RMIT UNIVERSITY

< DATA ANALYSIS FOR OPENSTREETMAP >

TECHNICAL REPORT

Version 5.0

Start Date: **28 /03 /2018**

Authors : **Lei Tao, s3552269**
Wenbo Peng, s3620955
Yifeng Wang, s1234567
Tianze Zhao, s3573724
Bogeng Xie, s3556192
Abdulaziz Ibrahim A Alrasheed, s3607495

DOCUMENT CONTROL

Version #	Implemented By	Implementation Date	Reviewed By	Approval Date	Reason
1	Yifeng wang, Lei Tao, Bogeng Xie, Wenbo Peng, Tianze Zhao	28/03/2018	Yifeng Wang	28/03/2018	
2	Yifeng wang, Lei Tao, Bogeng Xie, Wenbo Peng, Tianze Zhao	23/4/2018	Lei Tao	23/4/2018	
3	Yifeng wang, Lei Tao, Bogeng Xie, Wenbo Peng, Tianze Zhao	7/5/2018	Wenbo Peng	7/5/2018	
4	Yifeng wang, Lei Tao, Bogeng Xie, Wenbo Peng, Tianze Zhao	21/5/2018	Bogeng Xie	21/5/2018	
5	Yifeng wang, Lei Tao, Bogeng Xie, Wenbo Peng, Tianze Zhao	5/6/2018	Tianze Zhao	5/6/2018	

TABLE OF CONTENTS

EXECUTIVE SUMMARY **4**

INTRODUCTION **4**

REQUIREMENTS **7**

FUNCTIONAL REQUIREMENTS SPECIFICATION 7

NON FUNCTIONAL REQUIREMENTS SPECIFICATION 9

ARCHITECTURE **10**

TECHNICAL FRAMEWORK **12**

IMPLEMENTATION **13**

DEPLOYMENT INSTRUCTIONS **18**

TEST SPECIFICATIONS **21**

TESTING RESULTS **23**

OTHER CONSIDERATIONS **23**

REFERENCES **23**

1 Executive Summary

This project is an open task project, the main goals of it is to identify the potential problems within OpenStreetMap, detect anomalies in the data, and if possible, build a software to fix those problems. We may likely to use a range to technologies in order to fit different situations, from unix scripts to Java code. In the end, everything the group done will be open source and hosted on GitHub and be able to run on unix-based systems, so that it can be integrated into automated workflows into various projects. In the process, we will have lots of opportunities for creative engineering, problem solving, and learning, but importantly will deliver a tool that will be of real value to the community.

Our project is divided into two parts: find potential problems in original OSM data and compare OSM with google map.

Finding potential problems: When we look at the OSM data from OSM website,we can easily found that some tags are missing. Some others potential problems can be found through the analysis of the data. Details of analysis will explicit in implementation part.

Comparing OSM with google map: To find potentially incorrect routes and analyse them, the program also provides a function of comparing direction data from OSM and Google because Google is usually thought to be an more accurate comparator. The function includes time and distance comparison. While user defines a threshold which can be changed any time, the program can analyse the routes input by the user or generated by the program automatically. The analysis results conclude summary analysis result and detailed analysis result. Detailed result is written from the data of dividing potential problem routes into 20 parts evenly and getting obviously different results. All the results can be stored in local path as csv file for further use to identify real problem in OSM database.

2 Introduction

OpenStreetMap(OSM) is a collective project for editing the map of the world, to which everyone could make contribution, and it completely free. The data of the map contains information about roads, paths, buildings, areas, etc., and they are all being updated by volunteers throughout the world. Compared with google map, which is still the dominant map out there, OSM is rapidly gaining favor among many companies. The advantages of OSM are availability, customizable and updated by wider community, but maintaining the map is quite challenging and many problems could come with it, like inaccurate data, vandalism, missing details.

The basic components which consist of OpenStreetMap data model are three elements, nodes (defining points in space),ways (defining linear features and area boundaries), and relations (which are sometimes used to explain how other elements work together).
Like other XML files OSM data, all types of data have Tags which describe the meaning of the particular element to which they are attached. The common attributes are id, user, uid, timestamp, visible, version and changeset.

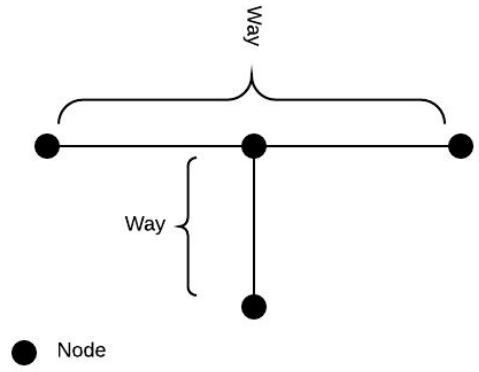


Figure 1: <an open way that describes a linear feature which does not share a first and last node. Many roads, streams and railway lines are open ways>

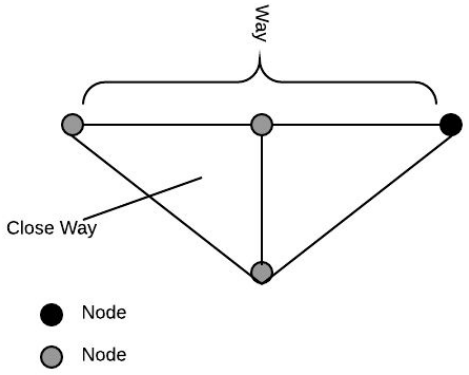


Figure 2: <last node of the way is shared with the first node with suitable tagging. The close way also describes the roundabouts and circular walks, barriers, such as hedges and walls, that go completely round a property>

Comparison of OSM with Google Algorithm

This algorithm firstly searches on OSM with origin/destination pairs as input and get the routest detailed information with the help of Graphhopper API. Meanwhile, to make sure that Google is similarly using a same route as OSM, we evenly get 20 nodes from the OSM response and put them into the request as waypoints, then send back to Google Waypoints API to get a Google route finally. The reason to divide the OSM route by 20 Is that 22 is the maximum number to be input into a Google Waypoints API request.

From the response of OSM and Google, we are able to get the total distance and time consumed from the two maps. Once the user sets a difference threshold for reporting, our program would print the summary analysis of all the routes that exceed the threshold. With this summary analysis, user can then use any of those routes as an input to the program again to get a detailed analysis. The detailed analysis gives the user all the distance and time consumption of 20 individual parts from the two maps. Then it is more easy to observe where comes the potential problem. But remember that even we find a big difference from those parts, we still can't make sure OSM is right and Google is wrong or vise versa because they can be both wrong. This algorithm is giving the user a great possibility to scan a big *.osm file and locate potential problems. No matter what that problem is, the user gets the coordinates and all distance and time consumption details from those routes, and the data can be made use on other algorithm later.

Here's an example. When using origin and destination (-36.3218766,148.6076293 and -35.2288026,149.0753201) as input into OSM and evenly putting 20 nodes from the response back to Google API, the program finds the total distance difference percent from two maps is 19.679 %. Once considering there exists a potential problem, then we can look into the detailed analysis which finally tells us that a part in OSM route has a 0.208 km distance when it's 49.205 km in Google. In the visualisation below, it's easy to observe that OSM can't pass a thin water area (left-hand side))when the water area's kind of blocking the way.

