**Peer review - Rasmus Bäckström on Othman**

Section 1: Core assignment

?Q1: Does the application run
.The application runs
?Q2: Does the application display the complete map of tram lines
.Yes, the application maps the network
?Q3: Is it possible to query shortest path between any two points
It is possible to query the shortest path, however only the blue path is displayed. The blue path is supposed to show which paths are included in both the quickest and the shortest path, however it is the only path being displayed. There is no green or yellow lines respectively, therefore something is wrong in how the code looks to display the colors for the different routes. However the Dijkstra algortihm seems to be implemented well since both the distances in the shortest path and times in the .quickest path are displayed accurately

:Section 2: Optional tasks
.No bonus tasks has been adressed

Section 3: Code quality
In general the code looks good and the code from lab 2 has been properly reused. The code avoid boilerplate code since there's not any necessary recursion going on. As mentioned above the Dijkstra algortihm seems to work properly with a correct assessment of the cost function, thus giving the proper paths for the quickest and shortest paths. As mentioned earlier, one thing that is lackluster is that the coloring .for the different paths doesn't work as intended

In general I think that the code is well structure in terms of object orientation, however when looking at certain functions from lab 1 (calculate travel time in particular), there's a few variables which are named a little bit unclear. One improvement would be to try to be as concise as possible with what the function wants to accomplish and which the belonging variables should be properly named to avoid confusion. Also there's almost no comments in the code, which would make it .clearer for the viewer to understand each step of that particular section of the code

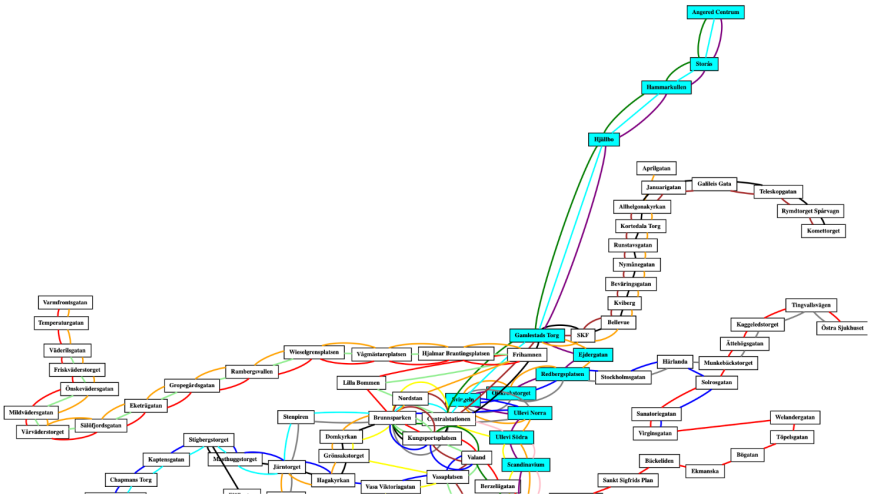
:Section 4: Screenshots
Screenshots are posted but since I am on a Mac it was hard to get really good .screenshots, however the information is all there

## Chalmers-Angered Centrum

**Red - Departure Yellow - Destination Orange - Shortest route Green - Shotest route Blue - Both routes**

**Quickest: Kapellplatsen, Vasaplatsen, Grönsakstorget, Domkyrkan, Brunnsparken, Centralstationen, Gamlestads Torg, Hjällbo, Hammarkullen, Storås, Angered Centrum. Time it took: 22.00 min.**

**Shortest: Korsvägen, Scandinavium, Ullevi Södra, Ullevi Norra, Svingeln, Olskrokstorget, Redbergsplatsen, Ejdergatan, Gamlestads Torg, Hjällbo, Hammarkullen, Storås, Angered Centrum. Distance traveled: 13.420 km.**




```python
def show_shortest(dep, dest, cost=lambda u, v: 1):

    network = readTramNetwork()


    distance_geo, shortpath_geo = dijkstra_algorithm(network, dep, cost_func=network.geo_distance)
    shortest_geo = shortpath_geo[dest]
    distance_traveled_geo = distance_geo[dest]

    # Find quickest path based on travel time
    time, quickpath = dijkstra_algorithm(network, dep, cost_func=network.calculate_travel_time)
    quickest = quickpath[dest]
    total_time = time[dest]


    timepath = f'Quickest: {", ".join(quickest)}. Time it took: {total_time:.2f} min.'


    geopath = f'Shortest: {", ".join(shortest_geo)}. Distance traveled: {distance_traveled_geo:.3f} km.'
    def colors(v):
        if v in shortest_geo:
            return 'cyan'
        else:
            return 'white'


    # Color the SVG network with the specified colormap
    color_svg_network(colormap=colors)

    # Return the formatted path texts for display
    return timepath, geopath
```