Lab 6. Graphs

Release: 4th March 2024
Supervision: 12th March 2024
Due: 18th March 2024

Vladimir Tarasov

1 Introduction

In this lab you will implement a program to find a solution to a very practical problem: what is the shortest travelling path from one city to other cities.

2 The Problem

The road map (see Fig. 1) shows the roads that directly connect several cities. Each road has a weight that represents the length of the road.

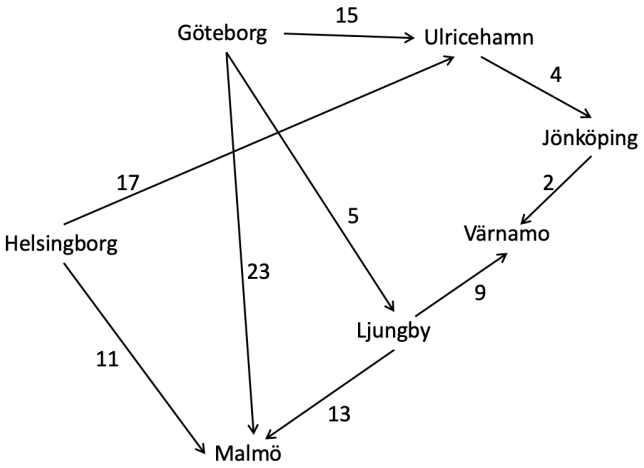

Figure 1: The Road Map

A guide to implement the program is given as below.

Details of the implementation

1. 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.
2. Apply *Dijkstra's algorithm* to the graph to calculate the distances of shortest paths between the cities.
3. To make it more efficient, use a *min binary heap* to organize the priority queue when implement Dijkstra's algorithm.

3 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 `dijkstra_supplied_codes.zip`). As defined in the graph ADT, the graph is represented using *adjacency list*.

Hint: adapt the program implemented in the exercise for the adjacency list representation of graph.

In the queue ADT (implemented as a *min binary heap*), there is a function, called `decreaseKey`. This function is to decrease the key value of a node and move up the node until the min heap property is satisfied.

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).

```
0: Jönköping, 1: Ulricehamn, 2: Värnamo, 3: Göteborg, 4: Helsingborg, 5: Ljunby, 6: Malmö
Enter the city :  3

The distance of the shortest path for travelling from Göteborg to
Jönköping is 19
Ulricehamn is 15
Värnamo is 14
Göteborg is 0
Helsingborg !!! no connection between these two cities
Ljungby is 5
Malmö is 18
```

Figure 2: example program output

4 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.

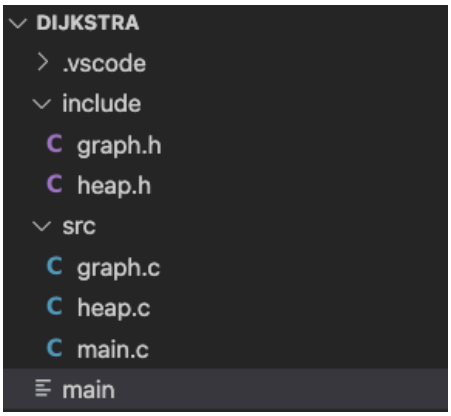

Figure 3: file organization

The provided header files *must be included* in your implementation. It is up to you to decide how you code the `main()` function.

The source code has to be thoroughly tested as well as to adhere to the recommendations on *Standard C and portability*, which you can find in Canvas.

You can compile your code using more strict flags `-Wall` together with `-pedantic` before submitting the code, e.g.

```
$ gcc -Wall -pedantic program.c -o program
```

Please include a comment in your program (all `.c` files) with the year and your name to indicate authorship:

```
// 2024 Your Name
```