Course 02249, DTU Informatics
Carsten Witt

Computationally Hard Problems – Fall 2010
Assignment Project

Date: 12.10.2010, **Due date**: 09.11.2010, 10:15

This project counts for three weekly assignments. It should be performed in groups consisting of either two or three students (3 is a hard maximum).

The following exercises are **mandatory**:

Exercise Project.1: Consider the following problem.

Problem: [LazyTSP]
Input:

- a number $n \in \mathbb{N}$,
- a matrix $d\colon (i,j) \to \mathbb{N}$, $1 \le i, j \le n$;
- a number $B \in \mathbb{N}$,
- two numbers $k, m \in \mathbb{N}$, where $n-k-m \ge 2$.

Output: YES if there is a sequence $i_1 = 1, i_2, \dots, i_{n-k-m}$ of pairwise distinct positive integers such that

$$\left(\sum_{j=1}^{m} d(j,j+1)\right) + \left(\sum_{j=1}^{n-k-m-1} d(i_j+m, i_{j+1}+m)\right) + d(i_{n-k-m}+m, 1) \;\le\; B \qquad (*)$$

Otherwise output NO.
Note: The set $\mathbb{N}$ is defined as $\{1,2,\dots\}$ and does not contain 0.

What you have to do:

a) Read and understand the problem. Describe in colloquial terms what the problem is about and explain the main differences to the classical TSP.

b) Show that LazyTSP is in $\mathcal{NP}$.

c) Show that LazyTSP is $\mathcal{NP}$-complete. What can you say about the complexity of the problem if you know that $k = n - a$ or $m = n - a$ for some $a \in \mathbb{N}$, which does not depend on n?

d) Find an algorithm which always gives the correct answer for an input to the LazyTSP, i. e., which always stops and replies YES if it is given a YES-instance and NO otherwise. The algorithm is allowed have exponential worst-case running time but may use “smart” techniques to deal faster with some instances. In case of a YES, your algorithm has to construct a solution $i_1, i_2, \ldots, i_{n-k-m}$ in accordance with the problem definition. Finally, extend your algorithm in order to solve the optimizing version of the problem, i. e., it should find a solution $i_1, i_2, \ldots, i_{n-k-m}$ for which B is as small as possible.

   Describe in words how the algorithm works.

e) Prove the worst-case running time of your algorithm for the optimizing version.

f) There is a collection of benchmark inputs to the classical TSP on the web, see `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`. These instances are stored in files following the `.tsp` file format, see the documentation on the homepage.

   Implement the algorithm you developed in Part d) and run it on certain `.tsp` instances, treated as inputs to the optimizing version of LazyTSP. Your software should at least support those `.tsp` files where `TYPE: TSP` and `EDGE_WEIGHT_TYPE: EUC_2D` holds, see the file `berlin52.tsp` for an example. Solutions should be output along with their costs according to Formula (∗). Test how the cost of the solution varies as you increase the parameters k and m.

   Instead of developing your software from scratch, you may build upon existing software packages, provided all legal restrictions are obeyed.

The program including the source code and an instruction how to execute it on a `.tsp` file has to be delivered to the teaching assistant. Accepted programming languages are Java, C, C++, C#. Other languages have to be agreed upon with the teaching assistant.
Your programs will be run on some `.tsp` files.

The three blocks [a),b),c)], [d),e)], and [f)] have approximately equal weights in the grading.

End of Exercise 1