CMSC 754: Spring 2020 Dave Mount

Sample Problems for the Final Exam

The final exam will be released **Fri, May 15, 8:00am EDT** (on the class handouts page) and will be due through Gradescope upload by **Mon, May 18, 8:00am EDT**.

Disclaimer: The following problems have been collected from old homeworks and exams. They do not necessarily reflect the actual difficulty or coverage of questions on the final exam. The final will be comprehensive, but will emphasize material since the midterm.

In all problems, unless otherwise stated, you may assume general position, and you may use of any results presented in class or any well-known result from algorithms and data structures.

Problem 0. You should expect a problem that involves working through some algorithm covered in class on a given input set.

Problem 1. Give a short answer (a few sentences) to each question. Unless explicitly requested, explanations are not required, but may be given for partial credit.

(a) A *dodecahedron* is a convex polyhedron that has 12 faces, each of which is a 5-sided pentagon. Every vertex has degree 3. How many vertices and edges does the dodecahedron have? Show how you derived your answer.

(b) Given a set P of n points in the plane, what is the maximum number of edges in P's Voronoi diagram? (For full credit, express your answer up to an additive constant.)

(c) When the ith site is added to the Delaunay triangulation using the randomized incremental algorithm, what is the worst-case number of edges that can be incident on the newly added site? What can you say about the expected-case number of such edges (assuming that points are inserted in random order)?

(d) For each of the following assertions about the Delaunay triangulation of a set P of n points in the plane, which are True and which are False?

(i) The Delaunay triangulation is a t-spanner, for some constant t

(ii) The Euclidean minimum spanning tree of P is a subgraph of the Delaunay triangulation

(iii) Among all triangulations of P, the Delaunay triangulation maximizes the minimum angle

(iv) Among all triangulations of P, the Delaunay triangulation minimizes the maximum angle

(v) Among all triangulations of P, the Delaunay triangulation minimizes the total sum of edge lengths

(e) An arrangement of n lines in the plane has exactly n^2 edges. How many edges are there in an arrangement of n planes in 3-dimensional space? (Give an exact answer for full credit or an asymptotically tight answer for half credit.) Explain briefly.

(f) Given an n-element point set P in $\mathbb{R}^d$ and a scalar $s \geq 1$, it is known that an s-WSPD consists of $O(s^d n)$ pairs. Given any point $p \in P$, what is the *maximum* number of pairs $\{P_u, P_v\}$ of the decomposition such that either $p \in P_u$ or $p \in P_v$. (Express your answer as a function of s, d, and n. Explain briefly.)

Problem 2. You are given a set P of n points in $\mathbb{R}^2$. Present data structures for answering the following two queries. In each case, the data structure should use $O(n^2)$ space, it should answer queries in $O(\log n)$ time. (You do not need to explain how to build the data structure.)

(a) The input to the query is a nonvertical line ℓ. Such a line partitions P into two (possibly empty) subsets: $P^+(\ell)$ consists of the points lie on or above ℓ and $P^-(\ell)$ consists of the points of P that lie strictly below ℓ (see Fig. 1(a)). The answer is the maximum vertical distance h between two lines parallel to h that lie between $P^+(\ell)$ and $P^-(\ell)$ (see Fig. 1(b)).

For simplicity, you may assume that neither set is empty (implying that h is finite).

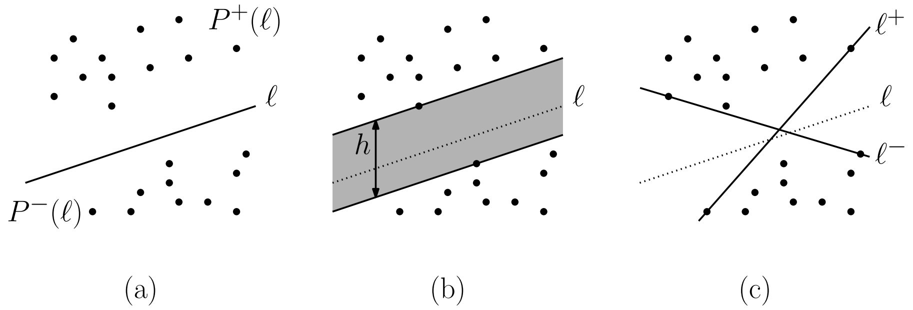

Figure 1: Separation queries.

(b) Again, the input to the query is a nonvertical line ℓ. The answer to the query consists of the two lines ℓ^- and ℓ^+ of minimum and maximum slope, respectively, that separate $P^+(\ell)$ from $P^-(\ell)$ (see Fig. 1(c)). You may assume that $P^+(\ell)$ from $P^-(\ell)$ are *not* separable by a vertical line (implying that these two slopes are finite).

Problem 3. You are given a set P of n points in the plane and a path π that visits each point exactly once. (This path may self-intersect. See Fig. 2.)

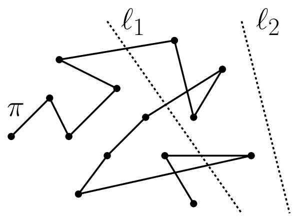

Figure 2: Path crossing queries.

