Simon Rödén reporting on Lukas Sjöstrand and Malte Kasemo

Section 1: Core assignment

- Q1: Does the application run? --> yes
- Q2: Does the application display the complete map of tram lines? --> yes
- Q3: Is it possible to query shortest path between any two points? --> yes

Section 2: Optional tasks

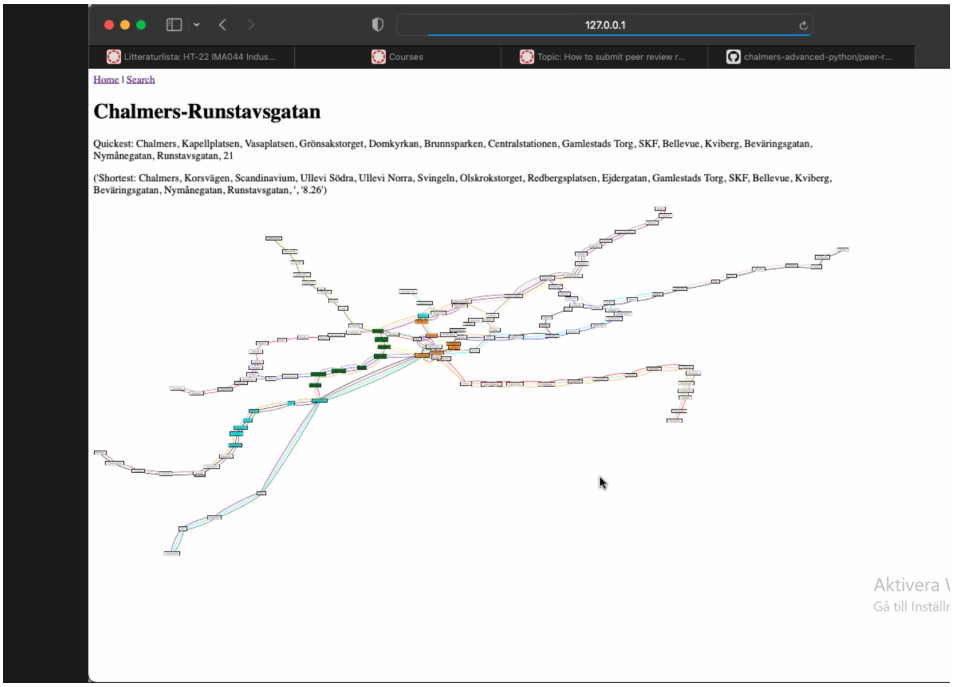
- B2: Is the submission successfully accounting for Bonus Part 2? --> yes

Section 3: Code quality

The code has been implemented from lab 2 without any problems or unnecessary repetitions. Dijsktra works well and the code uses different functions to calculate cost and distance.

Section 4: Screenshots:

Screenshot 1



screenshot 2

```python
def show_shortest(dep, dest):
    network = readTramNetwork()
    departure, destination = getting_objects_by_names(network, dep, dest)
    quickest_path = dijkstra(network, departure, network.get_weight)[destination]['path']
    quickest_path_time = dijkstra(network, departure, network.get_weight)[destination]['dist']

    qp = []
    for object in quickest_path:
        qp.append(object._name)
    qp.reverse()

    shortest_path = dijkstra(network, departure, network.geo_distance)[destination]['path']
    shortest_path_distance = dijkstra(network, departure, network.geo_distance)[destination]['dist']
    shortest_path_distance = round(shortest_path_distance, 2)

    sp = []
    for object in shortest_path:
        sp.append(object._name)
    sp.reverse()


    # First you need to calculate the shortest and quickest paths, by using appropriate
    # cost functions in dijkstra().
    # Then you just need to use the lists of stops returned by dijkstra()
    #
    # If you do Bonus 1, you could also tell which tram lines you use and where changes
    # happen. But since this was not mentioned in lab3.md, it is not compulsory.

    shortest = [departure, 'Chalmers', destination]

    timepath = 'Quickest: ' + ', '.join(qp) + ', ' + str(quickest_path_time)
    geopath = 'Shortest: ' + ', '.join(sp) + ', ' + str(shortest_path_distance)
```

```python
def colors(v):
    if v in qp and v in sp:
        return 'cyan'
    elif v in qp:
        return 'orange'
    elif v in sp:
        return 'green'
    else:
        return 'white'


# this part should be left as it is:
# change the SVG image with your shortest path colors
color_svg_network(colormap=colors)
# return the path texts to be shown in the web page
return timepath, geopath
```