Fall 2017 CSE 440 - Assignment 1

Due date: Sunday 3rd, October, 2017

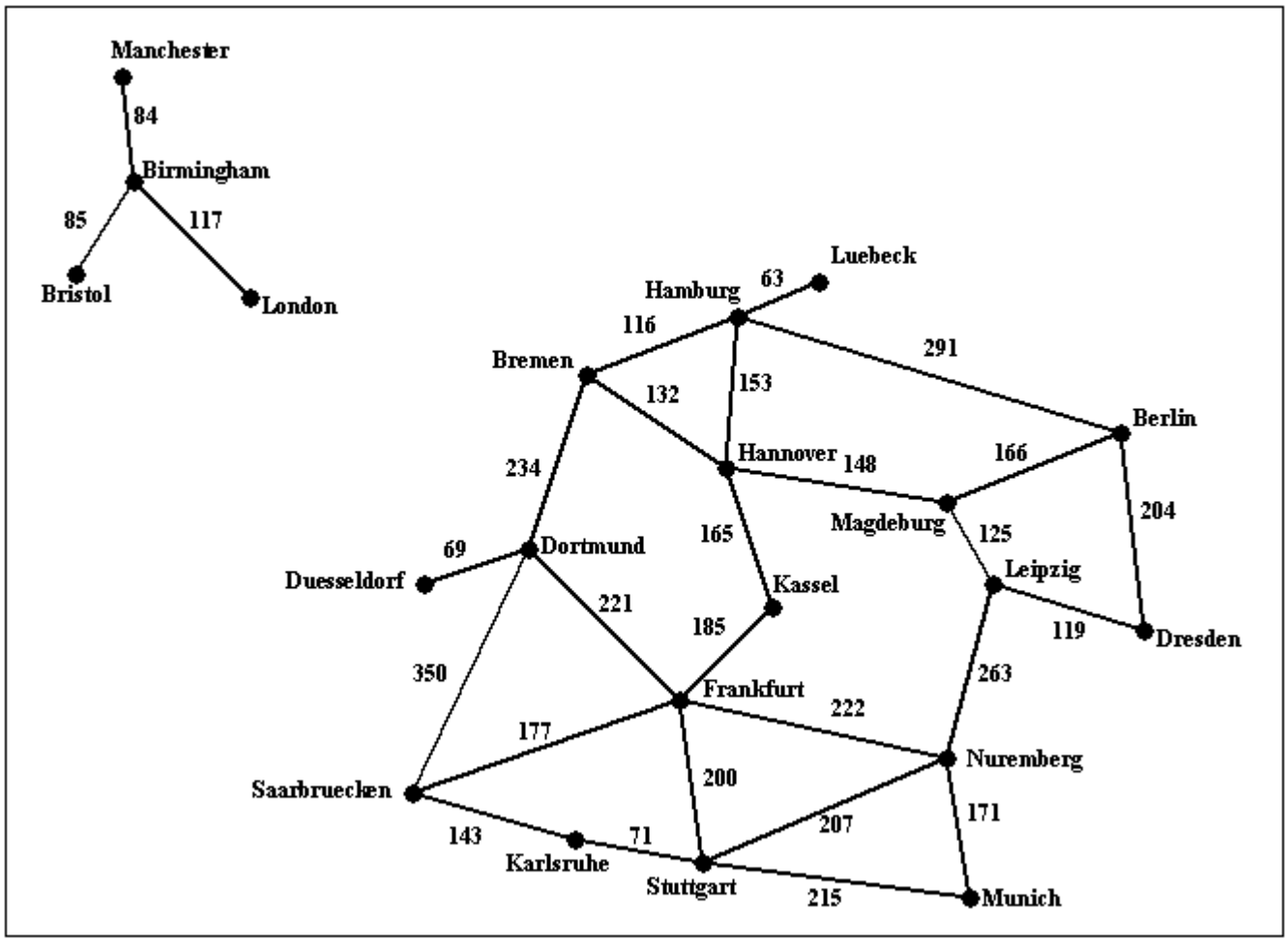

Figure 1: A visual representation of the road system described in file input1.txt.