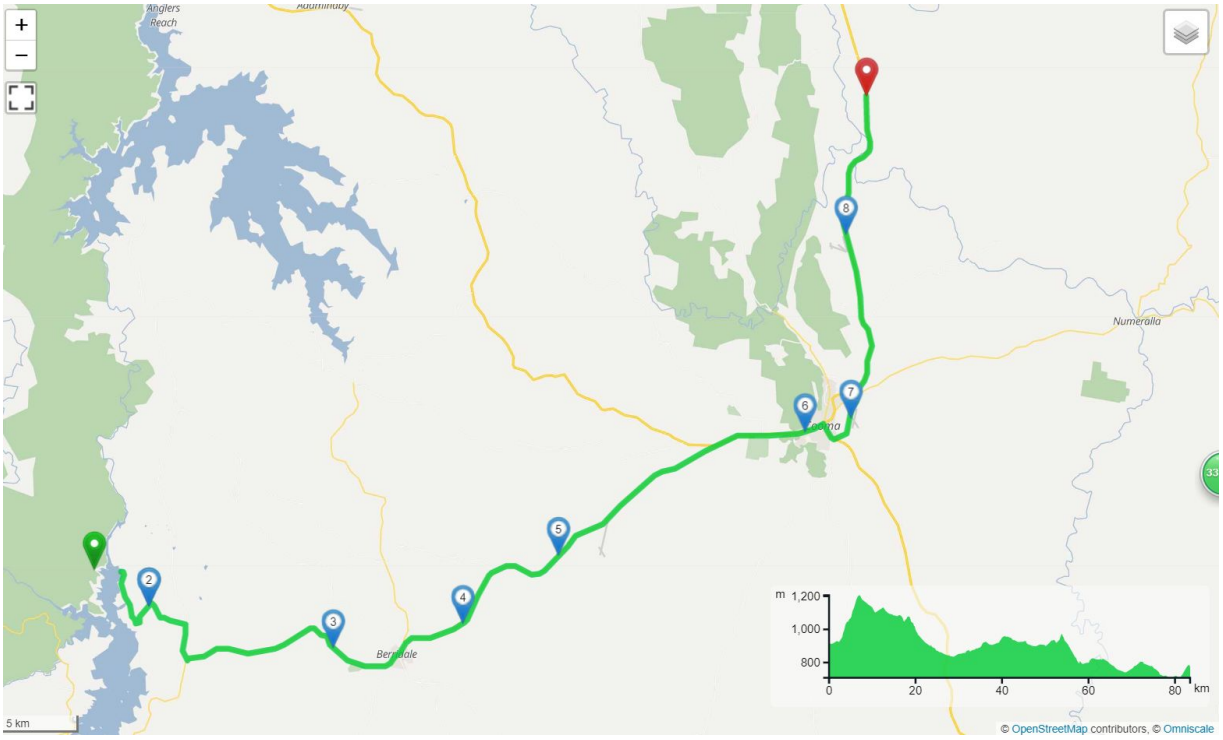


Figure 3: <The example route from OSM (The difference is in left-hand side)>

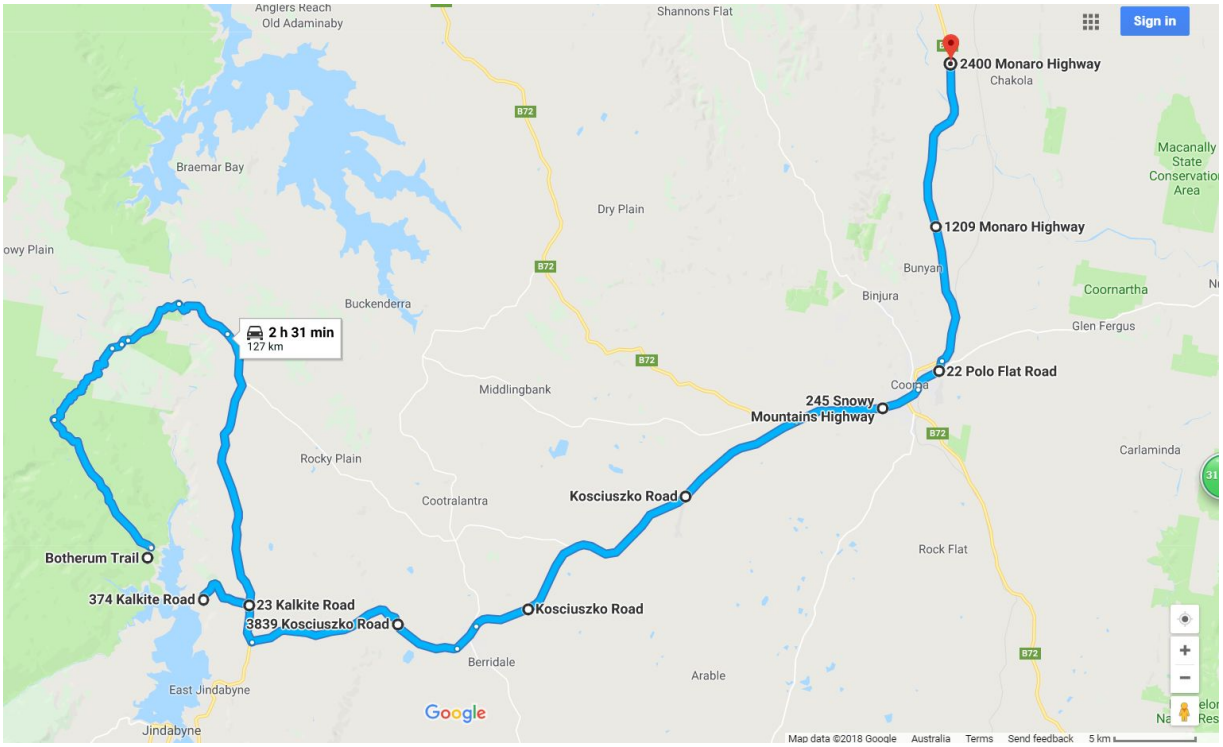


Figure 4: < The example route from Google>

3 REQUIREMENTS

FUNCTIONAL REQUIREMENTS SPECIFICATION

Use case 1: The program converts osm file into SQlite database file.
Use case 2: The program finds the data which has “maxspeedgap” problem.
Use case 3: The program finds the data which has “maxspeedInWayRelation” problem.
Use case 4: The program finds the data which has “FromViaToInRelation” problem.
Use case 5: The program finds the data which has “NameShortName” problem.
Use case 6: The program finds the data which has “SameRouteInRWP” problem.
Use case 7: The program finds the data which has “TypeBoudayInRelation” problem.
Use case 8: The program finds the data which has “TypeRestrictionInRelation” problem.
Use case 9: The program finds the data that missing some data and outputs the id of the data on the screen.
Use case 10: The program writes data to the .txt file.

