1.The modular structure of your code

1.1 module and function

**1.1 Function module: this module is used to collect data from the map file**

①double round6(double d); these function is used to find the absolute value with 6 decimal number

②int findstring(char\* str1,char\* str2); these function is used to find the specific “string” from file

③int findstring2(char\* str1, char\* str2); because in the <link>, there are two same string “id”. The function will find the second certain “id” after the same string “id” in the file

④long trunc2int(char total[], char left[], char right[]);these function is used to read the id of each nodes.

⑤long trunc2int2(char total[], char left[], char right[]);these function is used to read the first node’s id in each path.

⑥long trunc2int3(char total[], char left[], char right[]);these function is used to read the second node’s id in each path.

⑦double trunc2float(char total[], char left[], char right[]);these function is used to read the longitude and latitude of each nodes and the distance of each path. Both of them are type float.

⑧int getnewnodeid(long oldnodeid); because the data’s id is too big, this function is used to assign the old id to a new and small one by the input’s order.

1.2 algorithm: **this module is used to find the closest path between two nodes.**

①void dijkstra(long src);the main dijkstra algorithm

②int minDistance(long src,int sptSet[]);find the closest node for entered node

③int isalonenode(int newnodeid);this function is used to judge the node is alone or not, which can save the time if the user enter two ids has no link.

**1.3 initialization: this module are used to initialize all array and various and then assign them data from file.**

①void initial(); this function is used to assign 0 to all array and various in the data struct.

②void readmapfile(); this function is used to read file from map and assign the value to each array and various in the data struct.

③void writemapfile();this function is used to write basic node and line from data struct as the background of the map.

**1.4 Output: this module is used to write data of closest path to file which can print the map in gunplot.** **(a visual platform)**

①int writePath(int inputnode1,int inputnode2); this function is used to write the closet path with algorithm

②void printMap(); this function is used to start to print the whole map by gnuplot

**1.6 data struct: this module is used to store the data and represent them in proper way.**

①struct Node{

long oldID; the id of node

double x; the latitude of node

double y; the longitude of node

}node[NODENUM];

②struct Link{

long oldID; the id of the link

int from,to; two node in the link

}link[NODENUM];

③struct Path{

double distance; the distance between two nodes

int route; the node which is closest to the second index of path

}path[NODENUM][NODENUM];

1.7 headers



①Function.h is functions which parse data from file and the core algorithm, including the algorithm module and function module.

②Ini.h is used to assign 0 to various and array and read and write map directly before using algorithm, including the initialization module

③Output.h is used to start the algorithm and print the map, including the output module.

④ShortestPath.h is main data struct, including the data struct module.

2. Your project

2.1 how did you plan your work at a high level?

This project of find path can be divided into 3 parts.

The first is to get data from map file, create a proper data struct to store data and print the whole nodes and links.

The second part is calling an algorithm to find the closest path.

The third is writing the data in an appropriate way which the gnuplot can read.

2.2 what were the design iterations you used?

Version 1

Firstly, we try to analysis the input file. According to the format of data, we will complete the function which can read and succeed to get the necessary information from input file. At the same time, these data which we need will be stored into an output file.

When the version is finished, we can use the Gnuplot to see and check the map.

Version 2

In this version, we complete the core algorithm, which can find the closest path between two nodes. Meanwhile, the functions and arrays which represent the data correctly will be finished with the proper control of algorithm.

During this version, it is hard to me to decide choose which algorithm to be used, a or b. Finally, I complete the a and test the speed is so slow (nearly 1 minutes).and I finish b, the program with which just need less than 1 second.

Version 3

In this version, the write module will be developed. When the closest path is found, the function will store it in a formula which the Gnuplot can read and display.

When the version is finished, we can use the Gnuplot to see the closest path in the map. However, there may some errors in map itself or the input node.

