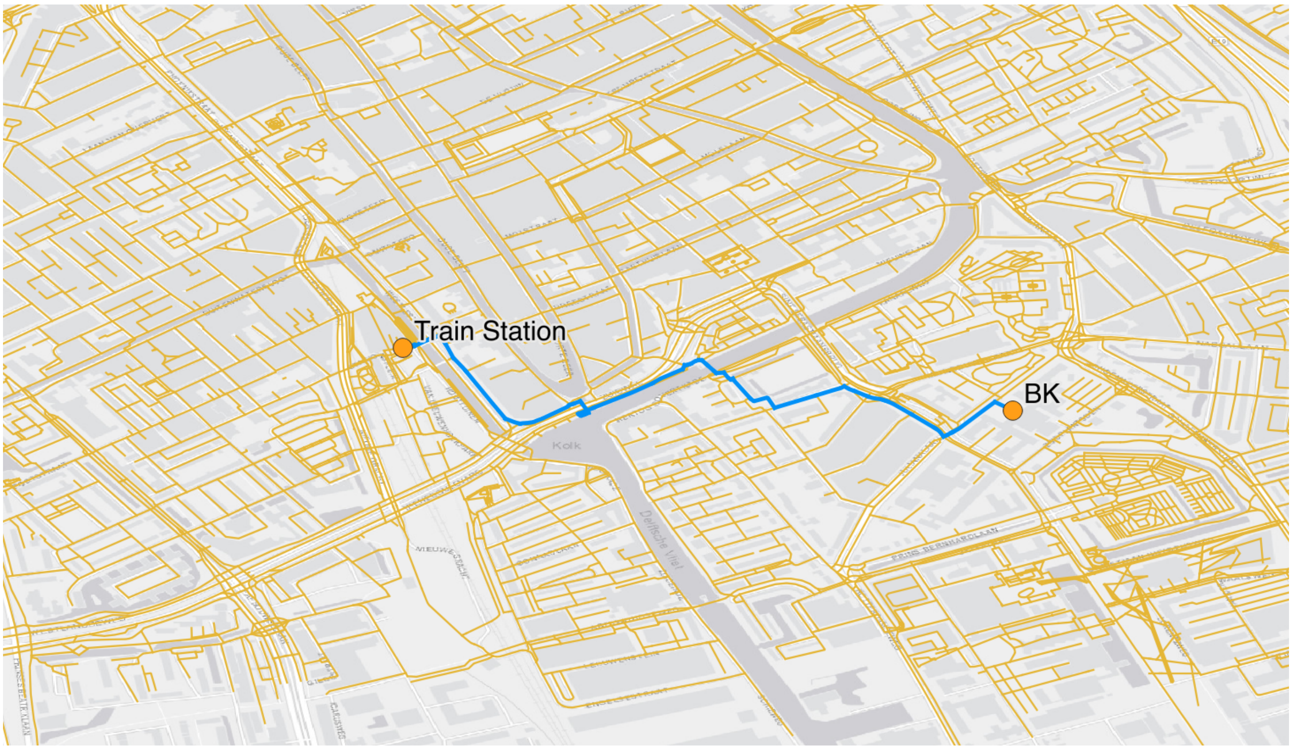
```
select *
from pgr_dijkstra(
    'select gid as id, source, target, cost, reverse_cost from ways',
    6667,
    6742,
    true
);
```