Comparison of OSM with Google:
Use case 11: The program generates origin and destination randomly within a specified radius and prints summary analysis.
Use case 12: The program reads a list of JSON-formatted origin/destination from user input and prints the summary analysis result.
Use case 13: The program stores the summary result into a local file.
Use case 14: The program generates and stores detailed analysis based on the a summary file.

Req. ID	Description	Priority	UC covered
1	When the user input the right file address, the program shall store the data from .osm file to the sqlite database.	high	UC 1
2.1	When the user input the right argument and run the program, the program shall search data from SQlite database.	high	UC 2
2.2	When the program get the data from SQlite database, the program shall find the data which have “maxspeedgap” problem	high	UC 2
2.3	When the data have problem, the program shall output these data to to the txt file.	high	UC 2
3.1	Related 2.1	high	UC 3
3.2	When the program get the data from SQlite database, the program shall find the data that have “maxspeedInWR” problem	high	UC 3
3.3	Related 3.3	high	UC 3
4.1	Related 4.1	high	UC 4
4.2	When the program get the data from SQlite database, the program shall find the data that have “FromViaToInRelation” problem	high	UC 4
4.3	Related 4.3	high	UC 4
5.1	Related 5.1	high	UC 5

5.2	When the program get the data from SQlite database, the program shall find the data that have "NameShortName" problem	high	UC 5
5.3	Related 5.3	high	UC 5
6.1	Related 6.1	high	UC 6
6.2	When the program get the data from SQlite database, the program shall find the data that have "SameRouteInRWP" problem	high	UC 6
6.3	Related 6.3	high	UC 6
7.1	Related 7.1	high	UC 7
7.2	When the program get the data from SQlite database, the program shall find the data that have "TypeBoudayInR" problem	high	UC 7
7.3	Related 7.3	high	UC 7
8.1	Related 8.1	high	UC 8
8.2	When the program get the data from SQlite database, the program shall find the data that have "TypeRestrictionInR" problem	high	UC 8
8.3	Related 8.3	high	UC 8
9.1	When the user enter the attribute and name, the program shall send the attribute and name to the SQlite database.	high	UC 9
9.2	When the SQlite receive the attribute and name, the SQlite shall search the data according to the attribute and the name.	high	UC 9
9.3	When the SQlite find the attribute and the name, the SQlite shall send the data to the program.	high	UC 9
9.4	When the program receive the data, the program shall display the data to the user.	high	UC 9
11	When user inputs the number of origin node that the program would generate randomly from database, max radius of destination and threshold for reporting potentially incorrect routes, the program would print summary result.	high	UC 11
12	When user inputs a list of origin/destination pairs and the threshold for reporting potentially incorrect routes, the program shall parse the list and print summary result	high	UC12
13	When user defines a file path to store the summary result, the program shall write the summary result into that path	medium	UC13
14	When user inputs a summary file and defines a list of route ID and a file path to store the detailed analysis result, the program shall find those routes from summary file and generates the detailed analysis into the path.	high	UC 14

NON FUNCTIONAL REQUIREMENTS SPECIFICATION

After Users/Analysers run Main functions to handle data from OSM, ".db"file will be automatically created.

Attribute	Score	maintainability	robustness	availability	integrity	flexibility	usability	interoperability	efficiency	testability	reliability	reusability	portability
Maintainability	6		<	<	^	^	<	^	^	<	<	^	<
robustness	5			^	^	<	^	<	<	<	^	^	<
availability	7				^	<	^	<	<	<	^	<	<
integrity	6					^	^	^	^	<	^	<	<
flexibility	2					^	^	^	^	^	^	^	^
usability	10							<	<	<	<	<	<
interoperability	3								^	^	^	^	^
efficiency	4									^	^	^	^
testability	3										^	^	^
reliability	8											<	^
reusability	7												<
portability	5												

USE-1 ALL FUNCTIONS KEY ON THE README FILE

AVL-1 SYSTEM SHALL BE BEARING THE LOAD OF USERS, AND IT WILL BE AT LEAST 99% AVAILABLE 24 HOURS

4 ARCHITECTURE

There are two parts, one is comparing Google Map with Open Street Map , another is finding basic problems like streets losing name, speed gap over 20km in one road etc. .

