Homework Exam 1, Geometric Algorithms, 2017

Distributed on May 11, 2017. Hand in **on paper** on May 23, at 13.15 (start of class, or earlier if you don't come to class), but start on solving the questions as early as possible. Give yourself the time to think about the questions and their solutions.

Be precise, correct, and succinct (but complete) in your explanations, algorithms, and running time analyses. You do not need to search for papers with similar problems, all questions can be answered by thinking and using or modifying the methods from the textbook on computational geometry. You do not have to copy whole code or proofs from that book, you can simply refer to them if you need it literally. But you must then refer precisely. Use proper notation and terminology in your solutions, the same or similar to the notation in the book. Never change notation that was given in the assignment itself (you immediately get points subtracted for this). Produce a carefully prepared document with your solutions, with proper type-setting (preferably with latex), and suitable figures. Adopt the style, proof detail, etc., of the textbook when answering the questions.

Geometric tests or constructions with constantly many elements are written in plain text ("If p lies strictly above the line ℓ ...", or "Let p be the leftmost intersection point of circle C and line ℓ."). Hand in the print-out of your solutions (with your name!). **It is essential that you spend time on careful formulations, consistent notation, succinctness, anything to make my life easy ;-).** This is a homework exam; you have plenty of time for this. We can grade fast if you provide clean and clear answers. Besides, clean and clear answers are good for your grade as well.

This homework exam is 25% of the final grade. If your score is below 5, you will have to do an additional (third) homework exam later during the course. Furthermore, if you also fail the second homework exam (below 5), you do not get to make this third homework exam; you fail the course directly. (If you participate in the course, you *really* participate.)

You may not collaborate with other students, although you may discuss an approach on a high level in an initial stage. Thinking about the technical details and the complete write-up must be done individually. You may ask questions or ask for hints (no later than May 19, at 17.00), and this will not influence your score for this exam. You get a hint only if you demonstrate that you have tried to solve the problem yourself already. You can ask questions to both lecturers Wouter and Marc.

You can earn 9 points for the answers to the questions. The 10th point is earned by a careful preparation of your hand-in: using a good layout, good spelling, good figures, no sloppy notation, no statements like "The algorithm runs in $n \log n$." (forgetting the $O(..)$ and forgetting to say that it concerns *time*), etc. Additional bonus points might be given if a solution is particularly good.

1. (1 point) Write pseudo-code for the following operation on a doubly-connected edge list: Given a vertex (object) v, return the number of *different* faces that are incident to v. Use the proper, agreed names of the doubly-connected edge list as in the textbook. Before giving the pseudo-code, first describe in a few sentences what your code does. Then give the pseudo-code.

   Analyze the efficiency of the solution and express the running time in the degree d of the vertex v and/or the number n of vertices in the whole subdivision represented by the DCEL.

2. (1 point) Write pseudo-code for the following operation on a doubly-connected edge list: Given a half-edge $\vec{e}$, report the coordinates of the vertex with the lowest x-coordinate in the DCEL which $\vec{e}$ is part of. Again, first describe in a few sentences what your pseudo-code does.

3. Let P be a set of $n \geq 3$ points in the plane. No two points of P coincide, and no three points of P lie on a single line; you may assume this in this whole question. We define the concept of a dented convex hull: a *dented convex hull of* P is a simple polygon Q whose vertices are points of P, all points of P lie in Q or on the boundary of Q, and at most one vertex of Q is a point of P that does not lie on the convex hull.

   (a) (1 point) Prove that if not all points lie on the convex hull, then there always is a dented convex hull with smaller area than the convex hull itself.

   (b) (1 point) Show that a set P of n points admits $\Theta(n^2)$ different dented convex hulls in the worst case. Recall that a worst-case $\Theta(..)$ bound represents a worst-case $O(..)$ bound and a worst-case $\Omega(..)$ bound.

   (c) (2 points) The *optimal* dented convex hull is the one with minimum area. Give an algorithm to compute the optimal dented convex hull and analyze its running time. Efficiency is good, but it is more important to make sure that your solution is correct.

4. Let S be a set of n line segments in the plane. Let k denote the number of intersection points among line segments in S. Assume that for every intersection point there are just two line segments of S that pass through it, and there is no pair of line segments that have more than a single point in common.

   We are interested in computing a shortest horizontal 3-stabber: a horizontal line segment that intersects at least three line segments of S and is as short as possible.

   (a) (1 point) Analyze what different cases can occur for the shortest horizontal 3-stabber, including degenerate ones that were not excluded in the problem statement above.

   (b) (1 point) Imagine a plane sweep algorithm to solve this problem. The initialization, the status, the status structure, and the event list can be exactly the same as in Chapter 2 of the textbook. Only the events need to be handled differently.

   Describe in sufficient detail how the events are handled.

   (c) (1 point) Assume that we are interested in the shortest horizontal m-stabber, where m is a parameter between 3 and n. Explain how the algorithm should be adapted to solve the shortest horizontal m-stabber problem, and analyze the running time, expressing it in n, k, and m (or possibly a subset of these).