Version 4

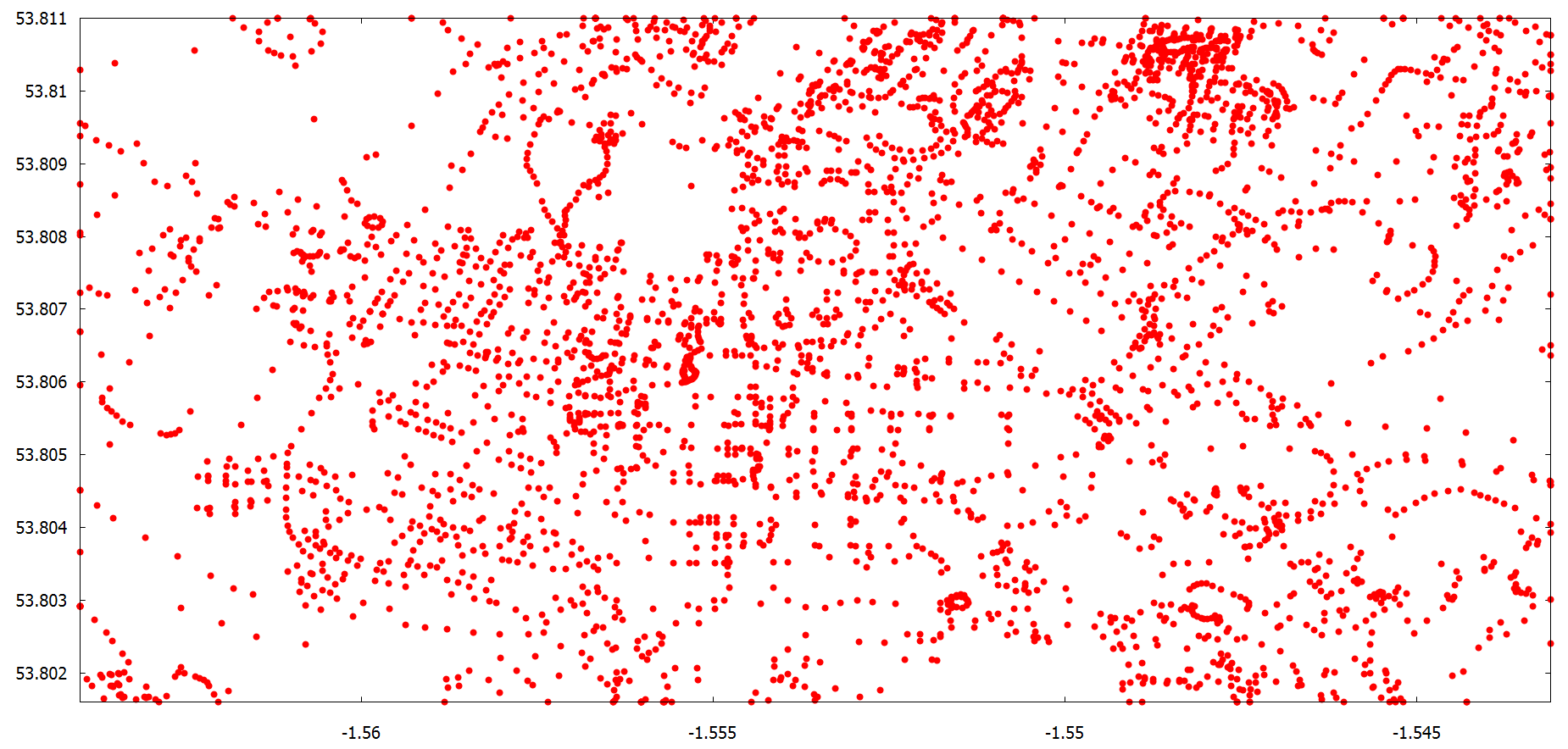
In this version, we will consider another possible situation that may happen when using. For example, the user writes the mistake id of node or the two nodes have no link between them. For those situations, the program should throw the exception waring and print the necessary information. Finally, we will polish the user interface module and check each case may cause the errors.