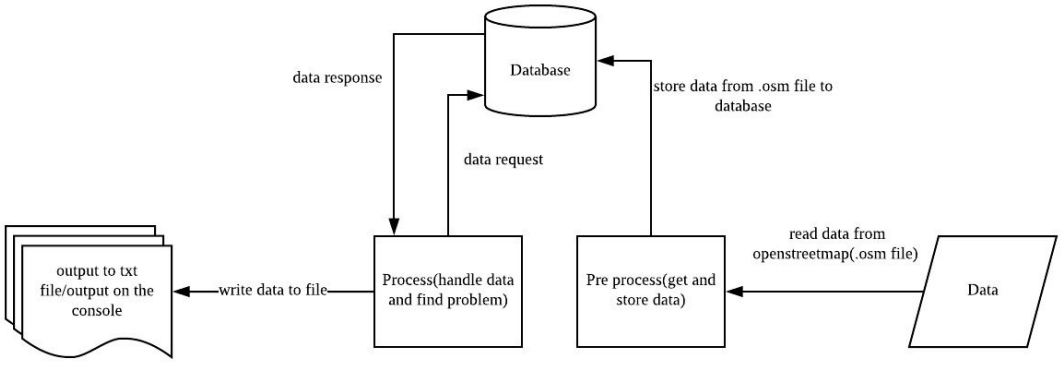


Figure 5: <Data processing of basic problems finding>

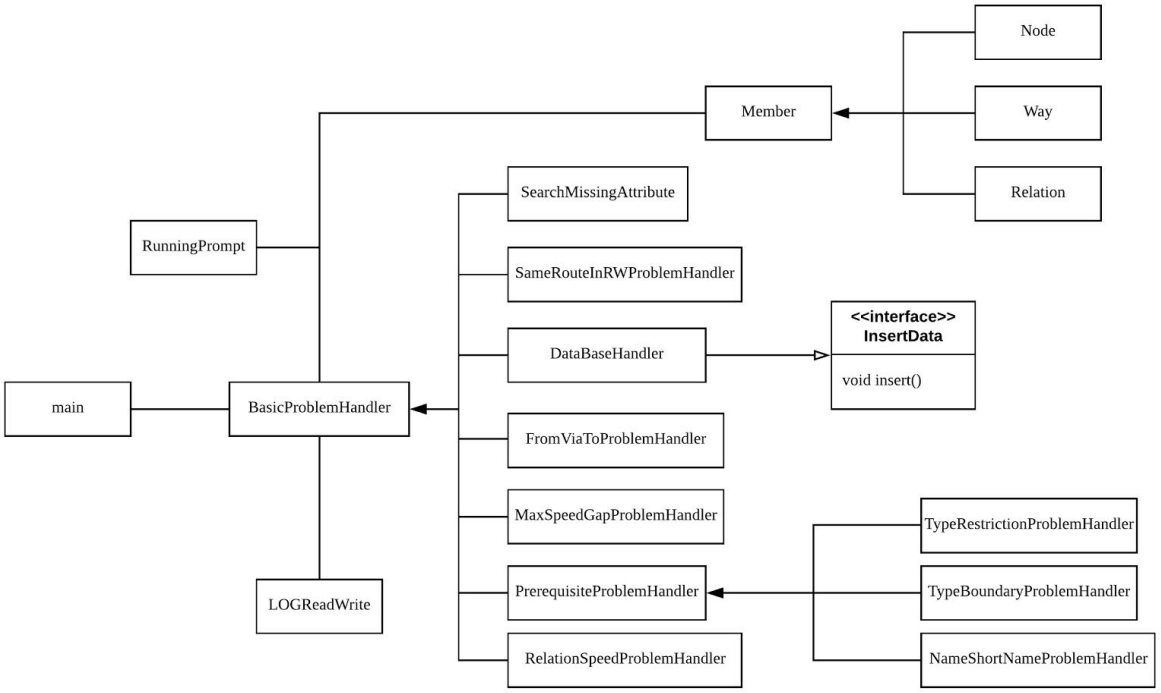


Figure 6: <Class diagram of basic problems finding>

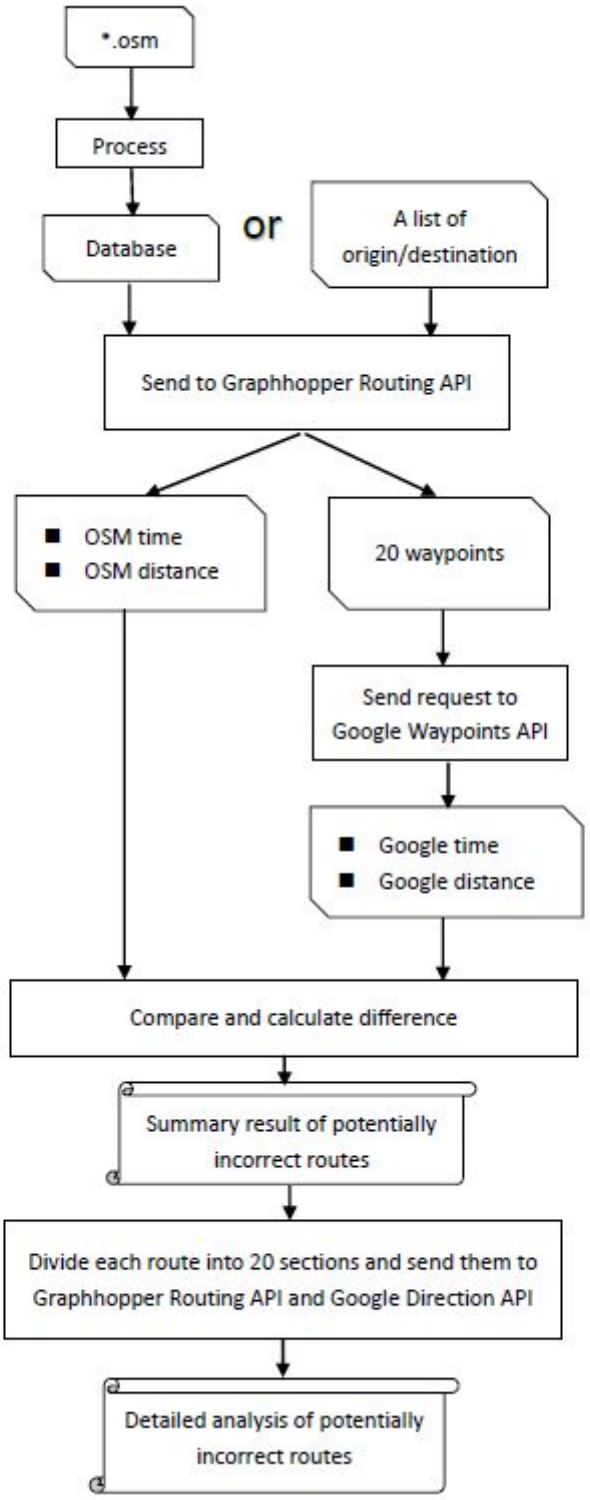


Figure 7:< OSM & Google Comparison data processing>

5 Technical Framework

The technologies, environments, and languages we apply in the project.

- JAVA: A popular computer-programming language. We decide to use this language because Java is easy to learn and use. In addition, the Java program can be compiled once and runs anywhere.
- Database(Sqlite): The SQLite database is a relational database and embedded into the end program. We decide to use this database rather than other database(for

example, MySQL) because our project is designed to be used simply and faster. Thus, SQLite is suitable for our project as it takes a little resources and can be used in many languages.

- Eclipse/IDEA: Eclipse is an integrated development environment for computer programming. And most developer who use Java could use this software. We use Eclipse because this software are used to develop application which using Java.
- Maven/Github/Git/Trello: Maven is a tool for build automation and developed by Java. the biggest reason we decide to use maven is that people do not need to download extra libraries as long as the project is built based on maven. Just a pom.xml file, when we use command "maven clean install" to build the program, the extra libraries will be download automatically according to the pom.xml. Github is a web service for version control which using git. We decide to use Github because it provides version control and code management. By using github, we can clearly manage our project and merge code from team mates. In addition, it provides strong collaboration functions, for example, bug tracking and task management.
- Graphhopper API: With the GraphHopper Directions API you can integrate A-to-B route planning, route optimization, isochrone calculations and more in your application. The GraphHopper Directions application programming interface connects your application with our algorithms.
- Google Direction API: The Google Directions API is a service that calculates directions between locations using an HTTP request. In this project , routing and waypoints function are used.
- DB Browser for SQLite: DB Browser for SQLite is a tool that developed to make the database visual. As we use SQLite database, we can clearly see the database and tables structure by using DB Browser for SQLite, as well as the data. It saves a lot of time for us because we do not need to look through or search the data by Java.

6 IMPLEMENTATION

Describe the implementation of the app/webpage/tool/… that you developed; please provide the corresponding screenshots.

OSM & Google comparison program

(1) To generate a database file from a *.osm file

```
D:\Semester 4\osm-analysis\tianze-bogeng\project>java -cp target/project-1.0-SNAPSHOT-jar-with-dependencies.j
ar osm.analysis.google.Main --osm-read-path mount_alexander_shire_network_2018.osm --db-store-path osm.db
```

(2) To get a summary result with manually coordinates input

```
D:\Semester 4\osm-analysis\tianze-bogeng\project>java -cp target/project-1.0-SNAPSHOT.jar osm.analysis.google.Main --list-of-orig/des
t {routes:[{ori:{lat:-37.6941552,lon:144.5793929},dest:{lat:-34.2968327,lon:147.5588931}},{ori:{lat:-36.0905765,lon:146.9311751},dest
:{lat:-35.5126728,lon:148.6093932}},{ori:{lat:-34.3001501,lon:142.1915312},dest:{lat:-35.6469135,lon:148.6763364}},{ori:{lat:-34.3607
745,lon:148.025851},dest:{lat:-36.6893697,lon:142.4108182}},{ori:{lat:-34.5535703,lon:146.4113448},dest:{lat:-34.6079865,lon:147.1291
069}},{ori:{lat:-37.9158962,lon:142.4296366},dest:{lat:-37.1788221,lon:144.0485408}},{ori:{lat:-36.3472808,lon:146.8028536},dest:{lat
:-37.2747424,lon:149.0709654}},{ori:{lat:-37.6874412,lon:144.4336144},dest:{lat:-37.7571033,lon:144.9347327}},{ori:{lat:-36.4217525,l
on:148.6168595},dest:{lat:-36.1342025,lon:146.9038279}},{ori:{lat:-35.2469941,lon:149.0977115},dest:{lat:-36.1342025,lon:141.284169}}
]} --dist-diff-reporting-threshold-percent 0.3 --summary-store-path summary.csv --detail-store-path detail.csv
```

