CS 533 Computational Geometry **Spring Semester, 2015**

Homework 2 (version 1.1)

Assigned: Feb 10 *Due: Feb 24*

1. Let G be a combinatorial embedding with the *next* and *twin* function defined on half-edges. G does not have to be connected. Let G^* be its combinatorial dual, which uses the same *twin* function, and with $next^*(\overrightarrow{e}) = prev(twin(\overrightarrow{e}))$; here $prev()$ is the inverse permutation of $next()$. Prove that $(G^*)^* = G$.

2. Prove that any polygon admits a triangulation, even if it has holes. A *hole* is another polygon with all its vertices contained in the interior of the given polygon. Holes cannot intersect. To triangulate a polygon with holes, we must obtain a connected plane graph that contains as edges all the segments of all the polygons, and where the given polygon is the border of the outer face, every hole is the border of a face, and every other face is a triangle. What is the maximum number of triangles one can obtain, as a function of the total number of segments in the input polygons?

3. Prove or disprove: Any y-monotone polygon has a triangulation such that the graph obtained from the dual graph of the triangulation after removing the vertex corresponding to the outer face is a path (we proved in class that this graph is a tree).

4. To triangulate a set of points means giving a set of non-intersecting segments such that we obtain a plane graph that is triangulated except for the outer face, and the border of the outer face is the convex hull of the given set of points. Give a $O(n \log n)$ algorithm to triangulate n points. One idea is to use the known $O(n \log n)$-algorithm that triangulates a polygon.

5. Give an algorithm that computes in $O(n \log n)$ time a diagonal that splits a simple polygon with n vertices into two simple polygons each with at most $2n/3 + 2$ vertices. Hint: Use the dual graph of a triangulation (after removing the vertex corresponding to the outer face).