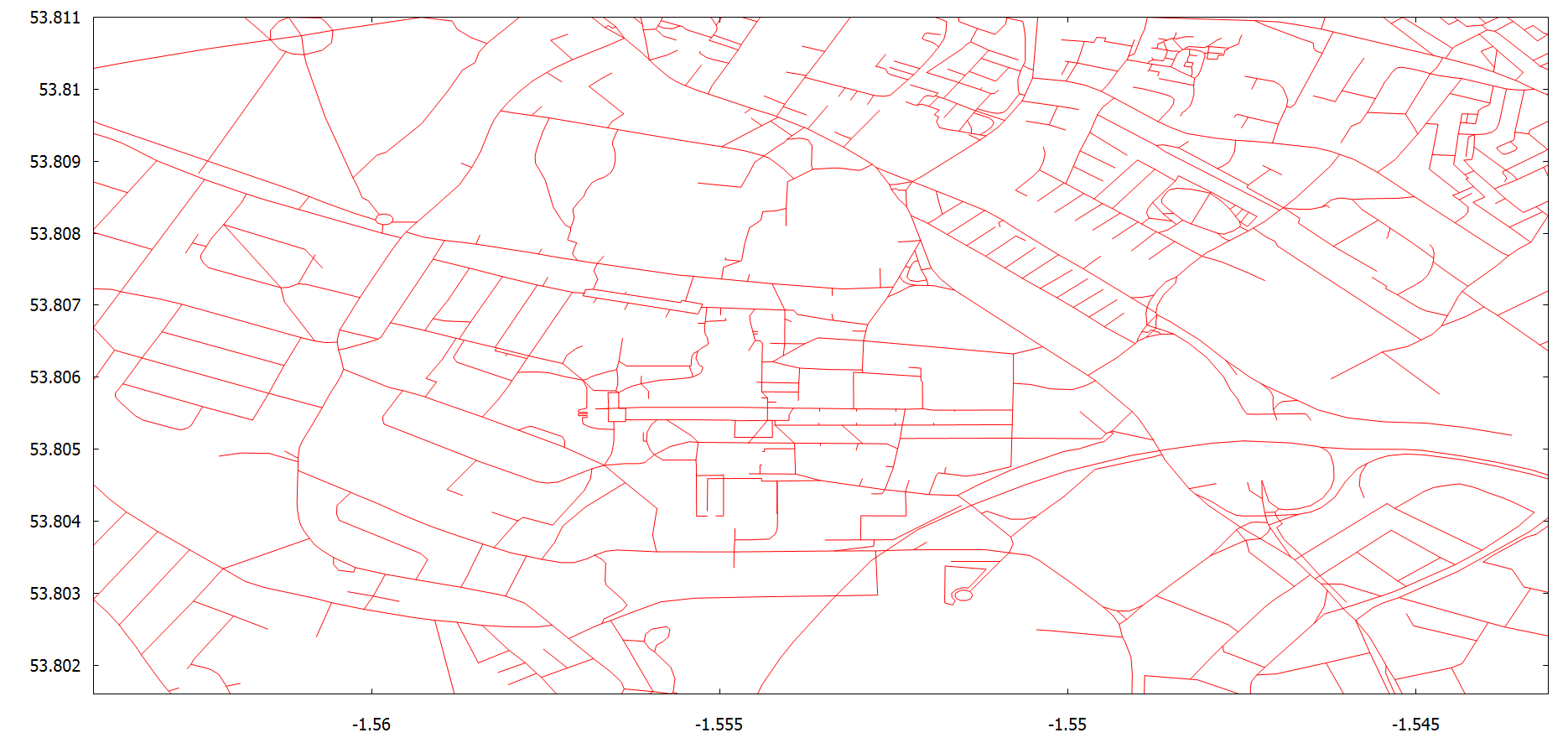
2.3 what were the specific application features associated with each iteration?