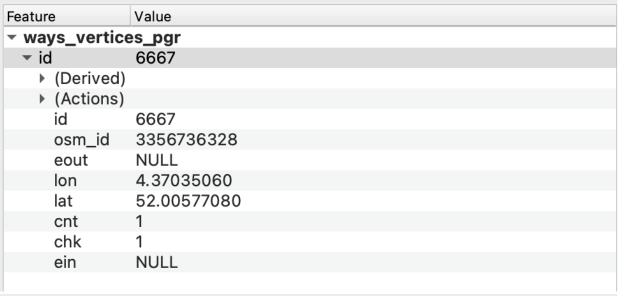
```
 seq | path_seq | start_vid | end_vid | node | edge  |          cost          |       agg_cost
-----+----------+-----------+---------+------+-------+------------------------+-----------------------
   1 |        1 |      6667 |    6742 | 6667 |  9317 |  0.0004372036825099167 |                     0
   2 |        2 |      6667 |    6742 | 6666 |  9316 |  0.0011226436706085205 | 0.0004372036825099167
   3 |        3 |      6667 |    6742 | 6524 |  9162 | 2.9115288079734118e-05 | 0.0015598473531184373
   4 |        4 |      6667 |    6742 | 7005 |  9712 | 2.0527542471024964e-05 | 0.0015889626411981714
   5 |        5 |      6667 |    6742 | 6525 |  9163 | 0.00011085015923683103 | 0.0016094901836691963
   6 |        6 |      6667 |    6742 |  225 |  8935 |   8.00323059781386e-05 | 0.0017203403429060272
   7 |        7 |      6667 |    6742 | 6310 |  8934 |  0.0002729362463638362 | 0.0018003726488841658
   8 |        8 |      6667 |    6742 | 4054 |  7347 |   0.000598481561956873 |  0.002073308895248002
   9 |        9 |      6667 |    6742 | 3767 |  7798 |  0.0005914979418577256 |  0.002671790457204875
  10 |       10 |      6667 |    6742 | 6348 |  8977 |  9.243576147843739e-05 | 0.0032632883990626003
  11 |       11 |      6667 |    6742 | 8200 | 11216 |  0.0003558243478417256 | 0.0033557241605410375
  12 |       12 |      6667 |    6742 | 8002 | 10972 | 0.00031217985606536697 | 0.0037115485083827632
  13 |       13 |      6667 |    6742 | 8082 | 11070 |  0.0002510000669917952 |   0.00402372836444813
  14 |       14 |      6667 |    6742 | 6242 |  9462 |  0.0006483832550692572 |  0.004274728431439926
  15 |       15 |      6667 |    6742 | 6806 |  7926 |  0.0009533915512530733 |  0.004923111686509183
  16 |       16 |      6667 |    6742 | 1339 |  1603 |  0.0006957079400772529 |  0.005876503237762256
  17 |       17 |      6667 |    6742 | 6020 |  6639 |   0.000571938705860186 |  0.006572211177839508
  18 |       18 |      6667 |    6742 | 3854 |  8321 |  4.783774660089182e-05 |  0.007144149883699694
  19 |       19 |      6667 |    6742 | 8326 | 11353 | 0.00021543453607580215 | 0.0071919876303005855
  20 |       20 |      6667 |    6742 | 8327 | 11354 | 0.00016755828836589106 |  0.007407422166376388
  21 |       21 |      6667 |    6742 |  280 |   327 |  5.779553195102134e-05 |  0.007574980454742279
  22 |       22 |      6667 |    6742 | 3153 |  3776 |  0.0002138184744110654 | 0.0076327759866933005
  23 |       23 |      6667 |    6742 | 3154 |  5754 | 0.00014660295289689124 |  0.007846594461104367
  24 |       24 |      6667 |    6742 | 5096 |   348 |  7.116249011909563e-05 |  0.007993197414001257
  25 |       25 |      6667 |    6742 |  294 |   347 | 0.00018849427730442762 |  0.008064359904120353
  26 |       26 |      6667 |    6742 | 3438 |  5512 |   0.000100553070565144 |   0.00825285418142478
  27 |       27 |      6667 |    6742 | 4836 |  9796 |  4.573412292791855e-05 |  0.008353407251989924
  28 |       28 |      6667 |    6742 | 7066 |  9793 | 0.00010156640056343927 |  0.008399141374917842
  29 |       29 |      6667 |    6742 | 7064 |  9792 |  0.0007317742616952006 |  0.008500707775481281
  30 |       30 |      6667 |    6742 | 4835 |  5511 |  0.0013910708240635339 |  0.009232482037176483
  31 |       31 |      6667 |    6742 | 4727 |  5410 | 0.00017501868471561584 |  0.010623552861240016
  32 |       32 |      6667 |    6742 | 4726 |  7449 | 0.00019201422300990723 |  0.010798571545955633
  33 |       33 |      6667 |    6742 | 4827 |  4624 |  0.0001802406230356184 |   0.01099058576896554
  34 |       34 |      6667 |    6742 | 3930 |  4623 |  7.913918119327973e-05 |  0.011170826392001157
  35 |       35 |      6667 |    6742 | 3931 |  5413 |  0.0002746369456838217 |  0.011249965573194437
  36 |       36 |      6667 |    6742 | 4729 |  7337 |  4.042387907946188e-05 |  0.011524602518878258
  37 |       37 |      6667 |    6742 | 3932 |  5411 | 0.00015616801849330898 |   0.01156502639795772
  38 |       38 |      6667 |    6742 | 4728 |  5412 |  9.581753492873097e-05 |  0.011721194416451029
  39 |       39 |      6667 |    6742 | 4153 | 10146 |  0.0011683444419404742 |   0.01181701195137976
  40 |       40 |      6667 |    6742 | 7332 | 10148 |  0.0003455238938916121 |  0.012985356393320234
  41 |       41 |      6667 |    6742 | 7333 | 10107 |   0.001350478499101417 |  0.013330880287211846
  42 |       42 |      6667 |    6742 | 7304 | 10412 | 0.00041187599958141706 |  0.014681358786313263
  43 |       43 |      6667 |    6742 | 7547 |  8112 |  0.0003490831753558306 |   0.01509323478589468
  44 |       44 |      6667 |    6742 | 4898 | 10417 | 0.00023078834349716626 |   0.01544231796125051
  45 |       45 |      6667 |    6742 | 7551 | 10418 |  6.984396895970432e-05 |  0.015673106304747678
  46 |       46 |      6667 |    6742 | 8315 | 11340 |  8.025027845464419e-05 |  0.015742950273707382
  47 |       47 |      6667 |    6742 | 6727 |  9378 | 2.6424609741899115e-05 |  0.015823200552162027
  48 |       48 |      6667 |    6742 | 8316 | 11341 | 2.5552494984111423e-05 |  0.015849625161903927
  49 |       49 |      6667 |    6742 | 8314 | 11339 | 2.7799667351215294e-05 |  0.015875177656888038
  50 |       50 |      6667 |    6742 | 8317 | 11343 | 3.3892329513857776e-05 |  0.015902977324239252
  51 |       51 |      6667 |    6742 | 8318 | 11344 |  0.0002938159457914009 |   0.01593686965375311
  52 |       52 |      6667 |    6742 | 8319 | 11346 | 2.1780036732271357e-05 |   0.01623068559954451
  53 |       53 |      6667 |    6742 | 7487 | 10341 |  3.848376282885292e-06 |  0.016252465636276782
  54 |       54 |      6667 |    6742 | 6723 |  9373 | 1.5684705926364355e-05 |  0.016256314012559666
  55 |       55 |      6667 |    6742 | 6743 |  9393 |  1.802775637545504e-05 |   0.01627199871848603
  56 |       56 |      6667 |    6742 | 6722 |  9372 |  0.0002482352311818147 |  0.016290026474861486
  57 |       57 |      6667 |    6742 | 6742 |    -1 |                      0 |  0.016538261706043302
(57 rows)
```

Visualzation in QGIS

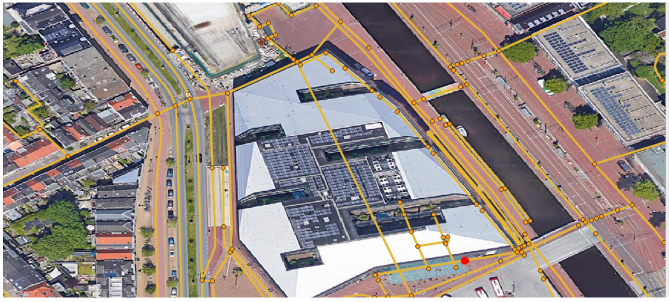
```sql
create table route as (
    with dijkstra as (
        select * from pgr_dijkstra('select gid as id, source, target, cost, reverse_cost from ways', 6667, 6742, true)
    )
    select d.*, w.gid, w.the_geom
    from ways w join dijkstra d on d.edge = w.gid
);
```

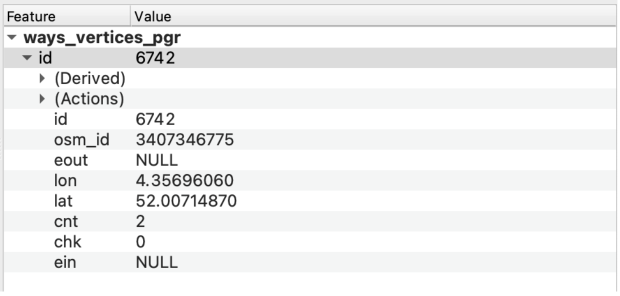


Starting and ending node



Feature	Value
ways_vertices_pgr	
id	6667
(Derived)	
(Actions)	
id	6667
osm_id	3356736328
eout	NULL
lon	4.37035060
lat	52.00577080
cnt	1
chk	1
ein	NULL





