Homework 6

Due by the start of class on Tuesday, 7. (Submissions will be through Gradescope.) Late homeworks are not accepted (unless an extension has been prearranged) so please turn in whatever you have completed by the due date. Unless otherwise specified, you may assume that all inputs are given in *general position*.

Problem 1. You are given a set of n sites $P = \{p_1, \ldots, p_n\}$ in the plane and a scalar $r > 0$. We say that two points s and t are *safely connected* if there exists a path from s to t such that every point along the path is within distance at most r of some point of P (see Fig. 1(a)). Note that if the distance from either s or t to its closest point in P exceeds r, then this pair is *not* safely connected.

Given P and r, present a data structure that answers safely-connected queries for any query pair (s, t) in time $O(\log n)$ time. Your data structure should use $O(n)$ space and should be constructable in $O(n \log n)$ time. Justify your data structure's correctness and derive its space and query time bounds. (**Hint:** You can use either Voronoi diagrams and/or Delaunay triangulations.)

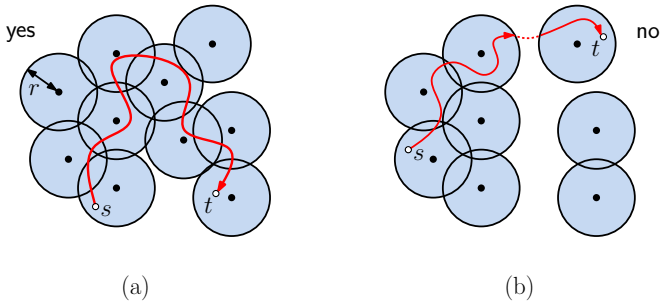

Figure 1: Answering safely-connected queries.

Problem 2. Let $P = \langle p_1, p_2, \ldots, p_n \rangle$ be the vertices of a *convex polygon*, given in counterclockwise order. The purpose of this problem is to develop a randomized incremental Delaunay triangulation algorithm that runs in $O(n)$ expected time.

Unlike the algorithm presented in class, we *do not* put all the sites within a large bounding triangle. We start with the triangle defined by three random sites from P. The remaining sites are inserted in random order. Each new site p is connected to the convex hull by adding its two two tangent edges, thus creating a new triangle. As in the standard randomized algorithm, an incircle test is performed on this triangle to determine whether it is locally Delaunay. If it fails, an edge flip is performed. We continue performing incircle tests and edge flips until all the newly created triangles are locally Delaunay (see Fig. 2).

(a) Show that the expected number of edge-flips performed with each insertion is $O(1)$.

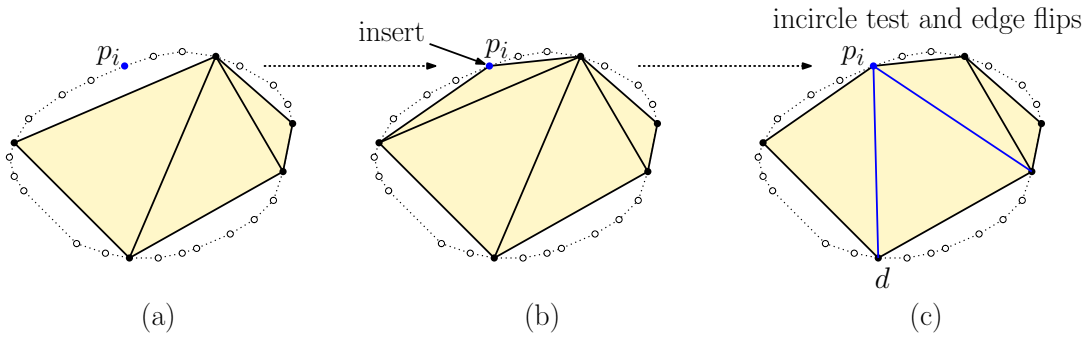

Figure 2: Delaunay triangulation in a convex polygon.

(b) Whenever a new site is added, we need to determine where along the current convex hull it is to be added. Explain how to do this in expected time $O(n \log n)$ over the entire algorithm. (Hint: Use bucketing.)