Explain how to build a data structure from P and π of space $O(n)$ so that given any query line ℓ, it is possible to determine in $O(\log n)$ time whether ℓ intersects the path. (For example, in Fig. 2 the answer for ℓ_1 is "yes," and the answer for ℓ_2 is "no.") (Hint: Duality is involved, but the solution requires a bit of lateral thinking.)

Problem 4. Consider the following two geometric graphs defined on a set P of points in the plane.

(a) *Box Graph:* Given two points $p, q \in P$, define box(p, q) to be the square centered at the midpoint of $\overline{pq}$ having two sides parallel to the segment $\overline{pq}$ (see Fig. 3(a)). The edge (p, q) is in the box graph if and only if box(p, q) contains no other point of P (see Fig. 3(b)). Show that the box graph is a subgraph of the Delaunay triangulation of P.

(b) *Diamond Graph:* Given two points $p, q \in P$, define diamond(p, q) to be the square having $\overline{pq}$ as a diagonal (see Fig. 3(c)). The edge (p, q) is in the diamond graph if and only if diamond(p, q) contains no other point of P (see Fig. 3(d)). Show that the diamond graph may not be a subgraph of the Delaunay triangulation of P. (Hint: Give an example that shows that the diamond graph is not even planar.)

Problem 5. You are given a set of n sites P in the plane. Each site of P is the center of a circular disk of radius 1. The points within each disk are said to be *safe*. We say that P is *safely connected* if, given any $p, q \in P$, it is possible to travel from p to q by a path that travels only in the safe region. (For example, the disks of Fig. 4(a) are connected, but the disks of Fig. 4(b) are not.)

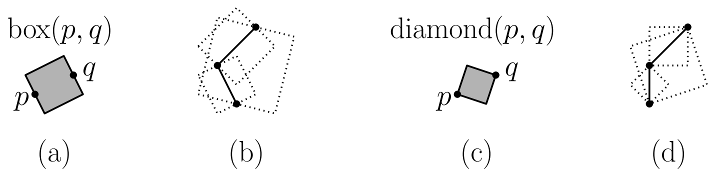

Figure 3: The box and diamond graphs.

Present an $O(n \log n)$ time algorithm to determine whether such a set of sites P is safely connected. Justify the correctness of your algorithm and derive its running time.

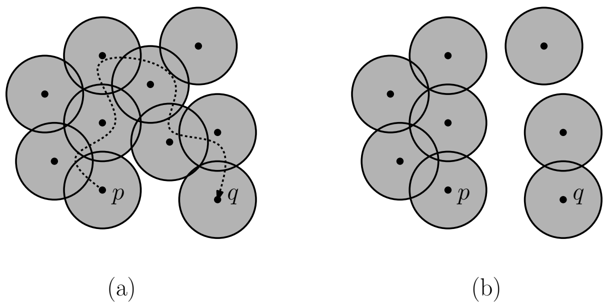

Figure 4: Safe connectivity.

Problem 6. 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 its the associated site. Reading the labels from left to right defines a string. (In Fig. 5 below the string would be $p_2 p_1 p_2 p_5 p_7 p_9 p_{10}$.)

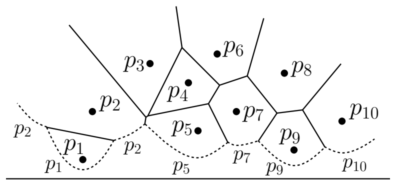

Figure 5: Beach-line complexity.

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

$$\ldots p_i \ldots p_j \ldots p_i \ldots p_j \ldots$$

(b) Prove that any string of n distinct symbols that does not contain any repeated symbols $(\ldots p_i p_i \ldots)$ and does not contain the alternating sequence[1] of the type given in part (a) cannot be of length greater than $2n-1$. (Hint: Use induction on n.)

[1]Sequences that contain no forbidden subsequence of alternating symbols are famous in combinatorics. They are known as *Davenport-Schinzel sequences*. They have numerous applications in computational geometry, this being one.

Problem 7. Given a set of n points P in $\mathbb{R}^d$, and given any point $p \in P$, its *nearest neighbor* is the closest point to p among the remaining points of P. Note that nearest neighbors are not reflexive, in the sense that if p is the nearest neighbor of q, then q is not necessarily the nearest neighbor of p. Given an approximation factor $\varepsilon > 0$, we say that a point $p' \in P$ is an *ε-approximate nearest neighbor* to p if $\|pp'\| \leq (1+\varepsilon)\|pp''\|$, where p'' is the true nearest neighbor to p.

Show that in $O(n \log n + (1/\varepsilon)^d n)$ time it is possible to compute an ε-approximate nearest neighbor for every point of P. Justify the correctness of your algorithm. Hint: This can be solved using either WSPDs or spanners.

Note: There exists an algorithm that runs in $O(n \log n)$ time that solves this problem exactly, but it is considerably more complicated than the one I have in mind here.