TU Berlin
Department of Mathematics
Prof. Dr. Martin Skutella
Daniel Dressler

Complexity theory and fixed parameter algorithms
Winter term 2008/9

Exercise XII

Problem 1

Consider the Vertex Cover problem. Given an undirected graph $G = (V, E)$ and $k \in \mathbb{N}$, does G have a vertex set $C \subseteq V$ of size at most k such that every edge $e \in E$ is incident to at least one vertex in C? A trivial algorithm for this problem checks all $\mathcal{O}(|V|^k)$ sets.

a) Come up with a search tree algorithm that branches on the edges. Show that the search tree needs at most $\mathcal{O}(2^k)$ nodes. Give a bound for the running time of this algorithm.

b) The input data can also be preprocessed by considering vertices of degree at least $k+1$. What can be said about the size of the resulting input graph?

Problem 2

Consider the Euclidean 2-dimenstional TSP. In this setting, each city $i \in \{1, \dots, n\}$ has coordinates $c_i = (x_i, y_i)$ and the distance $d(i, j)$ between two cities is exactly the Euclidean distances of the corresponding coordinates. Naturally, the Euclidean TSP has a lot of geometric properties that can be used to design algorithms. In particular, in an optimal solution none of the chosen edges intersect (assuming all coordinates are in general position, and not, for example, on a common line).

Let V^{bd} be the set of cities that lie on the border of the convex hull of the coordinates,

$$V^{bd} := \{i \in V : c_i \in \mathrm{bd}(\mathrm{conv}\{c_1, \dots, c_n\})\}.$$

Also, let $V^{in} := V \setminus V^{bd}$ and $k := |V^{in}|$. We assume that $V^{bd} = \{1, \dots, n-k\}$ and $V^{in} = \{n-k+1, \dots, n\}$. Note that V^{bd} can be determined in time $\mathcal{O}(n \log n)$ and we assume this has already been done and the vertices in V^{bd} are sorted according to a cycle around the convex hull.

a) Suppose $V^{bd} = V$, that is, all cities lie on the border of their convex hull. What is the optimal solution in this case?

b) Now come up with a dynamic programming algorithm for the general case that depends on k but not so much on n. More precisely, this algorithm should run in $\mathcal{O}(2^k k^2 n)$.