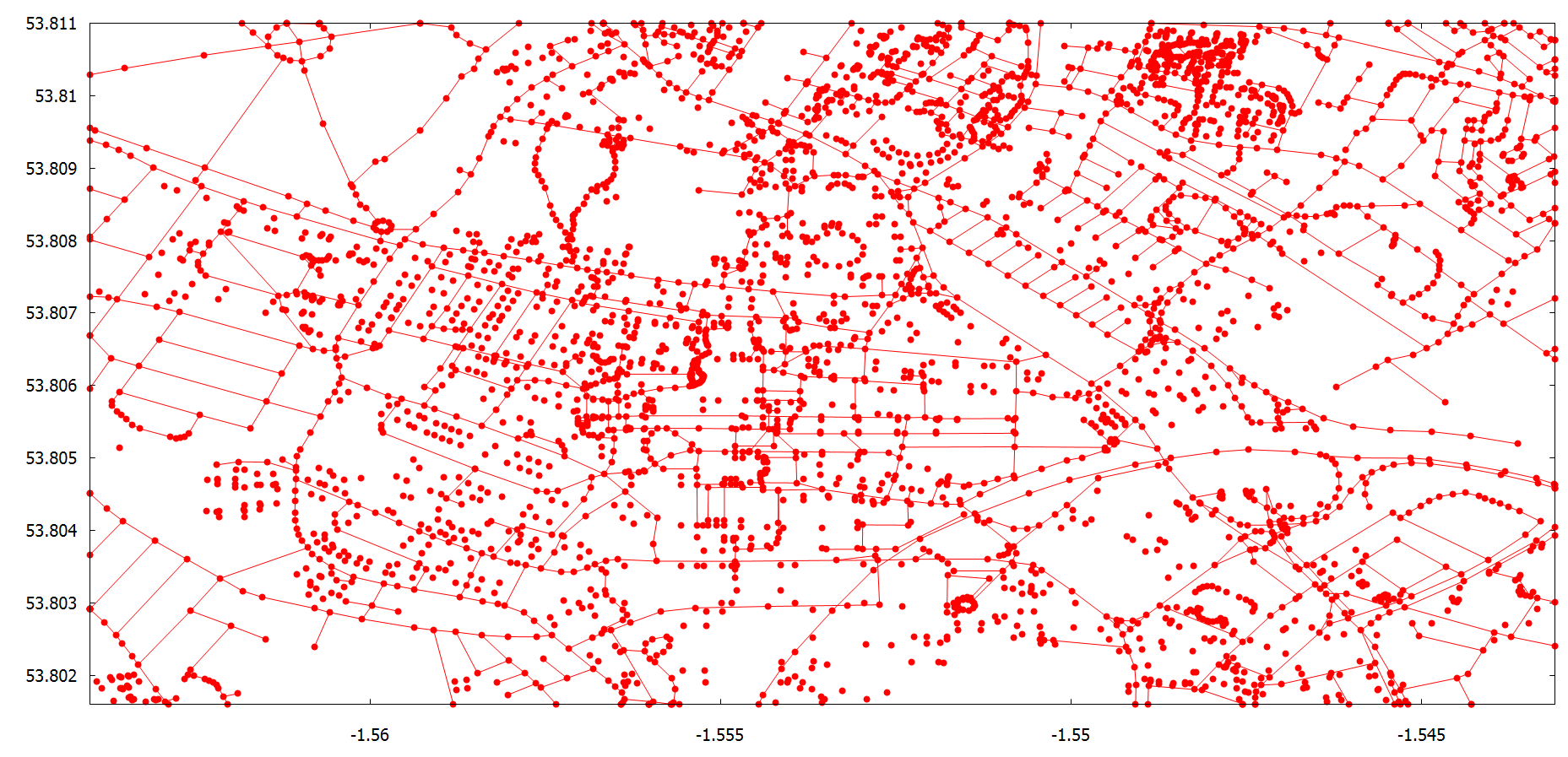
Version1:

