Data Structures Lab Course (start of summer 2023)

Weekly report 3

Learnings and tasks for week 3:

- IDA* algorithm was implemented and compared with Dijkstra's algorithm.
- Folium library was tested again, combined with Polyline library.
- Path results visualization successful, but still need improvements.
- Started with the Testing document.
- Code quality monitoring was added.

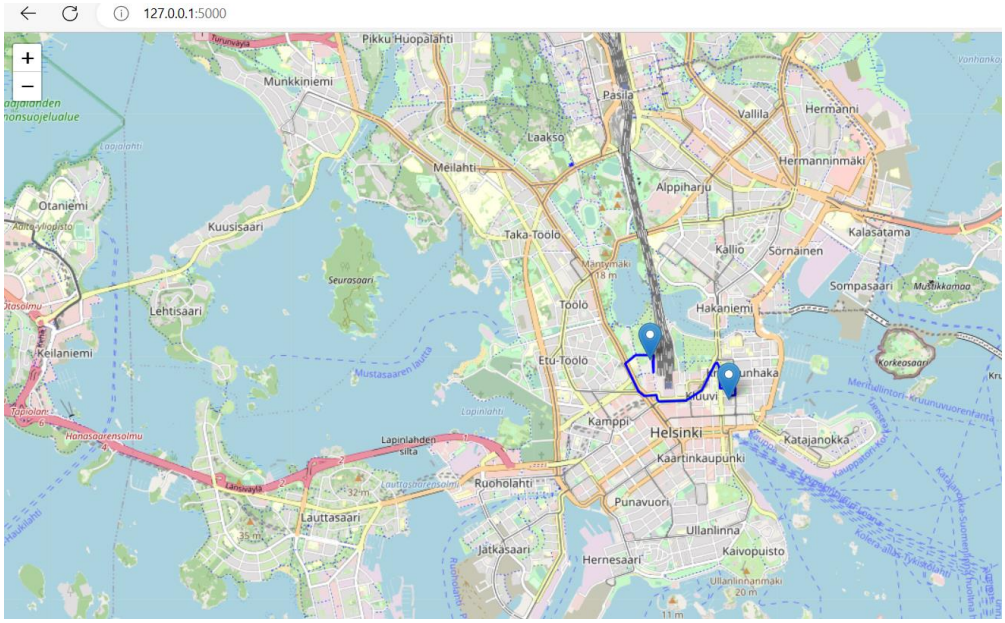
Iterative Deepening A* (IDA*) Algorithm uses a threshold value, which is an estimation of the cost from the start node to the target node. Heuristic function is needed to provide an estimate for the remaining cost. The search is iterative, need to increment the threshold until the solution is found. It explores path in a depth-first approach. It requires less memory compared to Dijkstra's algorithm. Time complexity of IDA* seems to be very uncertain, as few factors would impact on the time complexity, such as how accurate heuristic estimations is, how the graph branches, etc. Therefore, it took much longer time to run IDA* based Shortest_path.py file. Program didn't display results after 10 min, therefore the program run was terminated manually.

Dijkstra's algorithm takes all unvisited nodes into consideration, selecting the shortest distance node as the next node. As it maintains a priority queue, so requires additional memory to store all the nodes.

Based on above observation, it was decided that Dijkstra's algorithm will be used as the final algorithm to enable program display found path and outcomes via html files. This week's py file return was based on IDA* algorithm.

Folium library was tested by integrating with Polyline library, it was able to visualize graphical data with the current code version. But after modifying few lines, it failed to render with browser. Below addresses were used to verify if the program work as it is required.

Lentoasemantie 1, 01530 Vantaa
Unioninkatu 29, 00170 Helsinki
Töölönlahdenkatu 4, 00100 Helsinki
Poromiehentie 1, 96200 Rovaniemi



Code quality monitoring:

Code quality monitoring tool Pylint in Visual Studio was enabled this week. Regularly running Pylint can help to improve the quality of code, and maintain consistent coding standards.

Working hours:

Date	Hour	Content
29.5.2023	1	implement Iterative Deepening A* algorithm, compare with Dijkstra's algorithm.
30.5.2023	2	implement Iterative Deepening A* algorithm, compare with Dijkstra's algorithm.
31.5.2023	3	using Folium and PolyLine library replacing matplotlib.pyplot library
2.6.2023	2	unit testing & Testing document
3.6.2023	1	code quality monitoring
4.6.2023	1	draft weekly report 3
Total	10	

Sources:

Pylint: https://learn.microsoft.com/en-us/visualstudio/python/linting-python-code?view=vs-2022

Polyline library: https://pypi.org/project/polyline/