Homework Exam 1, Geometric Algorithms, 2010/2011

Distributed on Wednesday, February 16. Hand in on paper on Thursday, February 24, 2011, at 13.15 (start of class, or earlier if you don’t come to class). Be precise, correct, and succinct (but complete) in your explanations, algorithms, and running time analyses. Use proper notation and terminology. Produce a carefully prepared document, with proper type-setting. If your hand-writing is very clear, you may write and hand in on paper. Otherwise, hand in a print-out or a pdf by e-mail (preferably made using Latex) to marc@cs.uu.nl.

This homework exam is 25% of the final mark. If your score is below 5, you will have to do a similar homework exam but one that is more work and also more difficult, as a second chance.

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 to the lecturer (no later than February 22, at 17.00), and this will not influence your score for this exam.

1. (3 points) The algorithm given in Chapter 1 of the book to compute the convex hull of a set of points computes the upper convex hull and the lower convex hull separately. You could argue that this is not such an elegant solution. In the exercise we consider a variation where the whole convex hull is computed in the same style, but in once. The idea is to first determine the bottommost point $p$ of $S$. Then we sort all other points of $S$ around $p$ by angle. Let this sorted order be $p_1, \ldots, p_{n-1}$, in clockwise order seen from $p$. In other words, the vector $\vec{pp}_1$ makes the largest angle with the vector $(1,0)^T$, among all vectors $\vec{pp}_1, \vec{pp}_2, \ldots, \vec{pp}_{n-1}$.

Now we put $p_1$ and $p_2$ in a list, and use a for-loop very similar to the one on page 6 of the book, lines 3–6.

(a.) Which degenerate cases do you want to ignore at first?
(b.) Give the pseudo-code for this algorithm, ignoring those degenerate cases.
(c.) Argue that your algorithm returns the convex hull correctly.
(d.) Explain how the previously ignored degenerate cases should be handled.

2. (3 points) A triangulation of a point set $S$ is a maximal planar subdivision that has a vertex for every point in $S$, and no other vertices. Since a triangulation is a planar subdivision, it can be stored in a doubly-connected edge list (DCEL). You may assume that not all points of $S$ lie on a single line (which would be a highly degenerate case), and no two points of $S$ coincide.

Write pseudo-code, using the default pointer names like IncidentFace, etc., to perform the operation EDGE_SWAP($\vec{e}$). This operation takes a half-edge $\vec{e}$ of the DCEL as its argument. It tests if the edge $e$ that $\vec{e}$ (partially) represents can be removed from the subdivision and replaced by a different edge, restoring the property that the subdivision is a triangulation. If the test succeeds, the edge swap is also executed (in the DCEL of course).

You need not write code for a geometric test like: does $r$ lie to the right of the directed line through $p$ and $q$? This can be stated in normal English text. Make sure that your code is correct no matter what half-edge of the DCEL representation of the triangulation is passed to your code.
3. (4 points) Given a set $S$ of $n$ disjoint line segments in the plane and a set $C$ of $m$ disjoint circles in the plane. The circles can contain each other, as long as they don’t intersect (meaning: they don’t have any point in common).

We want to report all circles of $C$ that do not intersect any line segment of $S$. For this you will develop a plane sweep algorithm that does this efficiently.

(a.) Define the status of your plane sweep algorithm.

(b.) Describe the status structure and the event list.

(c.) Describe how events are handled.

(d.) Give the whole algorithm (which includes the initialization).

(e.) State and prove the running time of the algorithm.