PEER REVIEW REPORT: ERIK BRECHTER

By Niclas Lindmark

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

B1: Is the submission successfully accounting for Bonus Part 1? --> **no**

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

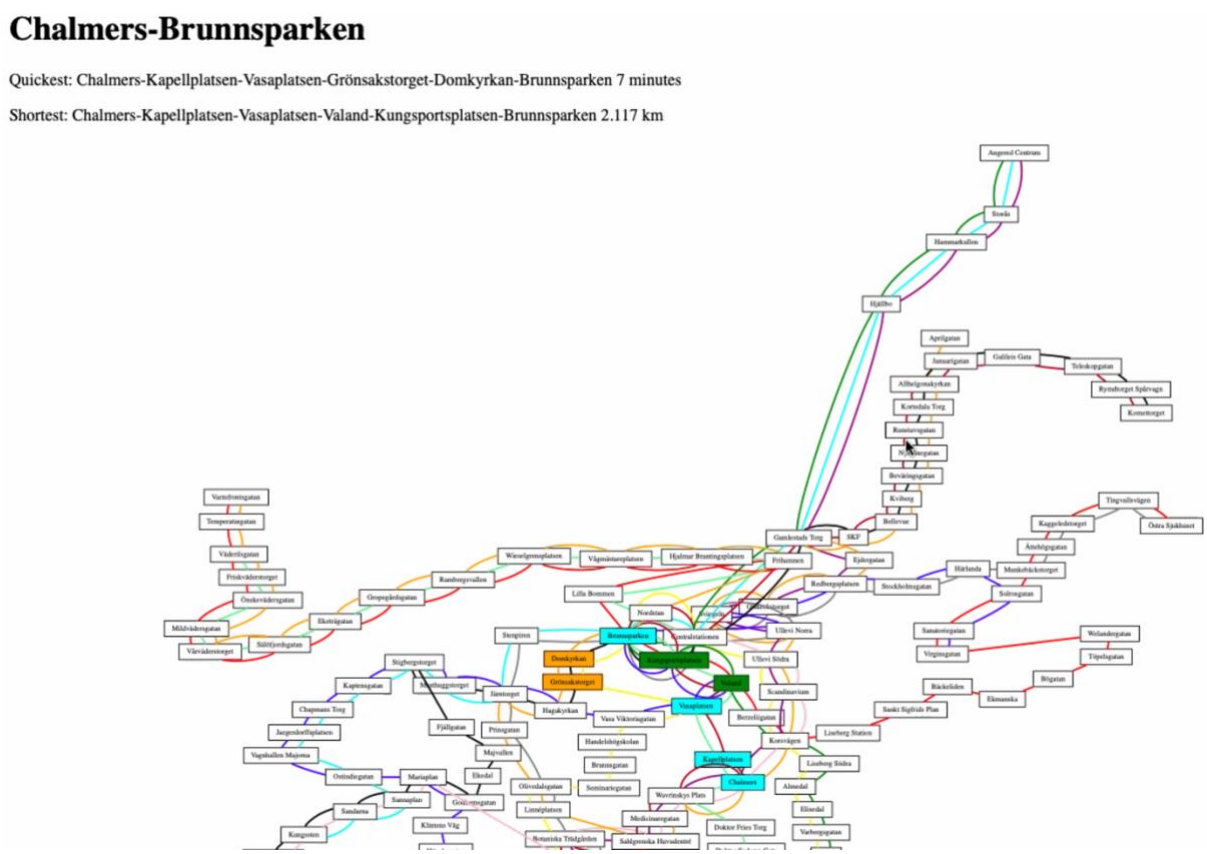
SECTION 3: CODE QUALITY

Dijkstra has been implemented as intended with the Priority Queue version. Different distances are obtained by changing the cost function and the function is only defined once.

Lab2 code has simply been copied over and changed where necessary. The code is efficient.

SECTION 4: SCREENSHOTS INSERT TWO SCREENSHOTS:

SCREENSHOT 1 Correctly displays the shortest path and quickest path from Chalmers to Brunnsparken.



SCREENSHOT 2 Screenshot of show_shortest() function.

```
def show_shortest(dep, dest):
    network = readTramNetwork()

    quickest = dijkstra(network, dep, cost=lambda u, v: network.get_weight(u, v))[dest]
    shortest = dijkstra(network, dep, cost=lambda u, v: network.geo_distance(u, v))[dest]

    distance = 0
    for i in range(len(shortest) - 1):
        distance += network.geo_distance(shortest[i], shortest[i + 1])

    time = 0
    for i in range(len(quickest) - 1):
        time += network.get_weight(quickest[i], quickest[i + 1])

    timepath = 'Quickest: ' + "-".join(quickest) + " " + str(time) + " minutes"
    geopath = 'Shortest: ' + "-".join(shortest) + " " + str(distance) + " km"

    def colors(v):
        if v in shortest and v in quickest:
            return 'cyan'
        elif v in shortest:
            return 'green'
        elif v in quickest:
            return "orange"
        else:
            return "white"

    color_svg_network(colormap=colors)
    return timepath, geopath
```