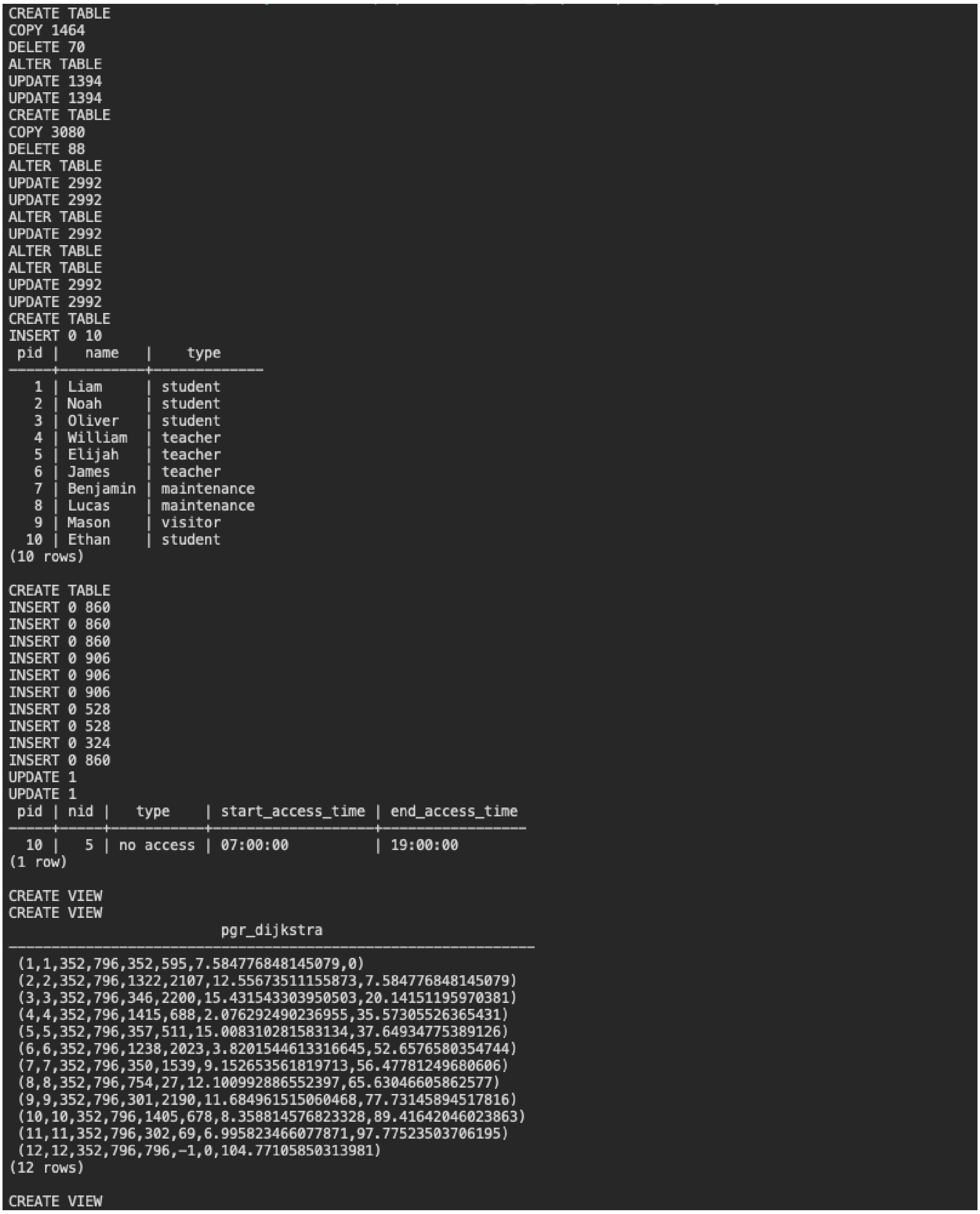
Feature	Value
ways_vertices_pgr	
id	6742
(Derived)	
(Actions)	
id	6742
osm_id	3407346775
eout	NULL
lon	4.35696060
lat	52.00714870
cnt	2
chk	0
ein	NULL

6.D Indoor Routing In BK

6.D.1 Start with new database for 3D indoor routing

```sql
create extension postgis;
create extension pgrouting;
```

6.D.2 Load data from our BK building



```
CREATE TABLE
COPY 1464
DELETE 70
ALTER TABLE
UPDATE 1394
UPDATE 1394
CREATE TABLE
COPY 3080
DELETE 88
ALTER TABLE
UPDATE 2992
UPDATE 2992
ALTER TABLE
UPDATE 2992
ALTER TABLE
ALTER TABLE
UPDATE 2992
UPDATE 2992
CREATE TABLE
INSERT 0 10
 pid |   name   |    type
-----+----------+-------------
   1 | Liam     | student
   2 | Noah     | student
   3 | Oliver   | student
   4 | William  | teacher
   5 | Elijah   | teacher
   6 | James    | teacher
   7 | Benjamin | maintenance
   8 | Lucas    | maintenance
   9 | Mason    | visitor
  10 | Ethan    | student
(10 rows)

CREATE TABLE
INSERT 0 860
INSERT 0 860
INSERT 0 860
INSERT 0 906
INSERT 0 906
INSERT 0 906
INSERT 0 528
INSERT 0 528
INSERT 0 324
INSERT 0 860
UPDATE 1
UPDATE 1
 pid | nid |   type    | start_access_time | end_access_time
-----+-----+-----------+-------------------+-----------------
  10 |   5 | no access | 07:00:00          | 19:00:00
(1 row)

CREATE VIEW
CREATE VIEW
                         pgr_dijkstra
---------------------------------------------------------------
 (1,1,352,796,352,595,7.584776848145079,0)
 (2,2,352,796,1322,2107,12.55673511155873,7.584776848145079)
 (3,3,352,796,346,2200,15.431543303950503,20.14151195970381)
 (4,4,352,796,1415,688,2.076292490236955,35.57305526365431)
 (5,5,352,796,357,511,15.008310281583134,37.64934775389126)
 (6,6,352,796,1238,2023,3.8201544613316645,52.6576580354744)
 (7,7,352,796,350,1539,9.152653561819713,56.47781249680606)
 (8,8,352,796,754,27,12.100992886552397,65.63046605862577)
 (9,9,352,796,301,2190,11.684961515060468,77.73145894517816)
 (10,10,352,796,1405,678,8.358814576823328,89.41642046023863)
 (11,11,352,796,302,69,6.995823466077871,97.77523503706195)
 (12,12,352,796,796,-1,0,104.77105850313981)
(12 rows)

CREATE VIEW
```

Make a second view for another user (one of the teachers)

```sql
-- Create node and edge view for teacher Elijah
create view node_vw_2 as select node.id, node.geom from node, rights, party
where party.name= 'Elijah' and rights.pid=party.pid and rights.type= 'access' and rights.nid=node.id;

create view edge_vw_2 as select edge.*  from edge, node_vw_2 nf, node_vw_2 nt
where nf.id=edge.source and nt.id=edge.target;
```

Find a source and destination pair for which the route of the student is different than the route of the teacher.