The program would keep printing "Generated: origin,destination" until none left, then print the summary result. Data storing is processed after.

```
Generated: -37.6941552,144.5793929/-34.2968327,147.5588931
Generated: -36.0905765,146.9311751/-35.5126728,148.6093932
Generated: -34.3001501,142.1915312/-35.6469135,148.6763364
Generated: -34.3607745,148.025851/-36.6893697,142.4108182
Generated: -34.5535703,146.4113448/-34.6079865,147.1291069
Generated: -37.9158962,142.4296366/-37.1788221,144.0485408
Generated: -36.3472808,146.8028536/-37.2747424,149.0709654
Generated: -37.6874412,144.4336144/-37.7571033,144.9347327
Generated: -36.4217525,148.6168595/-36.1342025,146.9038279
Generated: -35.2469941,149.0977115/-36.1342025,141.284169

---------------------------------------------------- Summary ----------------------------------------------------------

No.  Origin                  Destination             OSM(ms)    Google(ms)  Time Diff(%) Time Diff(ms) Dist Diff(%) Dist Diff(m)
1    -37.6941552,144.5793929 -34.2968327,147.5588931 22344943   21498000    3.790        846943        0.931        5359
2    -34.5535703,146.4113448 -34.6079865,147.1291069 5294140    4833000     8.710        461140        0.459        468
3    -36.3472808,146.8028536 -37.2747424,149.0709654 23894495   26274000    9.056        2379505       1.217        5593
4    -37.6874412,144.4336144 -37.7571033,144.9347327 2606201    2956000     11.833       349799        0.346        189
5    -35.2469941,149.0977115 -36.1342025,141.284169  36220376   36342000    0.335        121624        0.655        6092
Storing data...
Storing finished.
```

(3) To get a summary result with program randomly generating origins and destinations

```
D:\Semester 4\osm-analysis\tianze-bogeng\project>java -cp target/project-1.0-SNAPSHOT-jar-with-dependencies.jar osm.analysis.google.Main --generate-random-origins 10 --radius-of-dest 800 --db-read-path osm.db --time-diff-reporting-threshold-hr 0.1 --summary-store-path summary.csv
```

The program would keep printing "Generated: origin,destination" until none left, then print the summary result. Data storing is processed after.

```
Generated: -35.38219,143.66578/-37.0560419,144.2562355
Generated: -35.969291,144.929684/-36.1222097,145.5856395
Generated: -37.7094371,144.8502199/-37.544953,144.1460184
Generated: -37.8699262,145.1586532/-35.2057052,149.059621
Generated: -37.157475,142.530144/-37.0984295,147.2971504
Generated: -36.1538106,146.6065602/-34.044485,148.7308332
Generated: -35.264971,149.1257192/-34.9337247,148.7872865
Generated: -37.5690376,143.8366296/-36.0776248,146.9369275
Generated: -35.4000394,143.134947/-35.1809161,149.1098838
Generated: -35.485458,149.2713687/-35.2521672,149.1455089

---------------------------------------------------- Summary ----------------------------------------------------------

No  Origin                  Destination             OSM(hr) Google(hr) Time Diff(hr) Time Diff(%) OSM(km)  Google(km) Dist Diff(km) Dist Diff(%)
1   -35.38219,143.66578     -37.0560419,144.2562355 2.773   2.939      0.166        5.648        244.345  249.303    4.958         1.989
2   -37.157475,142.530144   -37.0984295,147.2971504 9.855   9.481      0.374        3.795        639.059  662.785    23.726        3.58
3   -35.264971,149.1257192  -34.9337247,148.7872865 0.948   1.458      0.51         34.979       71.739   94.8       23.061        24.326
4   -35.485458,149.2713687  -35.2521672,149.1455089 1.457   1.216      0.241        16.541       51.18    56.864     5.684         9.996
```

	A	B	C	D	E	F	G	H	I	J	K
1	ID	ORIGIN_LAT	ORIGIN_LON	DEST_LAT	DEST_LON	OSM_HR	OSM_KM	GOOGLE_HR	GOOGLE_KM	DIFF_TIME_PERCNET	DIFF_DIST_PERCENT
2	1	-35.38219	143.66578	-37.0560419	144.2562355	2.773	244.345	2.939	249.303	5.648	1.989
3	2	-37.157475	142.530144	-37.0984295	147.2971504	9.855	639.059	9.481	662.785	3.795	3.58
4	3	-35.264971	149.1257192	-34.9337247	148.7872865	0.948	71.739	1.458	94.8	34.979	24.326
5	4	-35.485458	149.2713687	-35.2521672	149.1455089	1.457	51.18	1.216	56.864	16.541	9.996

(4) To store detailed analysis

```
D:\Semester 4\osm-analysis\tianze-bogeng\project>java -cp target/project-1.0-SNAPSHOT-jar-with-dependencies.jar osm.analysis.google.Main --input-summary-file summary.csv --analyze-route-id 1-3 --detail-store-path detail.csv
```

```
Start storing detailed analysis into detail.csv. It takes about 15s each route.
Storing:-35.38219,143.66578/-37.0560419,144.2562355
Storing:-37.157475,142.530144/-37.0984295,147.2971504
Storing:-35.264971,149.1257192/-34.9337247,148.7872865
Storing finished.
```

