

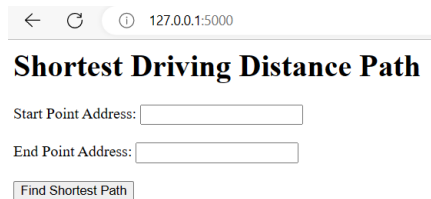
Data Structures Lab Course (start of summer 2023)

Weekly report 2

Learnings and tasks for week 2:

- The topic was refined, to make it more practical, the shortest distance in this project means shortest driving distance, therefore the visualization is based on driving path.
- Implementation of the program's core area, starting the user interface python file and html pages. Core codes were completed with basic framework, but need to be improved.
- Went through the materials: [yksikkötestaukseen tukimateriaalissa](#). Comprehensive unit testing of the code was planned, but not tested yet.

This week first step was to develop a functional user interface, which is important from the beginning. Python flask library was installed to develop the user interface web application, which is a simple page but allows users to input the starting point's address and destination point's address on the page. Currently the page is working on local server as shown in below picture:



← ↻ 127.0.0.1:5000

Shortest Driving Distance Path

Start Point Address:

End Point Address:

Secondly, the open question from last week “how to convert street address into a data format which supports the algorithms’ operation”, a good solution is to translate address into latitude and longitude data, so use geocoding as the process of converting addresses, which are critical for calculating distances and plotting locations on a map.

I practiced how to do retrieving data from the OpenStreetMap website. For example, addresses are used to convert into geographic coordinates (longitude and latitude) using the OpenStreetMap Nominatim API. OpenStreetMap Nominatim API is a good geocoding service. It allows to convert addresses into geographic coordinates using simple HTTP requests. Therefore, geocode function was added to the py file, in order to send requests to the map API to retrieve the coordinates for an address. The target here is to return the coordinates as a tuple (including longitude and latitude).

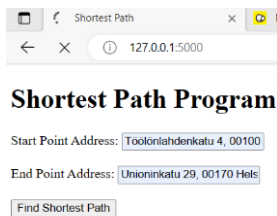
One of the next challenges are to visualize the route found and the nodes/jumping points passed right from the start. In order to integrate the path from map and visualization tasks, Python visualization libraries have been studied and compared, for instance, matplotlib was used in the codes, which failed to generate desired visual picture.

Thirdly, implementing the path finding operations was based on Dijkstra's algorithm in the function `def find_shortest_path()`. According to the updated specification document, IDA* vs Dijkstra will be compared for this project work. IDA* will be implemented next week.

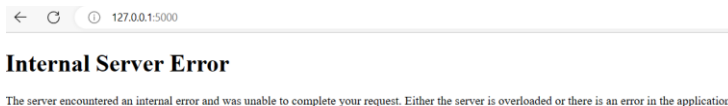
In conclusion, the entire program contains 4 major core operations: user interface handling user input, geocodes addresses, finds the shortest path with selected algorithm, and displays the result on a map.

How to start up the program:

In the Windows cmd, go to the file's directory, run the command `py Shortest_path.py` to start the Flask web server. The homepage will be visible when open web browser with address `http://localhost:5000`. Users should see the Shortest Path Finder page with the input fields for the start and end points. For example, it was tested that the html pages work well.



When run the application, below error displayed. Therefore, next week need to simply and improve the codes to make path results visualization possible.



Testing:

For the unit testing, below identified units will be tested during next 3 weeks, testing document will be started up:

`def home():`

`def geocode():`

`def find_shortest_path():`

`def plot_shortest_path():`

Hopefully testing can also help to optimize the system based on the results of testing and profiling.

Working hours

Date	Hour	Content
22.5.2023	2	Refined project structure. Start the user interface in Visual Studio.
26.5.2023	1	User interface work (index page etc.), tried the Tkinter library, but gave up and went with flask lib
	2	tried to implement Dijkstra's algorithm and Iterative Deepening A* algorithm.
27.5.2023	1,5	OpenStreetMap Nominatim API - convert addresses to nodes in the weighted graph.
	5	visualize the nodes and graph, visualization libraries in Python.
28.5.2023	4	coding the core part
	2	planned unit testing, draft weekly report 2, update the specification document
Total	17,5	

Sources:

Map representations: <http://theory.stanford.edu/~amitp/GameProgramming/MapRepresentations.html>

Implementation: <http://theory.stanford.edu/~amitp/GameProgramming/ImplementationNotes.html>

Open Street Map: <https://www.openstreetmap.org>

Geocoding: https://en.wikipedia.org/wiki/Address_geocoding & <https://nominatim.org/release-docs/develop/api/Overview/>

Open-source geocoding with OpenStreetMap data: <https://nominatim.org/>

Unit testing: [Testing-and-rmq/src/test/java/rmq/domain at master · TiraLabra/Testing-and-rmq · GitHub](#)

Matplotlib: Visualization with Python: <https://matplotlib.org/>