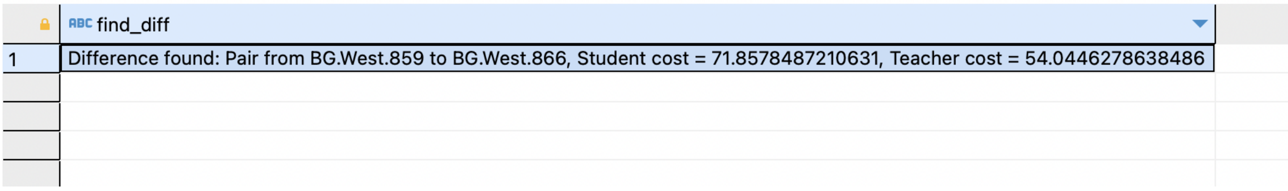
```sql
-- Which student's route differs from the teacher's?
-- Too slow to find one difference without using loop and function
-- Use pgr_dijkstraCost() here to reduce computation
-- Definition of view possible_pairs is below this code block

create or replace function find_diff()
returns text as $$
declare
    pair record;
    student_cost numeric;
    teacher_cost numeric;
begin
    -- Loop through each pair in the possible_pairs view
    for pair in
        select pair_from, pair_to from possible_pairs
    loop
        -- Get the Dijkstra cost for the student
        select agg_cost into student_cost
        from pgr_dijkstraCost(
            'select id, source, target, cost from edge_vw',
            (select source from edge_vw where fromnode = pair.pair_from limit 1),
            (select target from edge_vw where tonode = pair.pair_to limit 1),
            false
        );

        -- Get the Dijkstra cost for the teacher
        select agg_cost into teacher_cost
        from pgr_dijkstraCost(
            'select id, source, target, cost from edge_vw_2',
            (select source from edge_vw_2 where fromnode = pair.pair_from limit 1),
            (select target from edge_vw_2 where tonode = pair.pair_to limit 1),
            false
        );

        -- Check if the costs differ
        if student_cost <> teacher_cost then
            return format(
                'Difference found: Pair from %s to %s, Student cost = %s, Teacher cost = %s',
                pair.pair_from, pair.pair_to, student_cost, teacher_cost
            );
        end if;
    end loop;

    -- If no differences are found
    return 'No differences found.';
end;
$$ language plpgsql;
```

View *possible_pairs* definition

```sql
-- Generate all possible pairs of fromnode and tonode
create or replace view possible_pairs as (
    select
        a.fromnode as pair_from,
        b.fromnode as pair_to
    from edge_vw a, edge_vw b
    where a.fromnode <> b.fromnode and a.fromnode not like 'D%' and b.fromnode not like 'D%'
);
```

Run *find_diff()* function

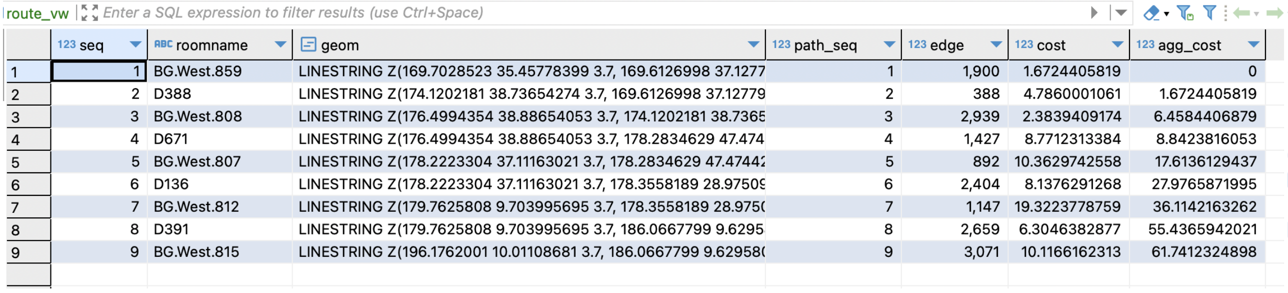
```sql
-- Now it's easy, the loop stops at first result so it's way faster:)
select find_diff();
```



	find_diff
1	Difference found: Pair from BG.West.859 to BG.West.866, Student cost = 71.8578487210631, Teacher cost = 54.0446278638486

Create view for route

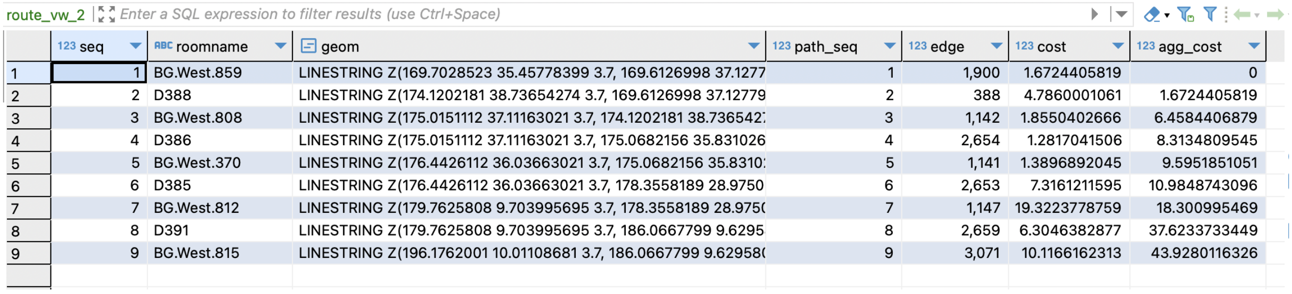
```sql
-- Liam's route (student)
create or replace view route_vw as
select distinct X.seq, Y.roomname, Z.geom, X.Path_seq, X.edge, X.cost, X.agg_cost
from
    pgr_dijkstra(
        'select id, source, target, cost from edge_vw',
         (select source from edge_vw where fromnode ='BG.West.859' limit 1),
         (select target from edge_vw where tonode ='BG.West.866' limit 1),
         false
    ) X
    join node as Y on X.node = Y.id
    join edge as Z on X.edge = Z.id
order by seq;
```



route_vw | Enter a SQL expression to filter results (use Ctrl+Space)

