Lab Assignment 4: Graphs

1   Introduction

In this lab you will implement a program to calculate solution for a very practical problem: what is the shortest path to travel from one city to another city. The road map (see Fig 1) shows the roads that directly connect the cities. Each road has a weight that represents the distance of the road.
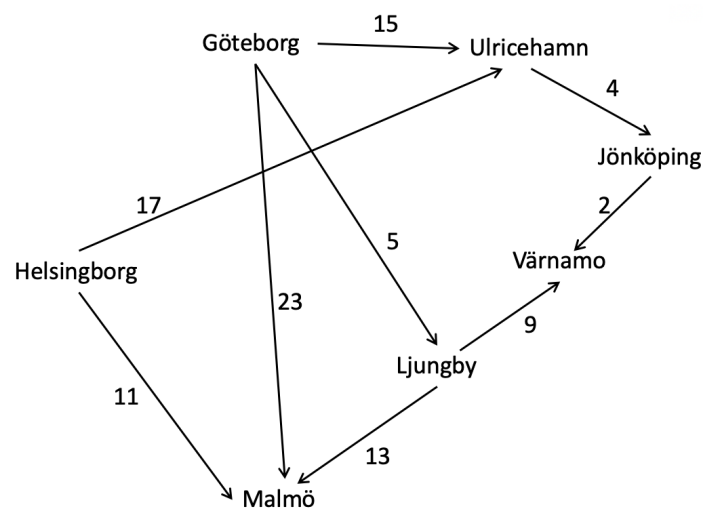

Figure 1: The Road Map

A guide to implement the program is given as below:

- Represent the road map in a weighted directed graph. The cities are represented as vertices (nodes) in the graph and the roads are represented as the edges connecting the vertices.

- Apply Dijkstra's algorithm over the graph to calculate the distances of shortest paths between the cities.

- To make it more efficient, use a min binary heap to organize the priority queue when implement Dijkstra's algorithm.

2   Requirements

You should implement the interfaces for graph ADT and priority queue ADT as defined in the header files and a main function (included in shortest_path_supplied_codes.zip). As defined in the graph ADT, the graph is represented using adjacency list. In the queue ADT, the function *decreaseKey* is to decrease the key value of a node and move up the node until the min heap

property is satisfied. For other functions you can reuse the code implemented for the task 3 in the lab 3, except that a structure is declared for heap node and the key value represents the road distance in this lab. As defined in the main.c file, the function *dijkstra* is to calculate the distances of shortest paths between cities. Once you've got your program implemented correctly, you should be able to produce output like the example shown in Fig 2, where the input from user is 3 (i.e. the starting city is Göteborg).

Figure 2: example program output

Deliverables

The deliverable is a zip file in which the files are organized as in Fig 3. Variables in your code files should have proper names, and the code has to include sufficient comments for readability.
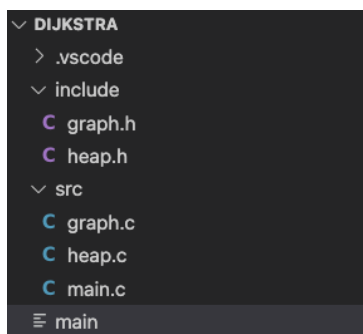

Figure 3: file organization