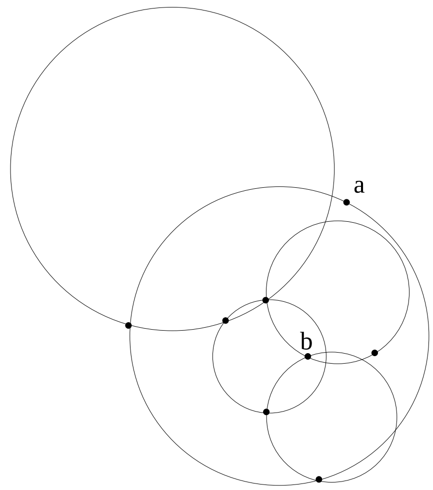
Another Practice Final

(1). [15 points] For the set S of 8 points shown below, do the following:

(a). Draw the (Euclidean) Delaunay diagram. In order to assist you in making some decisions (in case you do not have a compass with you), I have drawn a couple circles.



(b). Sketch also the Voronoi cells for points a and b. (You need not draw the entire Voronoi diagram.)

How many sides does each cell have?

(c). Draw the furthest-site Delaunay diagram (dual to the furthest-site Voronoi diagram)

(2). [20 points] For each of the computations below indicate how efficiently one can perform the calculation, in terms of $O(...)$ notation (e.g., $O(n)$, $O(\log n)$, $O(n^2)$, $O(n \log n)$, $O(k \log n)$, etc). Try to give the best (lowest) upper bound possible. If a bound uses a term (e.g., "k") that is not part of the input complexity, then be sure to define it. **For full credit, you must give a brief justification of each answer. (One sentence should suffice.)**

DO ANY 4 OF THE FOLLOWING 5 PROBLEMS. Clearly indicate which 4 you want to be counted, or I will simply pick (a)-(d).

(a). Given an arbitrary set of n triangles $\{T_1, \dots, T_n\}$ in the plane (in general position), such that no two triangles have edges that intersect (so the $3n$ edges are pairwise non-crossing), determine if there exists some triangle T_i fully contained in some other triangle T_j ($j \neq i$).

(b). Compute a triangle of minimum area whose three vertices come from a given set of n points in the plane.

(c). Given the minimum spanning tree, $MST(S)$, of a set S of n points in the plane (given in, say, a DCEL), compute the convex hull of S and compute a triangulation, $\mathcal{T}$, of $CH(S)$ that respects the edges of $MST(S)$ (i.e., each edge of $MST(S)$ is an edge of $\mathcal{T}$).

(d). Given a simple polygon P with n vertices, preprocess P so that one can quickly answer for a query point $q \in \Re^2$ if q is (i) inside P; or (ii) inside $CH(P)$ but outside P; or (iii) outside $CH(P)$. (Give the preprocessing time and query time.)

(e). Given a set S_1 of n points in $\Re^2$ and a set S_2 of m points in $\Re^2$, determine if there exists a circle, C, for which all of S_1 lies inside C and all of S_2 lies outside C.

(3). [25 points] For each of the following statements, state whether it is **ALWAYS TRUE**, **SOMETIMES TRUE** (but sometimes false), or **NEVER TRUE**. Briefly justify your answer, either with an example or with a very brief explanation.

Specifically, if it is sometimes true, sometimes false, show examples of *each* case, and try to suggest when it is true and when it is false. If it is always true or always false, state why (briefly).

DO ANY 5 OF THE FOLLOWING 7 PROBLEMS, (a)–(g). Clearly indicate which 5 you want to be graded, or I will simply pick (a)-(e).

(a). Assume that the Delaunay diagram of a set of points S in the plane is a triangulation. Assume that the points S are in general position (no three are collinear, no four are cocircular). Then, if pqr and pqs are adjacent triangles, sharing Delaunay edge pq, then the segment rs crosses the segment pq.

(b). For a set of n distinct points in the plane, the Voronoi edges and Voronoi vertices (not counting the "point at infinity" as a Voronoi vertex) form a connected graph.

(c). If $\{p_1, \dots, p_k\}$ is a set of distinct collinear points in the plane, then the duals of these k points are parallel to each other (using the standard definition of duality from the text).

(d). If (p, q) is a Delaunay edge, then the line segment pq crosses the Voronoi edge that is shared by the cells $V(p)$ and $V(q)$.

(e). Let S be a set of points in the plane, no 4 of which are cocircular and no 3 of which are collinear. Let abc be the triple of three point of S that minimizes the area of Δabc. Then, Δabc is a Delaunay triangle.

(f). When the Bentley-Ottmann sweep algorithm is run on a set of n line segments in the plane that have at least one crossing point, the first crossing point found by the algorithm is the leftmost such point. (Assume that we use a vertical sweep line from left to right.)

(g). Let P be a simple polygon having $n \geq 10$ vertices, such that P has a unique triangulation. Then, P is star-shaped (i.e., 1-guardable).

DO ANY 2 OF THE FOLLOWING 3 PROBLEMS: (4), (5), (6). Clearly indicate which 2 you want to be graded!

