INTRO TO APPROXIMATION – HW3 **a taste of application**

Every task is worth two points. Deadline: **30. 1. 2017 08:00** via email. Please send source code and documentation (see below). Do not be afraid to email me or ask me if any task is unclear to you.

All four tasks are concerned with the symmetric travelling salesman problem on a metric space from a practical point of view. Our data set is a database of 6253 towns and municipalities in the Czech Republic, located here: `https://github.com/33bcdd/souradnice-mest/` in the file `souradnice.csv`. We will think of the coordinates as points in a two-dimensional Euclidean space.

Rules: Your task is to produce a program or a set of programs which will compute TSP approximations and other TSP-related computations. You are free to use any programming language that can be compiled on Linux (Csharp, Java and C++ are explicitly allowed, as well as anything else that can be found in the MFF computer lab).

You are free to make use of some third party libraries. To be specific, you can use any third party implementation of algorithms for shortest path, minimum spanning tree and perfect matching of minimum cost. You can also make use of any linear programming solver you want. However, for exercises 1-3, you are *not* allowed to use any library function or command-line tool which does the entire task by itself.

Your submitted programs should be able to take the CSV file as above, or any similar file (like a list of cities in Slovakia with the same CSV format) and output (into the standard output or into a file) what is required. There is no strict time limit; still, your programs should be able to finish within an hour on a modern computer (but ideally they should be much faster).

Your homework should also include a basic amount of documentation – namely what should I do (on Linux) to build your program and how to run it. If you use any 3rd party libraries or tools, you must disclose the fact and include a reference in the documentation!

Since there is no limitation on programming language or tool, you may run into problems like "I double checked everything but the program is just very slow". Contact me if any such problems arise.

Evaluation: I will first make sure your program works and it produces numbers which correspond to my results. Then, I will use a plagiarism-detection software to compare your source code with anyone else's as well as some selected solutions I found on the internet and Stack Overflow.

What this means that you should avoid plagiarism at all costs! Do not send your solution to anyone else; letting somebody copy your work is also a form of plagiarism. Some of you may feel this is too strong of a warning but the software caught people in the past and it was an unpleasant experience for everyone involved.

Good luck!

EXERCISE ONE Write an implementation of the 2-approximation algorithm for symmetric TSP that uses doubling of the minimum spanning tree. Apply it to the input above and write the total length of your tour as well as the Hamiltonian cycle on the output.

EXERCISE TWO Write an implementation of the Christofides 3/2-approximation algorithm for symmetric TSP. Apply it to the input above, write the total length and the Hamiltonian cycle on the output.

EXERCISE THREE In Homework 1, Exercise 5, we have learned of the standard linear programming relaxation for symmetric TSP. It is certainly possible to compute it in polynomial time, but it needs to be done carefully as there are constraints for all subsets of vertices.

Instead, your task is to use a linear programming solver to compute the solution to the following linear program:

$$\min \sum_{e \in E} d_e x_e$$

$$\forall v \in V: \qquad \sum_{e=vx} x_e = 2$$

$$\forall e \in E: \qquad 0 \leq x_e \leq 1$$

This program finds a fractional relaxation of the shortest cycle cover of the graph.

On output, write the value of the optimum as well as values of the edge variables in the format that your LP solver produces.

EXERCISE FOUR Use a modern TSP solving tool of your choice to find the optimum TSP tour on the first 2000 towns in the list. Write its length as well as the order on output.

Why 2000: Modern computers should be able to find the entire optimal TSP tour on 6253 vertices. The number was chosen so that it doesn't overwhelm your home computer.