CS 498 TC ✧ Computational Geometry ✧ Spring 2021

Homework 5

Due Wednesday, April 28, 2020 at 8pm

---

1. Suppose you are given a set S of n line segments in the plane. Describe and analyze an algorithm to find a line that intersects as many segments in S as possible. *[Hint: What is the dual of a line segment?]*

2. Let L be a set of n lines in the plane in general position.

   (a) Prove that $\sum_f \deg(f)^2 = O(n^2)$, where the sum is over all faces f in the arrangement of L, and $\deg(f)$ denotes the number of edges of faces f.

   (b) Prove that the arrangement of L contains $\Omega(n)$ bounded triangular faces.

3. Let P be a set of *moving* points in the plane, each represented by a starting position and a fixed velocity vector. For any real number t, a point with starting position (a, b) and velocity (u, v) is located at $(a + tu, b + tv)$ at time t. As the points in P move through the plane, their axis-aligned bounding box continuously changes.

   (a) Describe an algorithm to compute the time t when the bounding box of the moving points has smallest *perimeter*.

   (b) Describe an algorithm to compute the time t when the bounding box of the moving points has smallest *area*.

   [Hint: Consider the one-dimensional case first. The optimal time t could be negative!]

4. In class we saw the classical *funnel* algorithm to compute shortest paths inside a triangulated simple polygons. How would you modify this algorithm to find shortest paths in a polygon with holes?

   (a) Describe and analyze an algorithm to compute the shortest path between two given points in the interior of a polygon with *one* hole. *[Hint: Which way does the path go around the hole?]*

   (b) Describe and analyze an algorithm to compute the shortest path between two given points in the interior of a polygon with *two* holes.

   (c) **[Extra credit]** Describe and analyze an algorithm to compute shortest paths in a polygon with h holes; analyze your algorithm as a function of both n (the total number of polygon vertices) and h (the number of holes).

   In all cases, you can assume that you are given a triangulation of the input polygon. For full credit, your algorithms should run in $O(n)$ time (for any constant h).