(4). [10 points] We want to solve the following query problem: Given a set $\mathcal{P}$ of n disjoint line segments in the plane, determine if a query line ℓ intersects all n of the segments (in which case we call it a "stabber").

(a). Describe briefly a data structure and a method for this problem. Try to be as efficient as possible in both space and query time.

(i). Preprocessing time is $O(\qquad)$
(ii). Storage space (memory usage) is $O(\qquad)$
(iii). Query time is $O(\qquad)$

(b). Assume now that we know that ℓ is always axis-parallel (horizontal or vertical). How efficiently now can the problem be solved? (Give a brief justification.)

(i). Preprocessing time is $O(\qquad)$
(ii). Storage space (memory usage) is $O(\qquad)$
(iii). Query time is $O(\qquad)$

(5). [10 points]

(a). Given a set $\mathcal{C}$ of n unit disks in the plane (arbitrarily overlapping), explain briefly how to build a data structure to support efficient queries of the form: For a query point q, determine if q lies inside some disk of $\mathcal{C}$ and, if so, report one such disk. State the preprocessing time, storage space, and query time.

(i). Preprocessing time is $O(\qquad)$
(ii). Storage space (memory usage) is $O(\qquad)$
(iii). Query time is $O(\qquad)$

(b). Given a set S of n points in the plane, explain briefly how to build a data structure to support efficient queries of the form: For a query point q and a positive real number r, report all points of S that are within L_∞ distance r of q. State the preprocessing time, storage space, and query time. (Reminder: $d_\infty(p,q) = \max\{|p_x - q_x|, |p_y - q_y|\}$.)

(i). Preprocessing time is $O(\qquad)$
(ii). Storage space (memory usage) is $O(\qquad)$
(iii). Query time is $O(\qquad)$

(6). [10 points] We have studied a few randomized incremental algorithms during the semester, in each case doing some of the analysis based on the "backwards analysis" technique. Pick one randomized incremental algorithm and describe it briefly. Give enough of the backwards analysis of the expected running time of the algorithm to demonstrate that you understand the "punch line" of the method.

DO ANY 2 OF THE FOLLOWING 3 PROBLEMS: (7), (8), (9). Clearly indicate which 2 you want to be graded!

(7). [10 points] A car race is organized on an infinitely long straight road. The n cars start out in different locations and travel at constant speeds (possibly different for each car) in the same direction. The cars immediately in front and behind a given car c are called its *neighbors*.

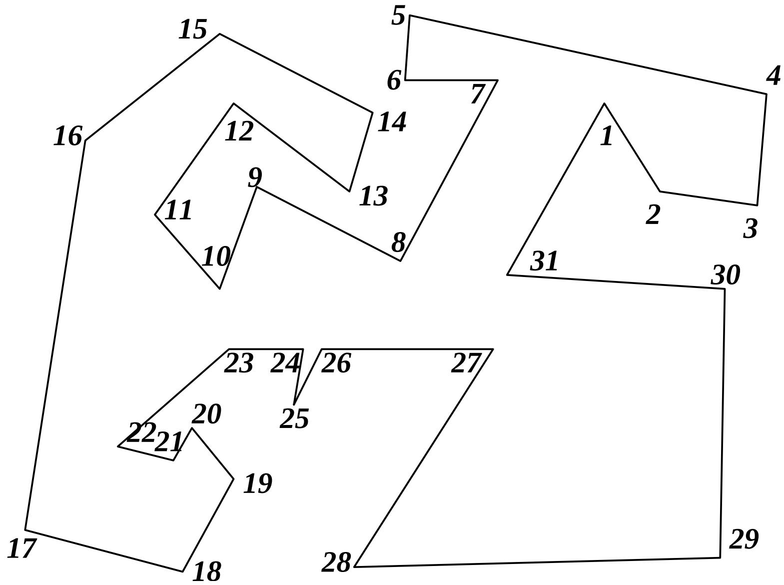
(a). For a particular car, c, give an upper bound (in big-Oh terms of n) on the maximum number of times the identities of its neighbor cars can change, during the entire duration of the race and briefly justify. (Assume that each car has its own lane along the road, so cars can freely pass each other without collision.)

(b). Let K be the maximum number of cars that are ever at the same position along the road (side-by-side, since they are in separate lanes). How efficiently can K be computed?

(8). [10 points] Let S be a given set of n axis-parallel rectangles in general position in the plane. Our goal is to compute very efficiently the *depth*, $\max_{p \in \Re^2} depth(p)$, of the set S, where $depth(p)$ is the number of rectangles of S that contain p. (We are computing the overall "depth" of the set S, not just the depth at any one point p.) Describe in some detail how a sweep algorithm can compute the depth efficiently. (Hint: a segment tree may be useful.) What is the running time and what is the space used (in big-Oh)?

(9). [10 points] For the simple polygon P below, do the following:

(a). Obtain the *point* guard number for P, allowing guards to be placed at *any* point (interior or boundary) of the polygon. Justify your answer! (Give an argument that fewer guards cannot suffice.) Show the guards in an optimal guard set.



(b). Obtain the *vertex* guard number for P. Justify your answer! (Illustrate on the figure below.)