Portfolio Project 3

The course portfolio consists of three parts:

Part 1: on object-oriented programming and software engineering.

Part 2: on databases and graphical user interfaces.

Part 3: on graphs and graph algorithms.

The first two parts are described in previous documents and part 3 is described below.

Submission dates
Submission deadline for the full portfolio: May 2nd at 10AM via eksamen.ruc.dk
The final submission of the full portfolio is limited to 48,000 characters, including spaces.
All students must submit the final version of the full portfolio via eksamen.ruc.dk on May 2rnd

Submission document
The submission should contain a single pdf document for descriptions and a single zip file of source code for all three parts of the portfolio.
The portfolio can be written in Danish or English.

Groups
The maximum group size is 4. Names of all group members should be listed at the first page of each part of the portfolio. You can be in different groups or alone in each part.
You can work on the portfolio in groups but be aware that the exam is individual, and you must be able to answer questions related to all parts of your portfolio.
You are expected to understand the programs to the level where you could have written them yourself, even if you are part of a group.

Material
For all parts, your solution may include code snippets from lectures.
Code, text, or diagrams from any other source should be clearly marked with reference to its origin according to normal rules for plagiarism. The requirement of references is same whether the source are books, articles, internet fora, people or AI tools (eg. ChatGPT or GitHub copilot).

Third part of the portfolio.

This portfolio is about graph algorithms.

The portfolio focuses on finding shortest routes in a graph.

A separate file contains a comma separated list of distances between Danish towns.

This part of the portfolio consists of the following parts.

1. Read the file of the current route network and represent it as a graph

2. Check that the graph is connected so that there is a possible route between any two place names.
3. Find the shorts path from “Helsingør” to “Esbjerg”.
4. Discuss how a priority queue could be used to improve the algorithm.

Part 1

You should read the file of the distances between Danish towns. The list can be found in a file named “distDK.txt” as a comma separated file of two towns and a distance measured in 100 meters. The list is extracted from a list of distances between town halls and should not be used for navigation. You should store the graph as an adjacency list or a matrix graph. The graph should be bidirectional so that any information about connected towns should be stored in both directions.

Part 2

You should check that the graph is connected. You should check that from a node you can reach all other nodes directly or indirectly. Write the check as a general algorithm that works on the graph representation you have used. What is the complexity of the algorithm?

Part 3

Implement an algorithm for finding the shortest path in a graph using Dijkstra’s algorithm. You may use the naïve implementation used in the lectures. You are expected to be able to explain how Dijkstra’s algorithm works. Use it to find the shortest path from “Helsingør” to “Esbjerg” in the graph.

Part 4

Discuss how the efficiency of the algorithm can be improved by using a priority queue. You do not need to implement but you should explain how the efficiency can be improved.