	A	B	C	D	E	F	G	H	I
1	ID	ORIGIN_LAT	ORIGIN_LON	DEST_LAT	DEST_LON	OSM_HR	OSM_KM	GOOGLE_HR	GOOGLE_KM
2	1	-35.38219	143.66578	-37.0560419	144.2562355	2.773	244.345	2.939	249.303
3	1.1	-35.38219	143.66578	-35.368029	143.584056	0.174	12.251	0.163	12.255
4	1.2	-35.368029	143.584056	-35.437707	143.612116	0.17	12.685	0.147	12.679
5	1.3	-35.437707	143.612116	-35.535893	143.745312	0.192	16.989	0.187	16.987
6	1.4	-35.535893	143.745312	-35.625462	143.807171	0.122	12.082	0.124	12.08
7	1.5	-35.625462	143.807171	-35.720173	143.886607	0.146	13.503	0.139	13.501
8	1.6	-35.720173	143.886607	-35.793574	143.949266	0.135	10.776	0.135	10.773
9	1.7	-35.793574	143.949266	-35.945357	143.939402	0.179	17.926	0.179	17.901
10	1.8	-35.945357	143.939402	-36.158849	143.938657	0.246	24.554	0.245	24.553
11	1.9	-36.158849	143.938657	-36.409621	143.974498	0.285	28.504	0.29	28.504
12	1.1	-36.409621	143.974498	-36.670369	144.197058	0.36	36.035	0.375	36.037
13	1.11	-36.670369	144.197058	-36.728585	144.186801	0.093	8.823	0.104	8.818
14	1.12	-36.728585	144.186801	-36.78515	144.152399	0.12	11.384	0.142	11.364
15	1.13	-36.78515	144.152399	-36.851171	144.166564	0.083	7.897	0.097	7.901
16	1.14	-36.851171	144.166564	-36.886974	144.213161	0.063	6.037	0.146	11.135
17	1.15	-36.886974	144.213161	-36.950994	144.234427	0.078	7.435	0.087	7.432
18	1.16	-36.950994	144.234427	-36.986616	144.260313	0.047	4.686	0.058	4.682
19	1.17	-36.986616	144.260313	-37.041192	144.289631	0.073	6.883	0.094	6.876
20	1.18	-37.041192	144.289631	-37.048359	144.267057	0.113	4.035	0.113	4.005
21	1.19	-37.048359	144.267057	-37.056202	144.257418	0.089	1.764	0.108	1.712
22	1.2	-37.056202	144.257418	-37.0560419	144.2562355	0.005	0.106	0.006	0.108
23									
24	2	-37.157475	142.530144	-37.0984295	147.2971504	9.855	639.059	9.481	662.785
25	2.1	-37.157475	142.530144	-37.294805	142.849998	0.49	42.798	0.565	42.796
26	2.2	-37.294805	142.849998	-37.391738	143.262245	0.424	40.055	0.469	39.879
27	2.3	-37.391738	143.262245	-37.487283	143.770693	0.481	47.232	0.502	47.22

detail

Figure 9: <detailed analysis result>

"SearchMissingAttribute" function

This function is used to output the node/way/relation which do not have some attributes.

```
if(choice.equals("node"))
{
    input = "node_id";
    sql = "SELECT node_id FROM nodes_tags WHERE tag_key='"+attName+"'";
}
else if(choice.equals("way"))
{
    input = "way_id";
    sql = "SELECT way_id FROM ways_tags WHERE tag_key='"+attName+"'";
}
else if(choice.equals("relation"))
{
    input = "relation_id";
    sql = "SELECT relation_id FROM relations_tags WHERE tag_key='"+attName+"'";
}
else
{
    System.out.println("incorrect input, please input 'node' or 'way' or 'relation'.");
    System.exit(0);
}
```

There are 4 steps to implement this function:

1. Input the attribute name
2. Search the IDs of node/way/relation which have this attribute.
3. List all the IDs of node/way/relation.
4. Compare the IDs of step 2 and step 3 and remove the same IDs.
5. Output the IDs of node/way/relation on the console.

"FromViaTo" problem

Some relations have members which follow "from→via→to" format, as below:

```
<member type="way" ref="45387806" role="from"/>
<member type="node" ref="577203475" role="via"/>
<member type="way" ref="45387805" role="to"/>
```

Thus, when the members follows this format, there must be a same node that appear in these members. If members follow this format while no same node in these members, it could be a problem.

There are 5 steps to find the problem:

1. Search all relations that have this format.
2. Pick the ways that in these relations and have this format by searching relation ID.

```
while(res.next())
{
    test++;
    if(res.getString("member_role").equals("from"))
    {
        nodes[0] = res.getString("member_ref");
        check = test;
    }
    else if(res.getString("member_role").equals("via") && (check == test + 1))
    {
        if(res.getString("member_type").equals("way"))
        {
            ifVIAway = true;
        }
        nodes[1] = res.getString("member_ref");
    }
    else if(res.getString("member_role").equals("to") && (check == test + 2))
    {
        nodes[2] = res.getString("member_ref");
    }
}
```

3. Judge if these ways from step 2 have a same node(condition 1)/ judge if the node is as same as the last node of the first way and the first node of the last way(condition 2).

```
for(int i=0;i<nodes_first.size();i++)
{
    for(int j=0;j<nodes_last.size();j++)
    {
        if(nodes_first.get(i).equals(nodes_last.get(j)))
        {
            if(ifVIAway)
            {
                for(int k=0;k<nodes_via.size();k++)
                {
                    if(nodes_via.get(k).equals(nodes_first.get(i)))
                    {
                        ifcount = false;
                        break;
                    }
                }
            }
            else
            {
                if(nodes_first.get(i).equals(nodes[1]))
                {
                    ifcount = false;
                    break;
                }
            }
        }
    }
}
```

4. Output the relation IDs which do not satisfy either the condition 1 or condition 2.

"MaxSpeedGap" problem

Some ways have attribute called "maxspeed", when two ways are connected or interacted and both of them have "maxspeed" attribute, while the gap of two "maxspeed" are too big, for example, 50, it could be a problem.

There are 5 steps to find the problem:

1. Search all ways which have "maxspeed" attribute(condition 1).

```
String sql = "SELECT way_id,tag_value FROM ways_tags WHERE tag_key='maxspeed'";
```

2. Calculate gaps between every two ways(condition 2).

```
for(int j=i+1;j<ways.size()-1;j++)
{
    int actual_gap = Math.abs(ways.get(i).getSpeed() - ways.get(j).getSpeed());

    //if the actual_gap > speed_gap, no need to compare the node, just continue
    if(actual_gap > speed_gap) {
```

3. Pick the ways which satisfy the condition 1 and condition 2.
4. Pick every two ways which are connected or intersected by judge if the nodes of these two ways are same.

```
//compare every two ways which actual_gap > speed_gap
for(int i=0;i<ways.size();i++)
{

    int i_size = ways.get(i).getNodesRef().size();
    way_i_f = ways.get(i).getNodesRef().get(0);
    way_i_l = ways.get(i).getNodesRef().get(i_size-1);

    for(int j=i+1;j<ways.size()-1;j++)
    {
        int actual_gap = Math.abs(ways.get(i).getSpeed() - ways.get(j).getSpeed());

        //if the actual_gap > speed_gap, no need to compare the node, just continue
        if(actual_gap > speed_gap) {

            int j_size = ways.get(j).getNodesRef().size();
            way_j_f = ways.get(j).getNodesRef().get(0);
            way_j_l = ways.get(j).getNodesRef().get(j_size-1);

            //if the two ways are connected but don't intersect
            if(way_i_f.equals(way_j_f) ||
                    way_i_f.equals(way_j_l) ||
                        way_i_l.equals(way_j_l) ||
                            way_i_l.equals(way_j_f))
            {
```

5. Output the IDs, maxspeed and gap of these ways.

```
// way_maxspeed: id1=speed1 id2=speed2 diff=speed3
String output = "way_maxspeed_diff: " + "way_" + ways.get(i).getId() + "=" + ways.get(i).getSpeed()
        + " " + "way_" + ways.get(j).getId() + "=" + ways.get(j).getSpeed() + " diff=" + actual_gap;
content = content + output + "\r\n";
```

Output:

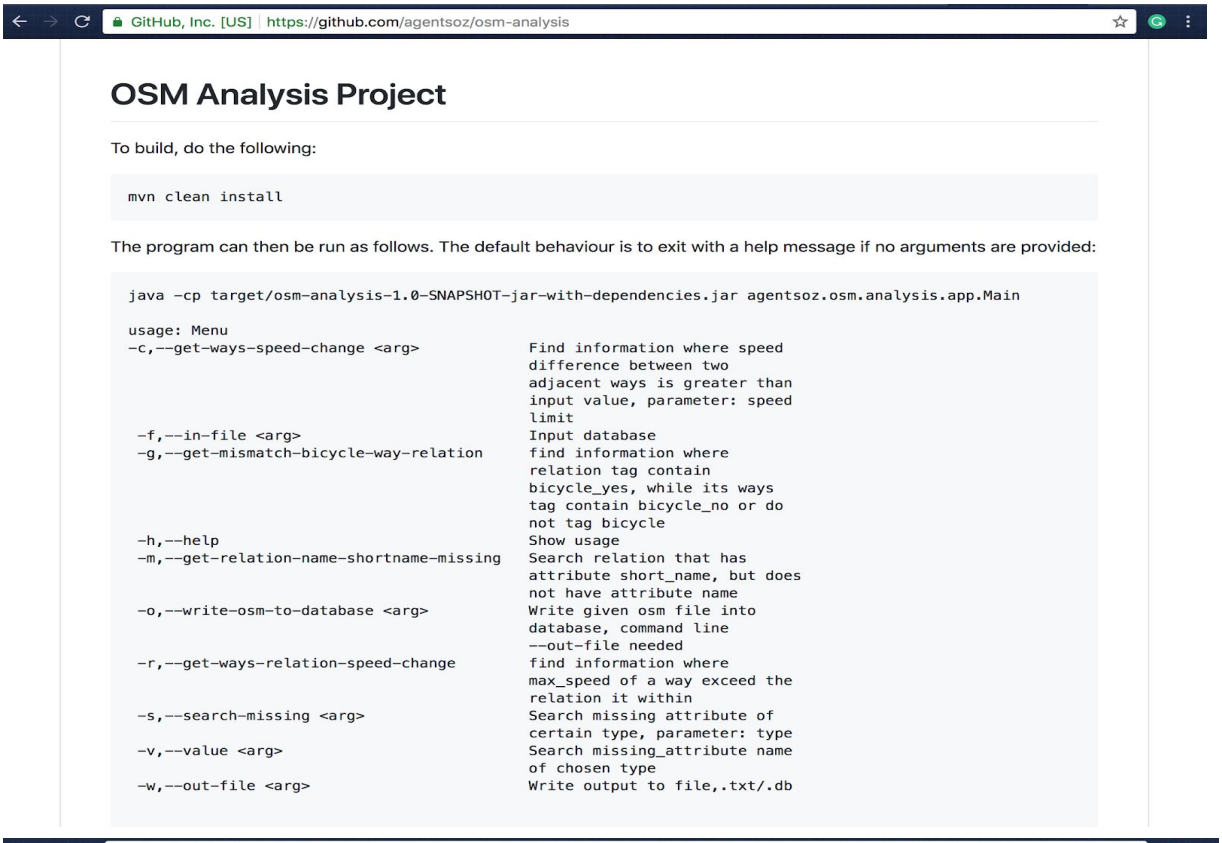
```
Console
<terminated> Main (13) [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (2018年6月4日 下午12:27:13)
way_maxspeed_diff: way_53055176=40 way_358744352=110 diff=70
way_maxspeed_diff: way_116486102=25 way_166479189=110 diff=85
way_maxspeed_diff: way_199296203=50 way_199468203=110 diff=60
way_maxspeed_diff: way_231297550=110 way_246794384=50 diff=60
way_maxspeed_diff: way_231297553=110 way_246794384=50 diff=60
way_maxspeed_diff: way_231297573=50 way_246794389=110 diff=60
way_maxspeed_diff: way_231297574=110 way_231297576=50 diff=60
way_maxspeed_diff: way_231297575=110 way_246794386=50 diff=60
way_maxspeed_diff: way_246794385=110 way_246794387=50 diff=60
```

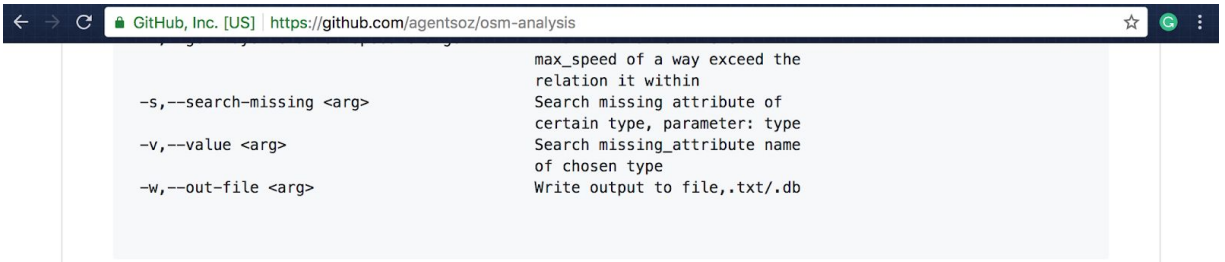
7 Deployment Instructions

Program of searching for basic problems

To build the program: maven clean install

To run the program:
input “java -cp target/osm-analysis-1.0-SNAPSHOT-jar-with-dependencies.jar agentsoz.osm.analysis.app.Main” with the following argument:





Here are some examples of how to use the program:

1. `-h, --help` : show usage/command line instruction; also the defautl behaviour if run with no arguments
2. `--write-osm-to-database mount_alexander_shire_network.osm --out-file osm.db` : This reads from `mount_alexander_shire_network.osm` and creates a `osm.db` in the root directory.
3. `--get-ways-relation-speed --in-file osm.db` : print list of ways in console
4. `--get-ways-speed-change 50 --in-file osm.db --out-file way_speed_change.txt` : write results into `way_speed_change.txt` .
5. `--search-missing way -value maxspeed --in-file osm.db` : search way which doesn't have attribute `maxspeed` .
6. `--search-missing relation -value name --in-file osm.db --out-file missing_relation_name.txt` : search relation which doesn't have attribute `name` and write result into `missing_relation_name.txt` .
7. `--get-relation-name-shortname-missing --in-file osm.db` : search relation that has attribute `short_name` but attribute `name` is missing.

Figure 10: <running instruction in github>

for example:
e.g.1: -h, --help: show usage/command line instruction; also the default behaviour if run with no arguments.

e.g.2: --write-osm-to-database mount_alexander_shire_network.osm --out-file osm.db: This reads from mount_alexander_shire_network.osm and creates a osm.db in the root directory.

e.g.3: --get-ways-speed-change 50 --in-file osm.db --out-file way_speed_change.txt: write results to text file.

e.g.4: --search-missing relation -value name --in-file osm.db missing_relation_name.txt: search relation which doesn't have attribute "name" and write result into missing_relation_name.txt.