3. Test cases and results

3.1 test for initialization

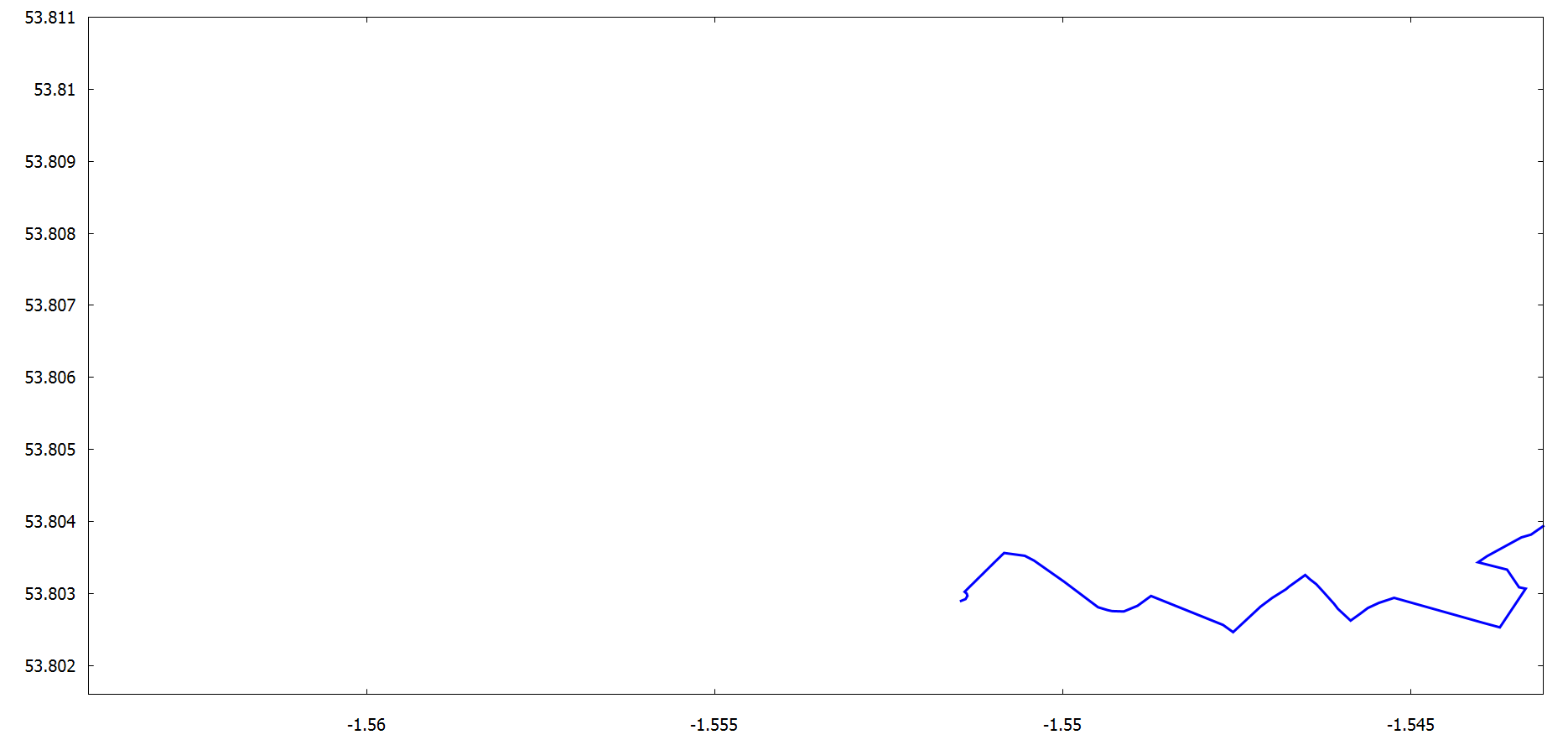


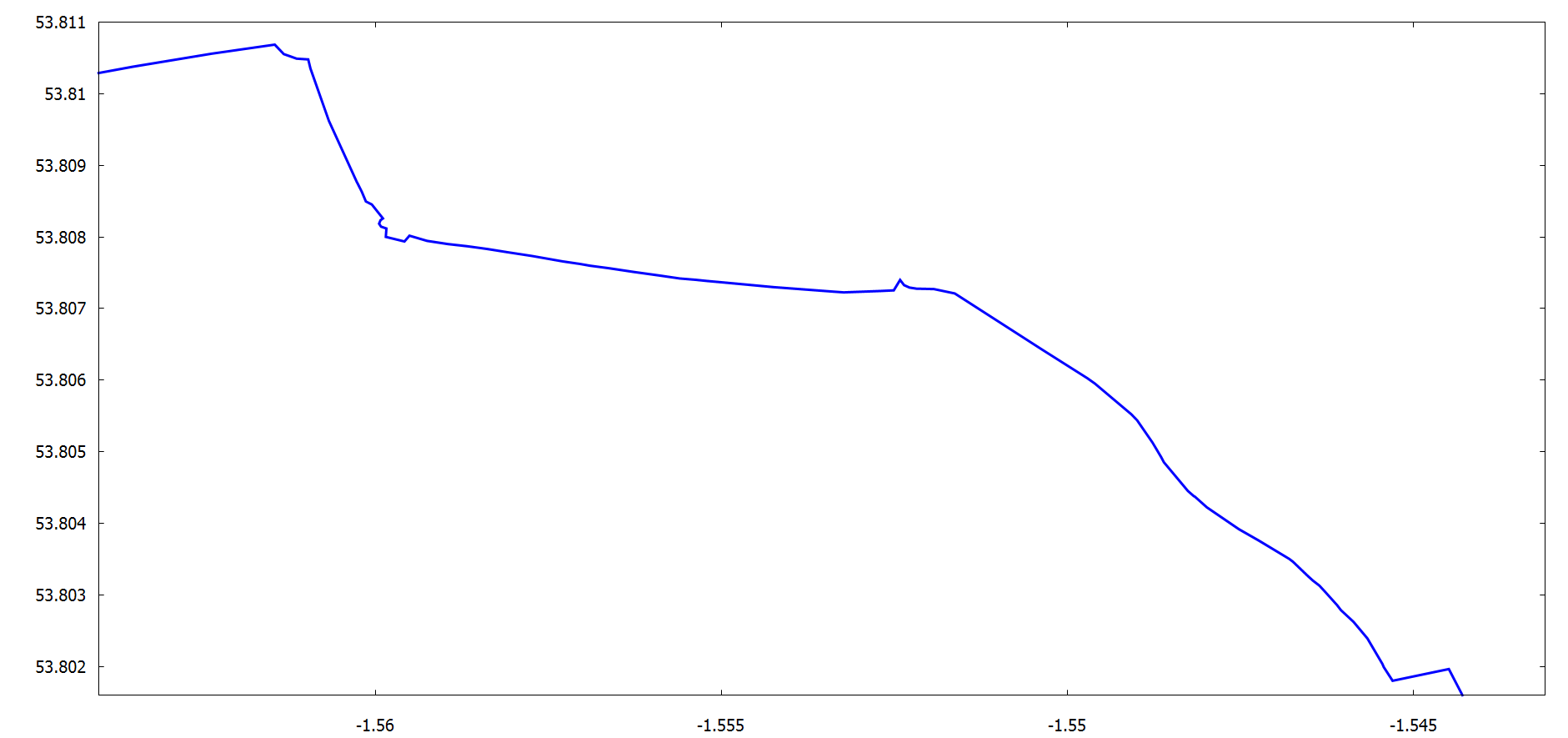




3.2 test for the core algorithm

3.3 test for write modules





3.4 test for malloc

3.5 test the results