Problem 3. In machine learning it is often desirable to determine whether two point sets in the plane can be partitioned by a line. When this is not possible, the goal is to find line that achieves the best possible split. Let A and B be two point sets in the plane each having n points. Given a nonvertical line ℓ, let ℓ^+ and ℓ^- denote the halfplanes lying above and below ℓ, respectively. Define ℓ's *separation defect* (see Fig. 3) to be

$$\delta(\ell) \;=\; \min\big(|A \cap \ell^+| + |B \cap \ell^-|,\;\; |A \cap \ell^-| + |B \cap \ell^+|\big).$$

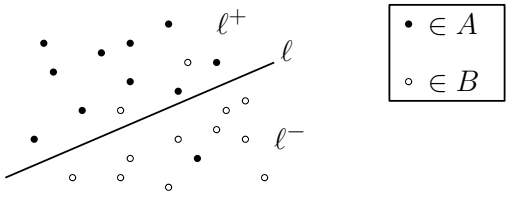

Figure 3: The line ℓ has separation defect 3, since there is one point of A below ℓ and two points of B above it.

Present an $O(n^2)$ algorithm that, given two n-element planar point sets A and B, computes a line ℓ that achieves the *minimum separation defect*. (There may generally be many. You can output any one of them. To avoid dealing with special cases in which ℓ passes through a point of A or B, you may restrict your algorithm to considering lines that do not pass through any point of either set.)

Problem 4. (Optional–Ungraded) Present $O(n^2)$ time algorithms for the following two problems. (The two solutions are closely related, so you can give one solution in detail and explain the modifications needed for the second one.)

(a) Given a set $S = \{s_1, \ldots, s_n\}$ of non-intersecting line segments in the plane, does there exist a (nonvertical) line ℓ that intersects none of the segments of S such that there is at least one segment of S lying above ℓ and at least one lying below? (See Fig. 4(a).)

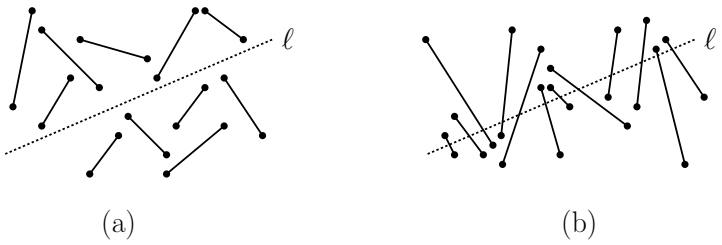

Figure 4: Stabbing segments.

(b) Given a set $S = \{s_1, \ldots, s_n\}$ of non-intersecting line segments in the plane, does there exist a line ℓ that intersects all of the segments of S? (See Fig. 4(b).)

(c) Part (b) looks very similar to a problem we have seen before of stabbing vertical segments. Provide an intuitive explanation as to what goes wrong if you were to try to adapt the LP-based solution to that problem to solve this problem.

Note: Challenge problems are not graded as part of the homework. The grades are recorded separately. After final grades have been computed, I may "bump-up" a grade that is slightly below a cutoff threshold based on these extra points. (But there is no formal rule for this.)

Challenge Problem: In class we argued that the number of parabolic arcs along the beach line in Fortune's algorithm is at most $2n - 1$. The goal of this problem is to prove this result in a somewhat more general setting.

Consider the beach line at some stage of the computation, and let $\{p_1, \ldots, p_n\}$ denote the sites that have been processed up to this point in time. Label each arc of the beach line with the index of its the associated site. Reading the labels from left to right defines a string (see Fig. 5).

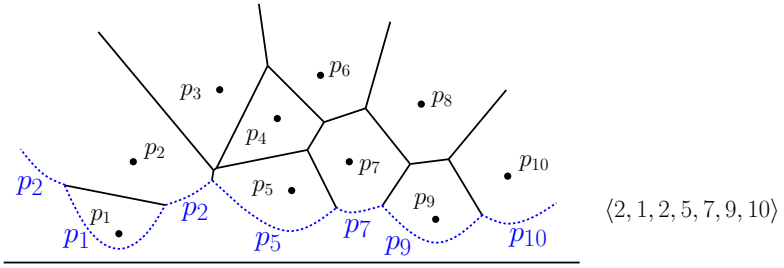

Figure 5: Beach-line complexity.

(a) Prove that for any two labels a, b, the following alternating subsequence *cannot* appear anywhere within such a string:

$$\ldots a \ldots b \ldots a \ldots b \ldots$$

(b) Prove that any string of n symbols that does not contain any repeated symbols ($\ldots aa \ldots$) and does not contain the alternating sequence of the form "$\ldots a \ldots b \ldots a \ldots b \ldots$" cannot

be of length greater than $2n - 1$. (**Hint:** Use induction on n. Fix any symbol, and split the string into three substrings based on this symbol's first and last occurrences. Analyze where the remaining symbols can occur, based on the alternation restriction.)

Sequences that contain no forbidden subsequence of alternating symbols are famous in combinatorics. They are known as *Davenport-Schinzel sequences*, and their growth properties are an interesting and open problem. They have numerous applications in computational geometry, this being one.