	seq	roomname	geom	path_seq	edge	cost	agg_cost
1	1	BG.West.859	LINESTRING Z(169.7028523 35.45778399 3.7, 169.6126998 37.1277	1	1,900	1.6724405819	0
2	2	D388	LINESTRING Z(174.1202181 38.73654274 3.7, 169.6126998 37.12779	2	388	4.7860001061	1.6724405819
3	3	BG.West.808	LINESTRING Z(176.4994354 38.88654053 3.7, 174.1202181 38.7365	3	2,939	2.3839409174	6.4584406879
4	4	D671	LINESTRING Z(176.4994354 38.88654053 3.7, 178.2834629 47.474	4	1,427	8.7712313384	8.8423816053
5	5	BG.West.807	LINESTRING Z(178.2223304 37.11163021 3.7, 178.2834629 47.4744	5	892	10.3629742558	17.6136129437
6	6	D136	LINESTRING Z(178.2223304 37.11163021 3.7, 178.3558189 28.9750	6	2,404	8.1376291268	27.9765871995
7	7	BG.West.812	LINESTRING Z(179.7625808 9.703995695 3.7, 178.3558189 28.975	7	1,147	19.3223778759	36.1142163262
8	8	D391	LINESTRING Z(179.7625808 9.703995695 3.7, 186.0667799 9.6295	8	2,659	6.3046382877	55.4365942021
9	9	BG.West.815	LINESTRING Z(196.1762001 10.01108681 3.7, 186.0667799 9.62958	9	3,071	10.1166162313	61.7412324898

```sql
-- Elijah's route (teacher)
create or replace view route_vw_2 as
select distinct X.seq, Y.roomname, Z.geom, X.Path_seq, X.edge, X.cost, X.agg_cost
from
    pgr_dijkstra(
       'select id, source, target, cost from edge_vw_2',
        (select source from edge_vw_2 where fromnode ='BG.West.859' limit 1),
        (select target from edge_vw_2 where tonode ='BG.West.866' limit 1),
        false
    ) X
    join node as Y on X.node = Y.id
    join edge as Z on X.edge = Z.id
order by seq;
```



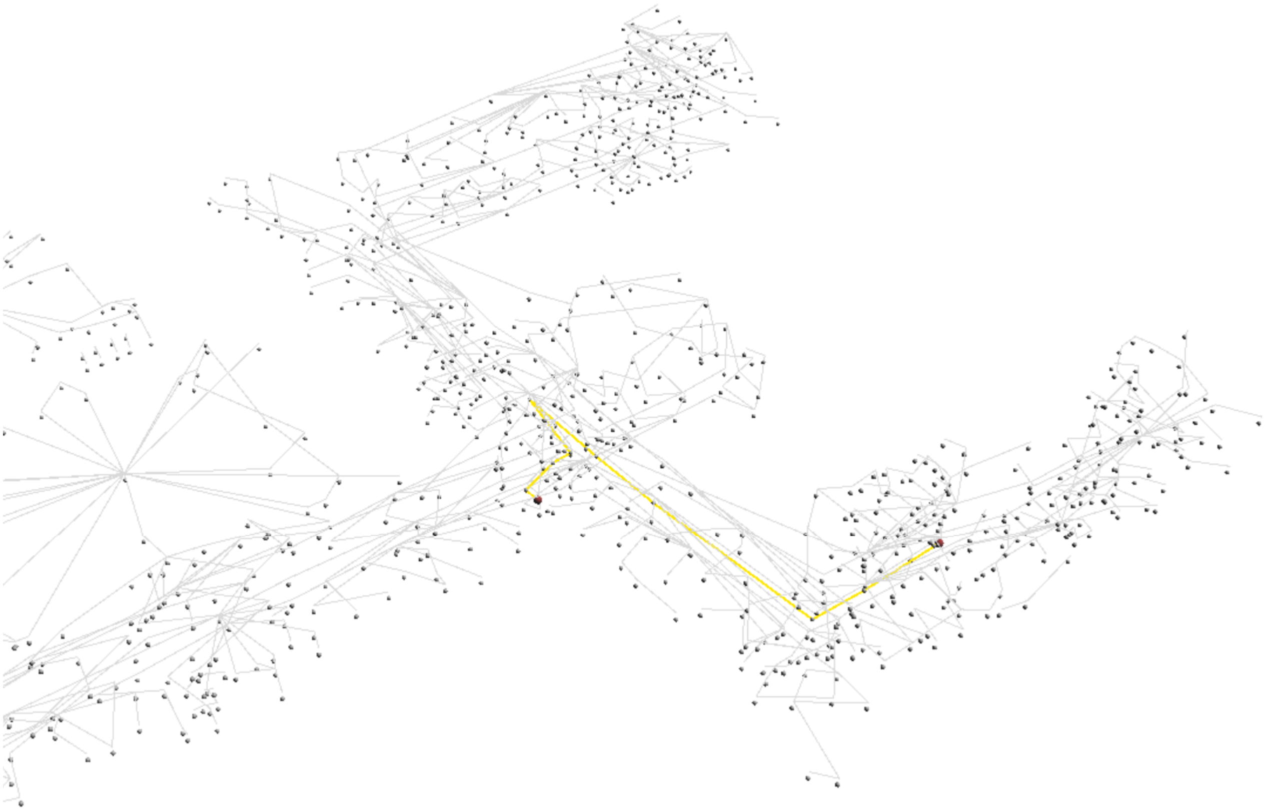
route_vw_2 | Enter a SQL expression to filter results (use Ctrl+Space)

	seq	roomname	geom	path_seq	edge	cost	agg_cost
1	1	BG.West.859	LINESTRING Z(169.7028523 35.45778399 3.7, 169.6126998 37.1277	1	1,900	1.6724405819	0
2	2	D388	LINESTRING Z(174.1202181 38.73654274 3.7, 169.6126998 37.12779	2	388	4.7860001061	1.6724405819
3	3	BG.West.808	LINESTRING Z(175.0151112 37.11163021 3.7, 174.1202181 38.736542	3	1,142	1.8550402666	6.4584406879
4	4	D386	LINESTRING Z(175.0151112 37.11163021 3.7, 175.0682156 35.831026	4	2,654	1.2817041506	8.3134809545
5	5	BG.West.370	LINESTRING Z(176.4426112 36.03663021 3.7, 175.0682156 35.8310	5	1,141	1.3896892045	9.5951851051
6	6	D385	LINESTRING Z(176.4426112 36.03663021 3.7, 178.3558189 28.9750	6	2,653	7.3161211595	10.9848743096
7	7	BG.West.812	LINESTRING Z(179.7625808 9.703995695 3.7, 178.3558189 28.975	7	1,147	19.3223778759	18.300995469
8	8	D391	LINESTRING Z(179.7625808 9.703995695 3.7, 186.0667799 9.6295	8	2,659	6.3046382877	37.6233733449
9	9	BG.West.815	LINESTRING Z(196.1762001 10.01108681 3.7, 186.0667799 9.62958	9	3,071	10.1166162313	43.9280116326