Program of comparing OSM with Google:

To build do:

"mvn clean install"

Then input:

"java -cp target/project-1.0-SNAPSHOT-jar-with-dependencies.jar osm.analysis.google.Main"

plus arguments below:

TO GENERATE A DATABASE FROM A *.OSM FILE FIRST

(1) Path of your .osm file (Compulsory)

--osm-read-path

(2) Path to store .db file (Compulsory)

--db-store-path

TO GET A SUMMARY RESULT

(1) Orig/dest input (Compulsory)

--list-of-orig/dest (JSON format)

or

--generate-random-origins (integer number)
--radius-of-dest (unit: km)
--db-read-path

(2) Choose one of the following options to set a threshold (Compulsory)

--dist-diff-reporting-threshold-percent (DEFAULT: 5)

--dist-diff-reporting-threshold-km (DEFAULT: 20)
--time-diff-reporting-threshold-percent (DEFAULT: 5)
--time-diff-reporting-threshold-hr (DEFAULT: 0.5)

(3) Store the summary analysis to a local CSV file (Optional). If not provided, then the output will be printed to stdout.

--summary-store-path (file path)

TO STORE DETAILED ANALYSIS

(1) (Compulsory) Store detailed analysis by route ID from a summary file

--input-summary-file

--analyze-route-id (int,int,int,...)
--analyze-route-id (int-int)
--analyze-route-id (int,int-int,int,...)

(2) (Compulsory) Define the file path to store detailed analysis

--detail-store-path

8 Test Specifications

Since our program consists of many command line arguments as options, there will be many combinations of these options. So our test cases are trying to explore as many usable combinations as we can to see if the results we get match the ones we expected.

Test Case ID	Req. covered	Test Objective	Preconditions	Steps	Test data	Expected result
01		Convert osm file into SQlite database file	osm file in root directory	cmd: --write-osm-to-database mount_alexander_shire_network.osm --out-file osm.db	mount_alexander_shire_network.osm	Generate a osm.db file in root directory
02		Convert osm file into SQlite database file	osm file in root directory	cmd: --write-osm-to-database test_area_2.osm --out-file osm_2.db	test_area_2.osm	Generate a osm_2.db file in root directory
03		Print in console: ways which maxspeed exceed the relation it within	osm.db in root directory	cmd: --get-ways-relation-speed --in-file osm.db	osm.db	List of ways that have this problem in console
04		Write to file: the speed difference between two adjacent ways is greater than 50	osm.db in root directory	cmd: --get-ways-speed-change 50 --in-file osm.db --out-file way_speed_change_50.txt	osm.db	Generate a way_speed_change_50.txt file in root directory
05		duplicate checking	osm.db in root directory	cmd: same as test case 04	osm.db	prompt user file already exist

06		Write to file: the speed difference between two adjacent ways is greater than 60	osm.db in root directory	cmd: --get-ways-speed-change 60 --in-file osm.db --out-file way_speed_change_60.txt	osm.db	Generate a way_speed_change_60.txt file in root directory
07		Print in console: all ways that value “maxspeed” are missing	osm.db in root directory	cmd: --search-missing way -value maxspeed --in-file osm.db	osm.db	List of ways that do not have value “maxspeed”
08		Write to file: all relation that value “name” are missing	osm.db in root directory	cmd: --search-missing relation -value name --in-file osm.db missing_relation_name.txt	osm.db	Generate a missing_relation_name.txt file in root directory
09		Print in console: ways which maxspeed exceed the relation it within	osm_2.db in root directory	cmd: --get-ways-relation-speed --in-file osm.db	osm_2.db	List of ways that have this problem in console
10		Write to file: the speed difference between two adjacent ways is greater than 50	osm_2.db in root directory	cmd: --get-ways-speed-change 50 --in-file osm_2.db --out-file test2_way_speed_change_50.txt	osm_2.db	Generate a test2_way_ Speed_change_5 0.txt file in root directory
11		find information where relation tag contain bicycle_yes, while its ways tag contain bicycle_no or do not tag bicycle	osm_2.db in root directory	cmd: --get-mismatch-bicycle-way-relation --in-file osm_2.db	osm_2.db	List of ways and relations in console
12		Read a list of JSON-formatted orig/dest	none	cmd: --list-of-orig/dest "{routes:[{ori:{lat:-37.840239,lon:145.0550085},dest:{lat:-34.2968327,lon:144.0660916}}]}”	"{routes:[{ori:{lat:-37.840239,lon:145.0550085},dest:{lat:-34.2968327,lon:144.0660916}}]}”	Print summary analysis result
13		Generate a list of random orig/dest pairs	osm.db in root directory	cmd: --generate-random-origins 10 --radius-of-dest 800 --db-read-path osm.db --time-diff-reporting-threshold-hr 0.1	osm.db	Print summary analysis result
14		Read from a summary analysis file and write a detailed analysis file	test_summary.csv file in root directory	cmd --input-summary-file test_summary.csv --analyze-route-id 2-4,6 --detail-store-path test_detail.csv	test_summary.csv	Generate the test_detail.csv file containing full analysis details

9 Testing Results

We did all our testings both under windows and unix environments. We have created two databases as input file to test all our functions. We've tested as many possible scenario as we can. Also we have duplicate check and wrong input check, the program could catch any kind of exception we expected.

10 Other Considerations

API key's restriction

The OSM & Google comparison program relies on a Graphhopper API key and a Google API key to send requests. The Graphhopper API key String which is already hard-coded in program code supports 500 requests per day, you can replace that String in Sender.java - osm() and Sender.java - osmSimple() with your own key to achieve more requests when running the program. As for Google key in Sender.java-goo() and Sender.java - gooSimple(), we're providing our own API key temporarily. So we will cancel the validation of it later. Make sure to replace it with your own Google API key if the hard-coded one doesn't work anymore.

11 References

GitHub. (2018). *agentsoz/osm-analysis*. [online] Available at: https://github.com/agentsoz/osm-analysis [Accessed 8 Jun. 2018].

OpenStreetMap. (2018). *OpenStreetMap*. [online] Available at: https://www.openstreetmap.org/ [Accessed 8 Jun. 2018].

En.wikipedia.org. (2018). *OpenStreetMap*. [online] Available at: https://en.wikipedia.org/wiki/OpenStreetMap [Accessed 8 Jun. 2018].

Sqlite.org. (2018). *SQLite Home Page*. [online] Available at: https://www.sqlite.org/index.html [Accessed 8 Jun. 2018].

Sqlitebrowser.org. (2018). *DB Browser for SQLite*. [online] Available at: https://sqlitebrowser.org/ [Accessed 8 Jun. 2018].

Wiki.openstreetmap.org. (2018). *Elements - OpenStreetMap Wiki*. [online] Available at: https://wiki.openstreetmap.org/wiki/Elements [Accessed 8 Jun. 2018].

Wiki.openstreetmap.org. (2018). *Relation - OpenStreetMap Wiki*. [online] Available at: https://wiki.openstreetmap.org/wiki/Relation [Accessed 8 Jun. 2018].

Wiki.openstreetmap.org. (2018). *Node - OpenStreetMap Wiki*. [online] Available at: https://wiki.openstreetmap.org/wiki/Node [Accessed 8 Jun. 2018].

Wiki.openstreetmap.org. (2018). *Way - OpenStreetMap Wiki*. [online] Available at: https://wiki.openstreetmap.org/wiki/Way [Accessed 8 Jun. 2018].