Professor Dr. Stephan Held
Susanne Armbruster

Programming Exercise 2

Exercise P.2. *Task:* Implement the algorithm for the UNDIRECTED CHINESE POSTMAN PROBLEM.

Usage: Your program should be named `chinese_postman`, and it should be called as follows:

```
chinese_postman input_graph output_tour
```

Input: The arguments `input_graph` and `output_tour` are mandatory, i.e. your program should exit with an error message if they are not present. Here, the file `input_graph` encodes the input graph for which your program should find a shortest Chinese Postman tour, and you should write the postman tour you found (and nothing else) to `output_tour` (see *Output* below).

The file `input_graph` is given in DIMACS format as defined on the previous programming exercise.

Output: Your program should return a shortest Chinese postman tour in G by writing the complete DIMACS encoding of the tour into the file `output_tour`. A tour is encoded as follows in DIMACS format. The encoding starts with a header of the form
`TYPE : TOUR`
`DIMENSION :` n
`LENGTH :` k
`TOUR_SECTION`
where n is the number of vertices in the instance and k is the total length of your tour. Each of the following k lines consists of a single integer i with $1 \leq i \leq n$, which are the vertices in the order in which they are visited in the Postman tour. Please note that for most tours, this includes listing some vertices several times. Finally, the encoding ends with these two lines:
`-1`
`EOF`

Programming conditions: Your program should be written in C or C++, although the use of C++ is strongly encouraged. By default, your program will be compiled using g++ 11.1.0 using C++20. Different compilers or compiler versions are available upon request. Your program will be compiled using `-pedantic -Wall -Wextra -Werror`, i.e., all warnings are enabled and each remaining warning will lead to compilation failure. Program evaluation will be performed on Linux. The standard library as well as a one of the provided libraries for solving the Minimum Weight Perfect Matching Problem (see *Help* below) can be used as you wish. No other libraries are allowed.

Submission Format: Your submission should consist of a single archive file in the .zip, .tar.gz or .tar.bz2 format, which contains all contents of your top level directory (but not the directory itself). For easier testing, your submission must contain a bash script `compile.sh` in its top level directory, which builds the executable (e.g. by directly calling the compiler or by executing some make command) when called without any arguments. Your executable must be called `chinese_postman` (as implied above) and be created in a subfolder called `bin` of the top level directory. Since you will (most likely) be linking against a library for solving the Minimum Weight Perfect Matching Problem, you should also provide a copy of the library located at the correct relative path with your submission.

Algorithm evaluation: The algorithm is to be implemented as described in the lecture by computing a T-join on the all vertices of odd degree and then computing a Eulerian tour in the graph combined with its T-join.

Code evaluation: Your code must implement the Undirected Chinese Postman Algorithm correctly. Running time in practice will also be evaluated, as well as the elegance, cleanness and organization of your code. Make sure to add good documentation and give the variables, functions and types meaningful names that make their role clear. Break your complicated functions into small simple ones, break your program into a few units etc. Of course, your program should not trigger undefined behavior. In particular, your program should be `valgrind`-clean, i.e. it should not leak memory and should not perform invalid operations on memory.

Help: The website for the exercise class contains a set of test instances for testing your code. Moreover, an updated class to store an undirected graph with edge weights, a public solver (free for academic use) for the Minimum Weight Perfect Matching Problem (blossomV) and example code for reading in a graph and calling the solver is provided, so you can start implementing the algorithm right away. Included is also a file `README` that contains further important information.

(64 points)

Deadline: December 22, via email to armbruster@or.uni-bonn.de.