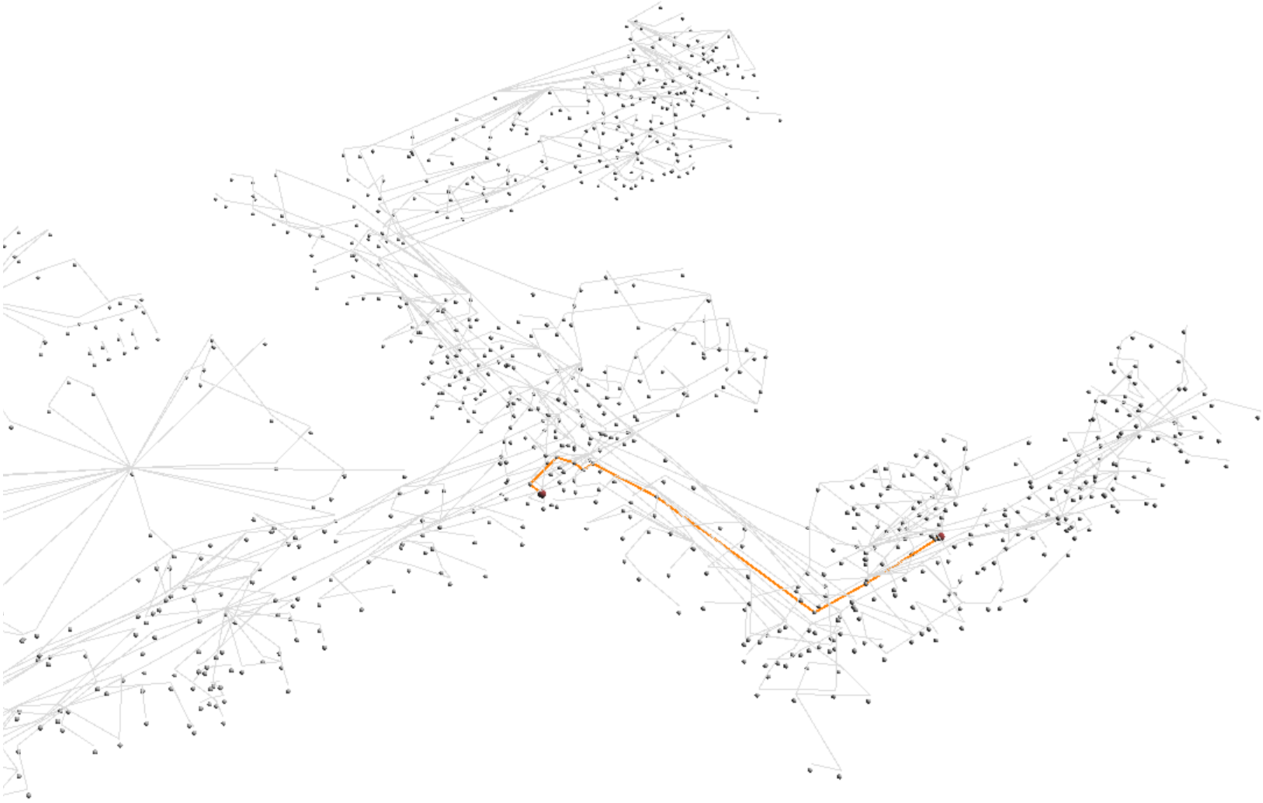
6.D.3 View data in QGIS

Visualize the BK building network (nodes, edges) in QGIS, together with the paths of the two building users (as mentioned above).

Liam's route (Student):



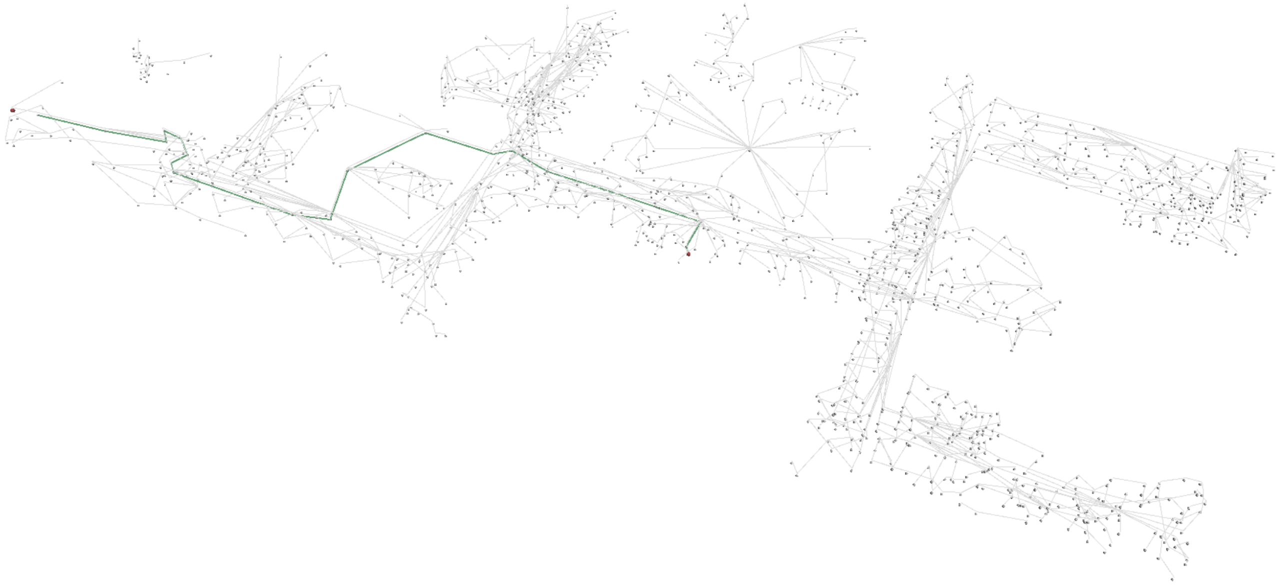
Elijah's route (Teacher):

