Assignment no. 5 (optional)

If you wish to have your assignment graded, submit it by June 10th, 2022. Feel free to submit solutions to any subset of the exercises.

Exercise 5.1 Let P be a set of n points in the plane. Give an $O(n \log n)$ time algorithm to find for each point p in P another point in P that is closest to p.

Exercise 5.2 To complete the sweep-line algorithm for computing the Voronoi diagram of points in the plane, write a procedure to wrap up a *valid* DCEL representation of the diagram within a sufficiently large bounding box. The box should contain all the sites and all the Voronoi vertices. The necessary information is available in the incomplete DCEL computed throughout the sweep as well as the status structure after all the events have been handled.

Exercise 5.3 The *Gabriel graph* of a set P of points in the plane is defined as follows: Two points p and q are connected by an edge of the Gabriel graph if and only if the disc with diameter pq does not contain any other point of P.
(a) Prove that the Delaunay graph of P contains the Gabriel graph of P.
(b) Prove that p and q are adjacent in the Gabriel graph of P if and only if the Delaunay edge between p and q intersects its dual Voronoi edge.
(c) Give an $O(n \log n)$ time algorithm to compute the Gabriel graph of a set of n points.
(CGAA Ex. 9.13)

Exercise 5.4 The *medial axis* of a polygon is the locus of centers of circles that are contained in the polygon and touch its boundary in at least two points. Give an efficient algorithm to compute the medial axis of a convex polygon. Analyze the running time of your algorithm.

Exercise 5.5 A Euclidean minimum spanning tree (EMST) of a set P of points in the plane is a tree of minimum total (Euclidean) edge length connecting all the points.
(a) Prove that the set of edges of a Delaunay triangulation of P contains an EMST for P.
(b) Use this result to give an $O(n \log n)$ time algorithm to compute an EMST for P.

Exercise 5.6 Given a simple polygon P with n vertices and a query point q, here is an algorithm to determine whether q lies in P. Consider the ray $\rho := \{(q_x + \lambda, q_y) : \lambda > 0\}$ (this is the horizontal ray starting in q and going rightwards). Determine for every edge e of P whether it intersects ρ. If the number of intersecting edges is odd, then $q \in P$, otherwise $q \notin P$. Prove that this algorithm is correct, and explain how to deal with degenerate cases. (One degenerate case is when ρ intersects an endpoint of an edge. Are there other special cases?) What is the running time of the algorithm?