Task: Uniform Cost Search

Implement uniform cost search for a program that can find a route between any two cities. Your program will be called `find_route`, and will take exactly three commandline arguments, as follows:

```
find_route input_filename origin_city destination_city
```

An example command line is:

```
find_route input1.txt Munich Berlin
```

Argument `input_filename` is the name of a text file such as input1.txt, that describes road connections between cities in some part of the world. For example, the road system described by file input1.txt can be visualized in Figure 1 shown above. You can assume that the input file is formatted in the same way as input1.txt: each line contains three items. The last line contains the items "END OF INPUT", and that is how the program can detect that it has reached the end of the file. The other lines of the file contain, in this order, a source city, a destination

city, and the length in kilometers of the road connecting directly those two cities. Each city name will be a single word (for example, we will use New_York instead of New York), consisting of upper and lowercase letters and possibly underscores.

IMPORTANT NOTE: MULTIPLE INPUT FILES WILL BE USED TO GRADE THE ASSIGNMENT, FILE input1.txt IS JUST AN EXAMPLE. YOUR CODE SHOULD WORK WITH ANY INPUT FILE FORMATTED AS SPECIFIED ABOVE.

The program will compute a route between the origin city and the destination city, and will print out both the length of the route and the list of all cities that lie on that route. For example,

```
find_route input1.txt Bremen Frankfurt
```

should have the following:

```
distance: 455 km
route:
Bremen to Dortmund, 234 km
Dortmund to Frankfurt, 221 km
```

and

```
find_route input1.txt London Frankfurt
```

should have the following output:

```
distance: infinity
route:
none
```

For full credit, you should produce outputs identical in format to the above two examples.

Suggestions

Pay close attention to all specifications on this page, including specifications about output format, submission format. Even in cases where the program works correctly, points will be taken off for non-compliance with the instructions given on this page (such as a different format for the program output, wrong compression format for the submitted code, and so on). The reason is that non-compliance with the instructions makes the grading process significantly (and unnecessarily) more time consuming.

Grading

The assignments will be graded out of 100 points.

- 40 points: The program always finds a route between the origin and the destination, as long as such a route exists.
- 30 points: In addition to the above requirement, the program terminates and reports that no route can be found when indeed no route exists that connects source and destination (e.g., if source is `London` and destination is `Berlin`, in the above example).
- 30 points: In addition to the above requirements, the program always returns optimal routes. In other words, no shorter route exists than the one reported by the program.
- Negative points: penalty points will be awarded by the instructor generously and at will, for issues such as: code not running on standard platform, submission not including precise and accurate instructions for how to run the code, wrong compression format for the submission, or other failures to comply with the instructions given for this assignment. Partial credit for incorrect solutions will be given ONLY for code

that is well designed and well documented. Code that is badly designed and badly documented can still get full credit as long as it accomplishes the required tasks.

How to submit

Implementations in C, C++, Java, and Python will be accepted. If you would like to use another language, please first check with the instructor via e-mail. Points will be taken off for failure to comply with this requirement. The assignment should be submitted via gmail (cse440nsu@gmail.com). Submit a ZIPPED directory called assignment1.zip (no other forms of compression accepted, contact the instructor if you do not know how to produce .zip files). The directory should contain source code and input file. The submission should also contain a file called readme.txt, which should specify precisely:

- Name and NSU ID of the student.
- What programming language is used.
- How to run the code, including very specific compilation instructions, if compilation is needed. Instructions such as "compile using g++" are NOT considered specific.

Submit the assignment as an attachment (zipped file) to the email: cse440nsu@gmail.com. The subject of the email must be “Assignment 1, NSU_ID”.
Insufficient or unclear instructions will be penalized by up to 100 Points.

Submission checklist

- Is the code running on a standard platform?
- Did you submit your source code?
- Does the submission include a readme.txt file, as specified?