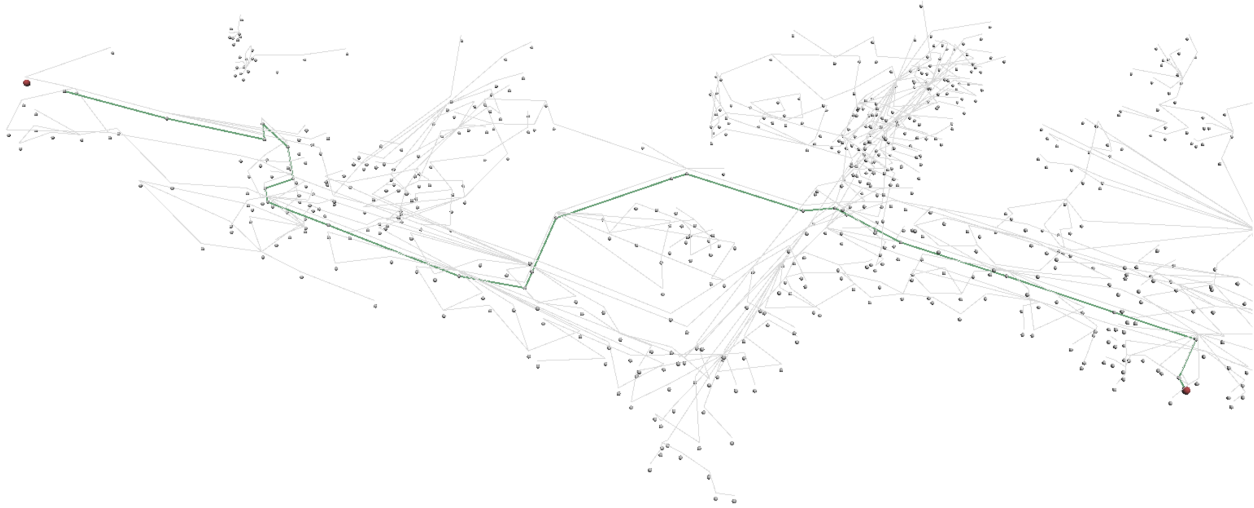


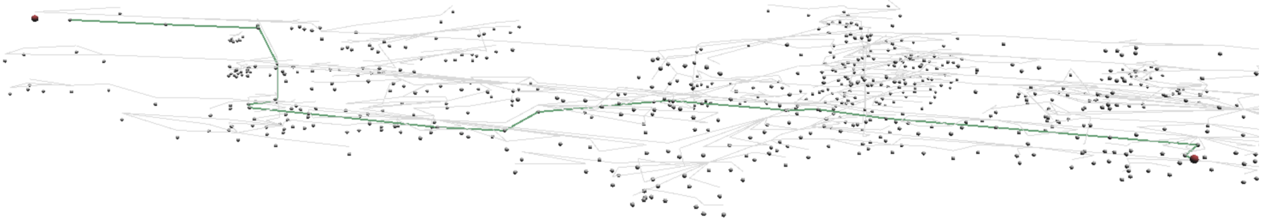
Define a route from the building entry (BG.Mid.803) to the Geolab (02.Oost.600) for one of the students.

```
create view node_vw3 as select node.id, node.geom from node, rights, party
  where party.name= 'Noah' and rights.pid=party.pid and rights.type= 'access' and rights.nid=node.id;

create view edge_vw3 as select edge.*  from edge, node_vw3 nf, node_vw3 nt
  where nf.id=edge.source and nt.id=edge.target;

create view route_vw3 as
SELECT distinct X.seq, Y.roomname, Z.geom, X.Path_seq, X.edge, X.cost, X.agg_cost
FROM
    pgr_dijkstra(
       'select id,source, target,  cost from edge_vw3',
        (select source from edge_vw3 where fromnode ='BG.Mid.803' limit 1),
        (select source from edge_vw3 where tonode ='02.Oost.600' limit 1),
        FALSE
    ) X JOIN
    node AS Y ON X.node = Y.id JOIN
    edge AS Z ON X.edge = Z.id
ORDER BY seq;
```

Route from entrance to Geolab for a student:

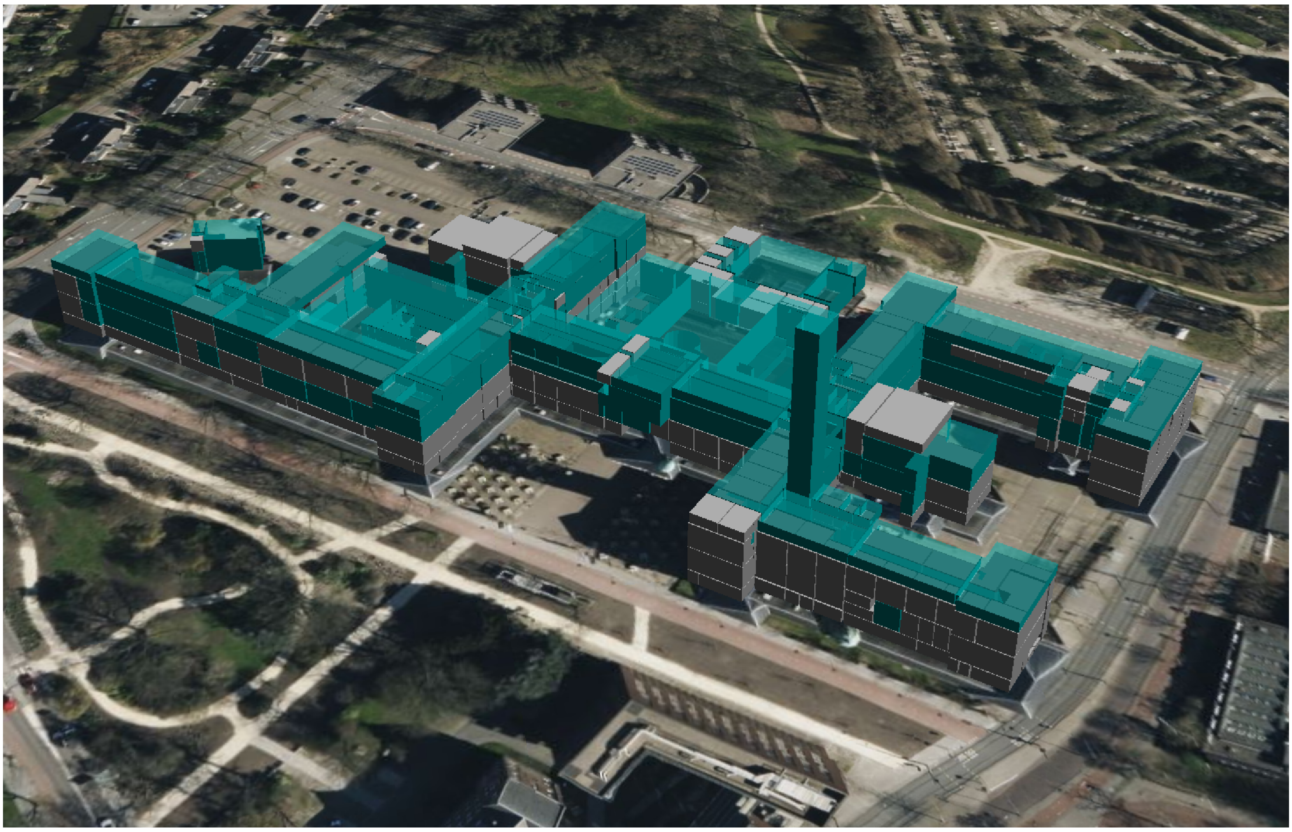




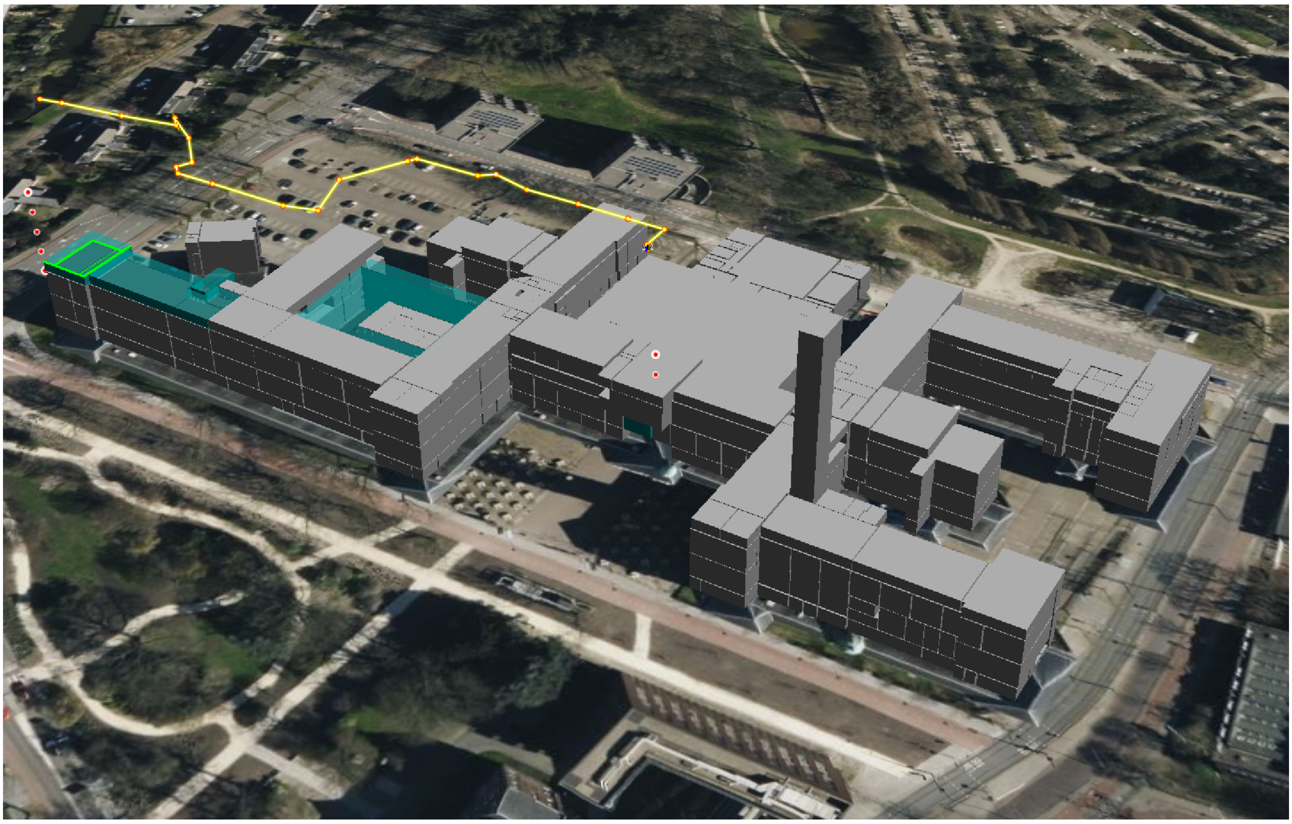


6.D.4 Use the 3D webviewer and do some interactive routing

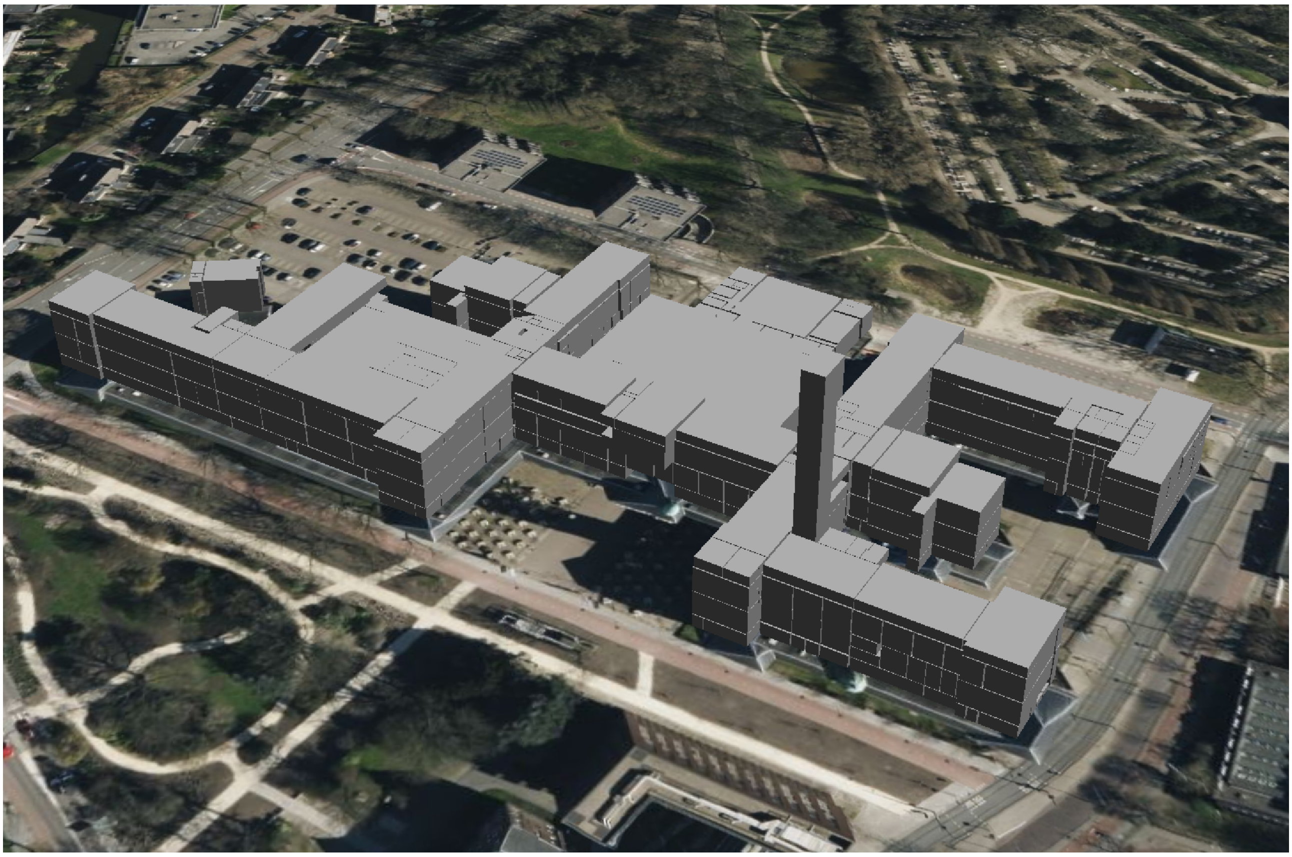
Rooms where the students have access/are allowed.



From the entrance (BG.Mid.803) to Geolab (02.Oost.600) for a student.



Start BG.Mid.080 to Geolab (02.Oost.600) for a student.
No result:



Start BG.Mid.080 to Geolab (02.Oost.600) for a staff member.